AN INFORMATION THEORY APPROACH TO SUPERVISED LEARNING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Maya Rani Gupta
March 2003

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Robert M. Gray
(Principal Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Richard Olshen
(Statistics)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Robert Tibshirani
(Statistics)

Approved for the University Committee on Graduate Studies:

Abstract

Supervised learning algorithms classify or estimate test points based on labelled training samples. Learning algorithms have been applied in diverse ares of engineering, including speech recognition, damage detection, and document analysis. Two difficulties in learning are the 'curse of dimensionality' and bias arising from the distribution of training samples.

In this thesis, a new nonparametric algorithm for supervised classification or estimation is presented. The algorithm extends linear interpolation using the principle of maximum entropy, and is termed LIME. Compared to other nonparametric methods, LIME is shown to ameliorate difficulties arising in high dimensions or from asymmetrical distributions of training data. Asymptotic theoretical results are shown, as well as noise robustness and analytical forms for LIME solutions. Simulations show that error rates are in some circumstances lower than those of other nonparametric algorithms, discriminant analysis methods, neural nets, regularized linear regression, and decision trees. The problem of supervised learning based on grids of training samples is considered in-depth. Application of LIME to color management and gas pipeline integrity are demonstrated. LIME is computationally more expensive than standard nonparametric algorithms, but the improvement in error rates may be a worthwhile trade-off. LIME may also be a valuable component in a hybrid classification system.

Acknowledgements

This dissertation is, of course, only a physical manifestation of the long intellectual journey that constitutes graduate school. The acknowledgements here reflect the process of becoming as well as this particular product. I apologize for any appreciations left unnoted.

Thanks go to James Harris, who was a hero to chair my defense at the last moment. I would like to thank Rob Tibshirani for being a great assistant adviser and serving on my defense and reading committee. Many thanks go to Richard Olshen, who was encouraging and patient with my mathematical efforts, and served as a defense and reading committee member. This year, Bob Gray was nominated for a Presidential Award for Excellence in Mentoring by the National Science Foundation, and I am honored to have had him as my primary adviser. In particular, I appreciated the independence he granted me, his wisdom, encouragement, and speedy revisions.

Behind the scenes of this thesis were the editing efforts, latex advice, and general positivity of Deirdre O'Brien; a 24 hour optimization helpline operated by Michael Friedlander; software and computer support from Ken Lin, Remco Teunen, and Mario Parente; counseling on convergence from Vincent Vanhoucke; and the efficient administrative assistance of Kelly Yilmaz.

I would like to thank Lorne Campbell for providing a key example and insight that clarified the procedure for finding an analytic solution to the l_1 LIME minimization problem.

Thanks and appreciation to everyone who inspired, mentored, advised, and showed me how during my graduate school years, including (but certainly not limited to) Rich Baraniuk, Anna Gilbert, Doug Abraham, Dan Mittleman, Amir Najmi, Warren Fox, Tom Cover, Elza Erkip, Trevor Hastie, Rob Wilson, Sidney Burrus, Vivek Goyal, Don Johnson, Bishnu Atal, Phil Chou, Zakkula Govindarajulu, and Anna Friedlander.

Thanks to all the folks at Ricoh's California Research Center for their suggestions, advice, and flexibility, especially Kathrin Berkner, Martin Boliek, Michael Gormish, Peter Hart, and David Stork.

The National Science Foundation provided generous financial support and gave me the confidence to go to graduate school. Additional financial support was kindly provided by Hewlett Packard, Ricoh, and Norsk Elektro Optikk.

Much thanks to my family for their continuous positive encouragement and asking hard questions.

Thanks also to Allison Marino and Al Luckow, for providing a place to hide out and get things done, complete with grand piano, wine, and wood-chopping.

Lastly, thanks go to Matt Swihart, who showed me how to think beyond my expectations.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

What has been is what will be...

Ecclesiastes 1:9

The objective of supervised learning is to estimate unknown quantities based on observed samples. For example, one may have samples of pollutant levels from certain points throughout Manhattan. Based on those samples, one would like to estimate the pollutant strength at non-sampled locations in the city. This is a problem of regression or numerical estimation. If the observation to be estimated is discrete or categoric, the problem is one of classification. An example is determining if incoming email is 'spam' or 'legitimate correspondence.'

Algorithms for supervised learning are useful tools in many areas of science and engineering, from estimating appropriate dosages of medicine for patients to predicting system failures. General references on supervised learning include [31], [54], [111], and [89]. Two reviews of the supervised learning literature which appeared recently are [78] and [59]. Supervised learning may be used as an end goal or as a preprocessing step for other systems. For example, classification of blocks of data into image or text might precede a document compression system that models the two categories differently. Estimation of likelihood of failure might establish priorities for a human safety evaluator.

In this dissertation, a new nonparametric method for supervised learning is presented which generalizes linear interpolation by using the principle of maximum entropy. In Chapter 1, the problem of supervised learning, and families of current solutions are reviewed. Certain aspects of supervised learning are not yet well-solved, particularly the curse of dimensionality and the bias that may occur from the distribution of training data. Traditional linear interpolation is shown to ameliorate these problems, and a generalization of linear interpolation is proposed for supervised learning. The generalization is termed 'linear interpolation with maximum entropy' (LIME). The LIME algorithm is presented in Chapter 2, and simulations are used to explore its behavior. The theoretical properties of LIME are shown in Chapter 3, including asymptotics, analytical form of the LIME weights, and robustness to noise. Chapter 4 takes an in-depth look at the problem of supervised learning with a regular grid of training samples. More real world examples and simulations are presented in Chapter 5. Chapter 6 moves beyond the proposed algorithm to take a look at the underpinnings of the principle of maximum entropy and discuss principles of inference in general. The concluding chapter summarizes advantages and disadvantages and considers future extensions of this work.

1.0.1 Philosophy and supervised learning

Supervised learning is a type of induction. Induction, as defined by John Stuart Mill, is *'that operation of the mind, by which we infer that what we know to be true in a particular case or cases, will be true in all cases which resemble the former in certain assignable respects.'* [88] A given training sampling of data could be taken from an infinite set of actual complete data sets. Estimates based on a given training sample are thus probabilistic inferences, never certain answers.

Deduction uses straightforward rules to reach a logical conclusion based on axioms or data. Induction makes statements about the unknown based on examples. Deduction comes with the guarantee of logic. Induction lacks this certainty. The philosopher David Hume criticizes induction in his *Enquiry Concerning Human Understanding* [57]. He shows, as should be clear, that the conclusions of induction are

not certain and not deductive. Cox moves beyond Hume's criticism to conclude, *'If we are willing to deal with probabilities rather than certainties and admit the rules of probable inference to the canon of reason, we should counterphrase this remark and say: If there be any possibility that the course of nature is uniform and that the past may be some rule for the future, all experience becomes useful and can give support to some inference.'* [26]. In fact, attributes often change slowly over time or space or can be correlated with other attributes. In many practical cases we can form probabilistic sets of inferred knowledge that do approximate the truth. However, it is circular (though self-consistent) to 'induce' that induction is useful in general because it has been seen to be useful in the past.

A method of induction can be viewed as a policy for estimating the unknown, and judged accordingly. Philosopher C. S. Peirce argued that this viewpoint of induction is justified [99], *'The validity of an inductive argument consists, then, in the fact that it pursues a method which, if duly persisted in, must in the very nature of things, lead to a result indefinitely approximating to the truth in the long run.'* Peirce's perspective has been supported by later thinkers in the field of probability, including Kneale [75], and this line of reasoning provides philosophical support to the pragmatic arguments for supervised learning. In particular, one may be interested in whether a method of estimation will converge to the expected conditional values in the limit of infinite training samples. Many supervised learning algorithms converge to the minimal probabilistic error in the limit of infinite training samples. In this dissertation, a new method for supervised learning is proposed that (amongst other useful qualities) is also 'indefinitely approximating to the truth in the long run.'

For the purposes of an engineering investigation, let us consider the question of the validity of induction closed. However, it should be noted that eminent philosophers are found arguing on both sides of the question. Further interesting discussions can be found in [90], [119], [105], and [124].

1.0.2 The supervised learning problem and notation

In this section the mathematical formulation of the supervised learning problem is presented. Other notation and definitions needed throughout this work are also clarified.

Let the term 'feature vector' denote a real valued random vector $X \in \mathcal{R}^d$, and the term 'observation' denote a real valued random variable $Y \in \mathcal{R}$. The term 'observation' may also be used in a classification context, in which case the observations are class labels: $Y \in \mathcal{G}$, where $\mathcal{G}$ is a discrete, typically finite, set of classes. The random variables X and Y have joint distribution $P_{X,Y}$.

Let n independent and identically distributed training samples be drawn from $P_{X,Y}$, and let these samples form the set of training data, with features and corresponding observation values, $\{(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), \ldots, (X_n, Y_n)\}$. A neighborhood for a feature vector X will be some subset of k training samples, that is, k features and corresponding observations, $\{(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), \ldots, (X_k, Y_k)\}$, where the indexing may be different than the indexing of the full training data set. The qth neighborhood training feature for X may be denoted $X_q(X)$. The neighborhood of X may be comprised of the k nearest neighbors or may be determined in some other manner. If the k nearest neighbors are used, then $X_q(X)$ is the qth nearest neighbor.

The central problem of supervised learning is to form an estimate of $P_{Y|X}$. An estimator for a feature vector x may be denoted $f(x)$, where the dependence on the set of training data is not made explicit. An estimate of an unknown Z may be represented by $\hat{Z}$. Common goals in supervised learning are to estimate the expected conditional observation $E[Y|X = x]$, or to estimate the value $\hat{Y}$ given X to minimize a cost. The cost of estimating $\hat{Y}$ when the true value is Y is denoted by the cost function $C(\hat{Y}, Y)$. For classification problems, a cost matrix C may specify costs where $C(i, j)$ is the cost of estimating class i when the true class is class j.

In some cases, the training samples $\{(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), \ldots, (X_k, Y_k)\}$ and the test vector X may not be drawn iid. For these cases, certain results, such as consistency, may not hold.

Other preliminary notation

Other notation used in this dissertation is reviewed here.

In general, the function $E[\cdot]$ represents the mathematical expectation of the bracketed random quantity. $E_Z[\cdot]$ more specifically represents the mathematical expectation of the bracketed quantity with respect to the distribution of the specified random variable Z.

The abbreviation 'iid' is used for 'independent and identically distributed.' The initials 'r.v.' denote 'random variable.' The abbreviation 'pmf' is used for 'probability mass function.'

The notation $z[m]$ denotes the mth component of a vector z.

The indicator function I_A is defined as $I_A(Z) = 1$ if $Z \in A$ and 0 otherwise. The indicator may also be used to enact a conditional statement. For example, $g(X)I_\theta$, where θ is a statement (such as $X^2 < a$) is defined as $g(X)$ if θ is true, and 0 otherwise. This use corresponds to Stone's notation [118].

A distortion function is a mapping $D : \mathcal{R}^d \times \mathcal{R}^d \rightarrow \mathcal{R}^+$ from the set of d dimensional real vector pairs into the set of non-negative real numbers. The distortion $D(x, \hat{x})$ is an indication of how wrong it would be to represent the vector x by $\hat{x}$. For example, total squared error is often used as a distortion function. Distortion functions must have the following intuitive property: $D(a, a) = 0$.

The definition of the support of a probability measure will also be useful: denote the probability measure for X by μ, and let $\mathcal{S}_{x,\epsilon}$ be the closed ball centered at x of radius $\epsilon > 0$. The collection of all x with $\mu(\mathcal{S}_{x,\epsilon}) > 0$ for all $\epsilon > 0$ is defined to be the support of μ, support(μ).

1.0.3 Goals of supervised learning

The primary goals of supervised learning are to create models of the correlative relationships between a feature space X and the observation space of interest Y, and to provide minimal risk estimates of the observations corresponding to feature vectors. However, a model designed to achieve a low risk on a given set of data may be overfit to that particular set of data and not generalize well to new data. Thus robustness of

the estimates is important. Other goals may depend on the application and resources, including complexity of calculations, computation time, ease of visualization of decision surfaces, smoothness of estimated surfaces, the ability to handle feature vectors with missing values, storage needed, and other engineering trade-offs and concerns.

An estimator $f(X)$ may be judged by its expected prediction error (EPE) [54],

$$EPE = E[C(f(X), Y)] = \int C(f(x), y) dP_{X,Y}(x, y).$$

For classification, if the conditional distributions $P_{Y|X)}$ are known for each class, then the Bayes' classifier is the classifier that minimizes the EPE by estimating the class

$$f_{Bayes}(x) = \arg\min_{\hat{y}} E_{Y|X=x}[C(\hat{y}, Y)].$$

1.1 Supervised learning algorithms

To apply a supervised learning algorithm, a training sample data set must be created. Determining the most appropriate or efficient features for an application is a challenging problem. Is the humidity of the factory an important variable in determining the probability of failure of a new part? How does the importance of low frequency fourier coefficients compare to the importance of high frequency coefficients in determining speech quality? Selecting the right data for features and designing relative scalings of these features is a difficult and important part of supervised learning, but not the focus of this work. In this work, it is assumed that these decisions have already been made and the concern is estimating the distribution of observations $P_{Y|X}$ or the expected observation $E[Y|X]$ for a feature vector X based on a given set of n identically distributed and independent training features and corresponding training observations, $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$.

Supervised learning algorithms can be loosely divided into two camps: parametric and non-parametric. For parametric algorithms there is an assumed model for the data source(s) and the parameters for that assumed model are estimated from the training data. Least squares hyperplane-fits and spline-fitting are examples of

parametric estimation algorithms. Parametric classification algorithms model the conditional class densities. For instance, a popular family of parametric techniques is to model each class density as a Gaussian.

Nonparametric algorithms do not assume a structure for the data. Neighborhood nonparametric algorithms, such as weighted nearest neighbor algorithms and kernels, will be discussed in more detail later. Other nonparametric algorithms, including decision trees or single-layer neural networks, constrain decision surfaces and may fit the decision surface by minimizing the empirical or estimated risk on a training set [54].

In the limit of increasingly large sets of training data, many classification algorithms will theoretically provide the same average error. Simulations with large data sets show that many algorithms do perform similarly in practice [55]. However, huge amounts of training data are a luxury. Depending on the application, the cost of training data may differ greatly, as well as the cost or even possibility of training data for all the classes under consideration. Some learning applications attempt to train a computer to mimic a human, including optical character recognition (OCR), speech recognition, speaker recognition, etc. For these applications, the cost of training data may be limited by the price of human labor. The Internet has led to movements such as Open Mind [4] which solicit free training data. Some learning applications imitate or augment human expertise, such as identifying cancer tumors from an X-ray. It may be prohibitively costly or difficult to collect a large training set of all the classes desired, such as 'no tumor' and 'tumor.' For many applications, there is simply too little training data to expect asymptotic error rates. Supervised learning algorithms that show advantages for finite data sets will always be able to find work.

1.2 Holes in the algorithmic blanket

There is a proliferation of algorithms for supervised learning, however, many of these algorithms are variations on one of a few approaches. Despite the large amount of work in this field, there are still some gaps in our ability to learn from data. Two persisting difficulties in learning are the curse of dimensionality [54], and bias from the

distribution of the training data. Ideally, an algorithm would always perform better given more (information carrying) features, and would always perform better given more training samples. However, this is simply not the case for many algorithms. In the next sections, a closer look is taken at how more features may hurt estimation, and how bias from the distribution of data samples can overcome the utility of having more training samples. The focus of this thesis is a new nonparametric method, termed LIME, that was developed to address these issues.

1.2.1 The curse of dimensionality

Practical descriptions of causal relationships between features and observations can lead to expansive feature spaces. For instance, classification of an image may be based on the values of thousands of pixels or wavelet coefficients. Consider the following short thought experiment about what Richard Bellman termed the 'curse of dimensionality.'

Imagine a line segment with endpoints at 0 and 1. Imagine ten points randomly drawn uniformly over the line segment and marked on the line. Clearly two points of the ten are on the boundary of the set of points. Each point has a nearest neighbor in a fairly predictable location - either to the right or to the left.

Next, imagine a three dimensional unit cube. Imagine ten points randomly drawn uniformly over the cube and marked in the cube. How many points are now on the boundary of the set? How easy is it to predict where a point's nearest neighbor will fall?

Last, imagine a twenty dimensional unit cube. Imagine ten points randomly drawn uniformly over the cube. How many points are on the boundary of the set? How easy is it to predict where a point's nearest neighbor will fall?

This simple thought experiment illustrates some of the issues that arise as the dimension of the feature space increases. For one, intuition built in one or two dimensions does not lead to a clear conception of twenty-dimensional space. Algorithms based on one-dimensional or two-dimensional intuition may not scale well to twenty-dimensions. Secondly, an increasing number of randomly drawn points (from just

about any distribution) will be at or near the boundary of the set. Learning methods which fit functions to data (such as least squares plane-fitting) tend to do worst at boundaries, resulting in biased estimates. Furthermore, learning methods that fit models or functions to data must accurately fit a growing numbers of parameters as the dimension increases. Even when the model is well suited to the data, there may be too many parameters to estimate sensibly. For instance, 1325 parameters must be estimated for a Gaussian in 50 dimensions (additional structure is often assumed in order to bring down the number of parameters needed). On the other hand, neighborhood algorithms that compute a weight for each training sample in a local neighborhood do not face this problem of increasing parameters with increasing dimension.

Neighborhood methods, such as k-NN and kernel algorithms, base weightings on the distance from test to each training point. Points scattered in high-dimensional spaces tend to be all far away from each other by a Euclidean distance. Thus those method's emphasis on pure distance becomes less useful.

High-dimensional feature spaces require more data to populate them densely than lower dimensional spaces. Thus a 'local' neighborhood of the k nearest neighbors may stretch far in any one dimension; see [54] for examples. Further, the variance in feature space of the closest neighbor tends to increase sharply with dimension.

Decision trees, including algorithms such as CART^{TM} [14] and MART [54], provide some relief from the curse of dimensionality. Most decision trees model the observation space as a set of rectangular compartments. An important advantage to decision trees is that they can avoid the confusion of multi-dimensional data by focusing on one or a few dimensions at a time. However, this advantage is also a disadvantage - linear splits along feature dimensions may be too constrained a model to represent accurately the decision boundaries in a multi-dimensional feature space.

Much of the intuition used to design supervised learning algorithms, and in particular parametric algorithms and neighborhood methods, is based on intuition built from one or two dimensions and generalized. The LIME algorithm presented in this work was originally developed for three dimensional color management problems [44]

and it will be shown through simulations that it is more suited than standard neighborhood methods for three and higher dimensions.

1.2.2 Local bias

One of the opportunities in nonparametric classification and regression is to reduce sensitivity to bias. Goin [40] discusses nonparametric estimation bias that is due to different training and test probability distributions, or due to the population of one class of training samples being larger than the population of another class of training samples.

Even when the training distribution and test distribution are iid, the distribution of feature values of the training data can still cause a local bias of learning estimates. One would like to estimate $P_{Y|X}$, which can be decomposed as,

$$P_{Y|X} = \frac{P_{X|Y} P_Y}{P_X}. \tag{1.1}$$

K-NN techniques estimate $P_{Y|X}$ by estimating the class conditional distribution $P_{X|Y}$. The estimated k-NN probability of class 1 at a point x depends on the local class conditional distributions,

$$\hat{P}(y = 1|x) \sim \frac{\int_V P(x|y = 1)}{\sum_g \int_V P(x|y = g)}, \tag{1.2}$$

where V is a local volume centered at x, and is dependent on the local distribution of training samples, P_X.

A local bias arises if the local mean of the class conditional distribution deviates from the distribution at the point x. In accordance with Friedman's definition of bias [35], let the local bias be defined,

$$bias\ \hat{P}(y = 1|x) = P(y = 1|x) - E[\hat{P}(y = 1|x)] \tag{1.3}$$

where the expectation is taken over all possible sets of training samples. For k-NN, the estimate $\hat{P}(X = x|Y = 1)$ depends on the local sum of training samples of class 1. A local bias may arise if there is a strong local gradient of the class conditional density.

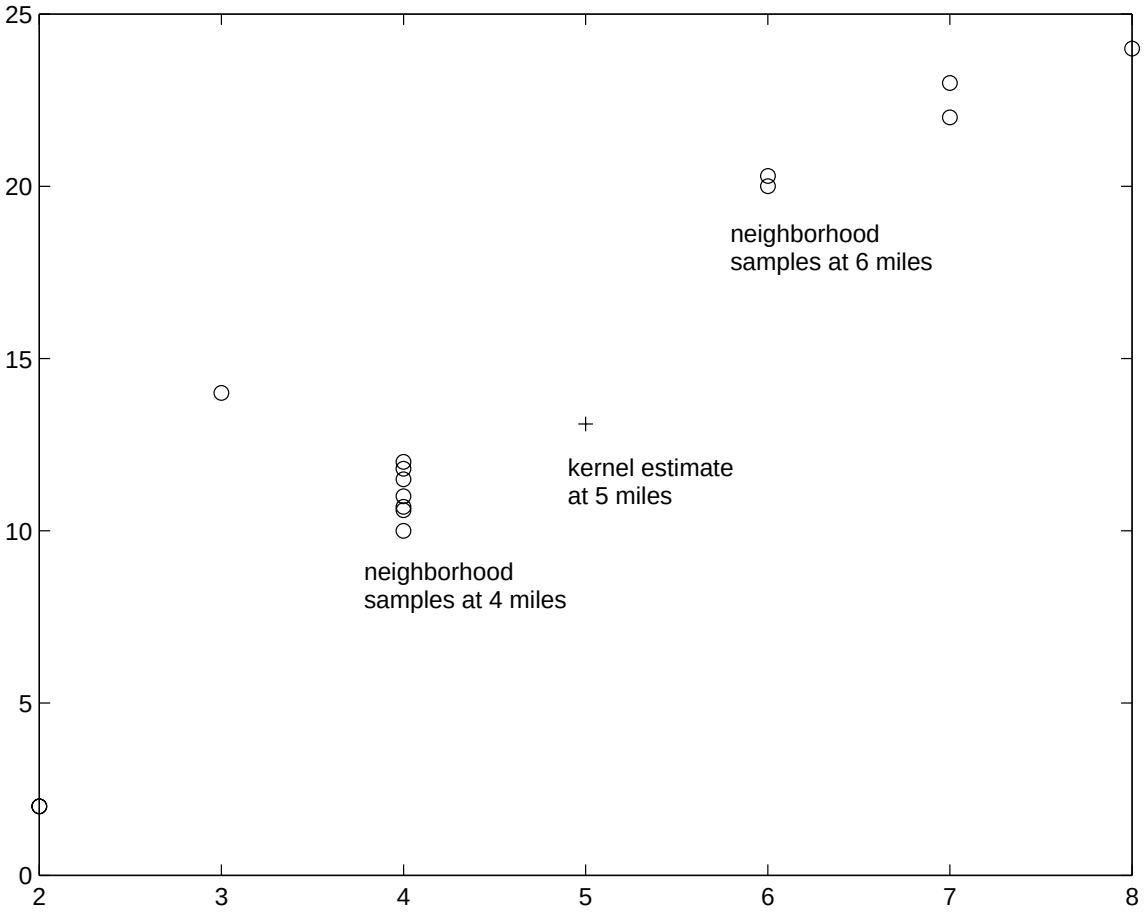

Figure 1.1: Example of training data bias affecting an estimate

Consider an example, as shown in Figure 1.1. In the example, the feature $X \in \mathcal{R}$ represents the number of miles north of a landmark that a measurement is taken. The observed variable $Y \in \mathcal{R}$ measures the strength of a cellular phone signal. The training samples are marked with 'o's in the Figure. The problem is to estimate the signal strength at $x = 5$ miles. Consider a neighborhood method with a symmetric neighborhood of 1 mile on either side of the test point. Since the training samples are equidistant from the test point, most neighborhood methods will give them the same weight. However, that leads to the biased estimate marked with an '+' at $x = 5$.

As a classification example of the effect of local bias, consider a two class problem

with a one-dimensional feature space, shown in Figure 1.2. The class conditional pdf's $P(x|y = 1)$ and $P(x|y = 2)$ have a Gaussian distribution, with mean 0 and $\sigma_1 = 1$ for class 1 and $\sigma_2 = 4$ for class 2. The two classes are equally likely. In Figure 1.2 the two conditional distributions are shown, as well as 60 samples drawn randomly from $P_{X,Y}$. The Bayes' decision region is bounded by where the two distributions are equal, and a classification algorithm will do best on average if it classifies points within the boundary as class 1, and points outside the boundary as class 2. Imagine a k-NN algorithm with some k. For points just outside the Bayes' decision boundary, a majority of the k nearest neighbors will be from inside the Bayes' decision region. This is because the local probability density is higher inside the decision boundary than just outside the boundary. Thus the probability distribution P_X biases the estimate.

The bias effect grows as more identical, independent Gaussian feature dimensions are added. Let the conditional probability $P(x|y = 1)$ have a two-dimensional Gaussian distribution with mean 0 and diagonal covariance; the standard deviation is one in each dimension. Similarly, the conditional probability $P(x|y = 2)$ has a two-dimensional Gaussian distribution with mean 0 and diagonal covariance; the standard deviation is four in each dimension. A two-dimensional example of 100 samples drawn randomly from $P_{X,Y}$ is shown in Figure 1.3. The Bayes' decision region is outlined with a circle on the figure. As in the one-dimensional case, test points that fall slightly outside the circle are more likely to have near neighbors within the circle. Since samples in the circle are more likely to come from class 1, test points outside the circle are more likely to be classified by k-NN as class 1. Thus, the distribution of the sample data P_X is biasing the classification.

Any gradient in the distribution of the training samples can lead to a bias in the estimate. Neighborhood methods such as k-NN, weighted k-NN, and kernels rely only on distance from a test point to a neighborhood sample to determine the

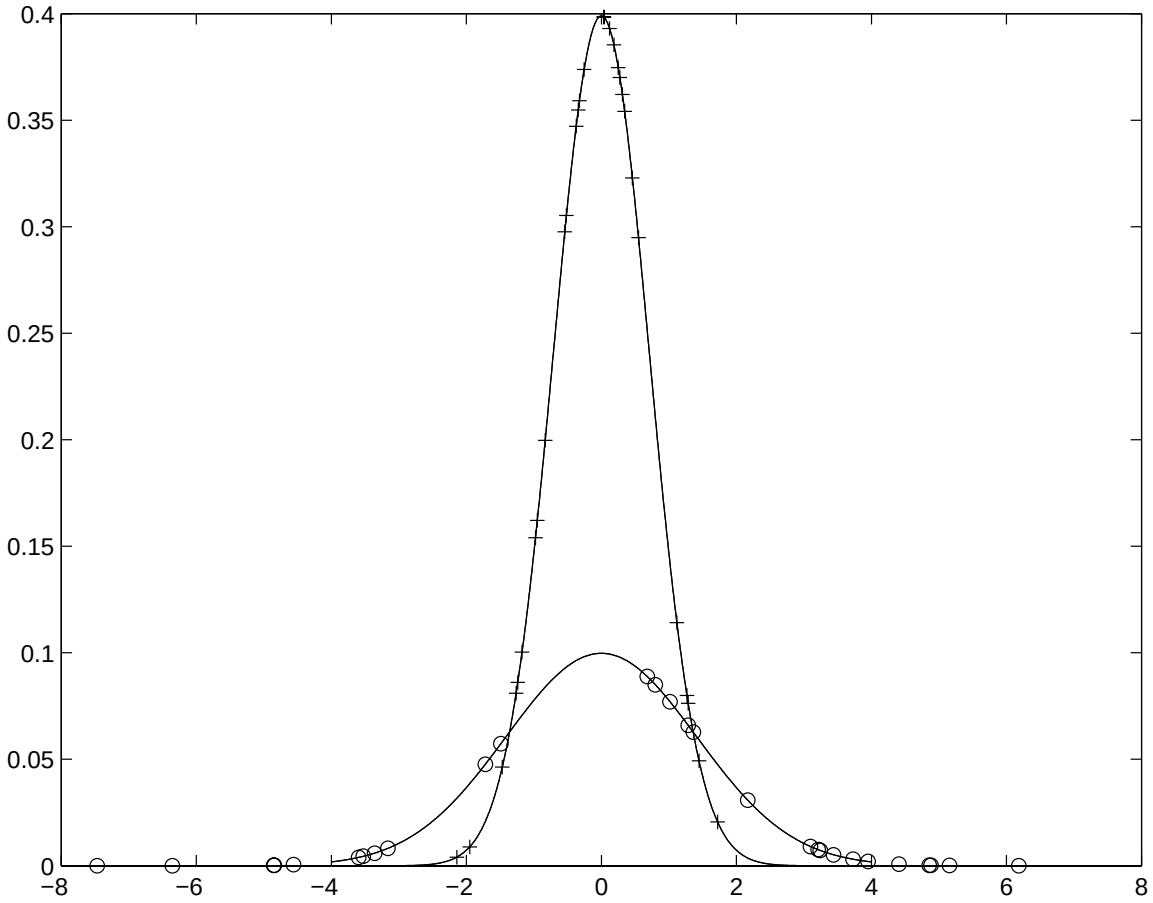

Figure 1.2: Example of how P_X can bias classification

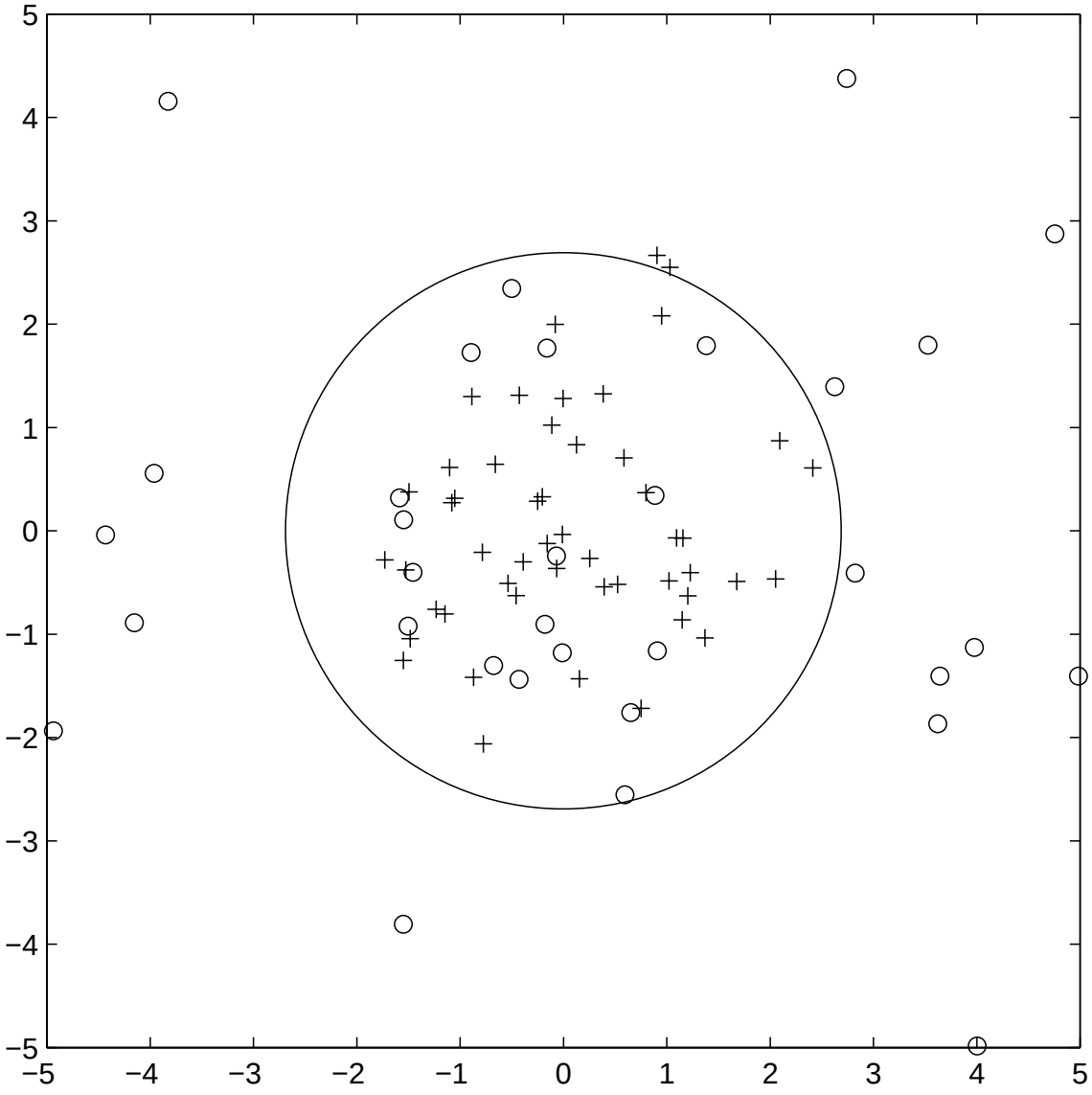

Figure 1.3: Example of how P_X can bias classification

weighting on a sample point. In Section 1.3.1, these methods are discussed in further detail. Other learning techniques fit parameters to minimize the error on training sets, creating estimates that may also be biased by the training sample distribution. For example, consider a least-squares plane-fitting. The plane is fit to minimize total squared error. If there are many more samples in one part of the feature space than in others, the plane may be preferentially fit to that part of the feature space, possibly with significant biases in other parts of the feature space.

Friedman's t compensation

The effect of local bias on classification error is explored in detail in papers by Fukunaga and Hummels [36], and Friedman.

Friedman [35] considers the two-class problem and explores adjusting the estimate $\hat{P}(y = 1|x)$ by a global parameter $t \in [-1, 1]$:

$$\tilde{P}(y = 1|x) = \hat{P}(y = 1|x) + t \tag{1.4}$$

Friedman's adjusted estimate $\tilde{P}(y = 1|x)$ is not guaranteed to fall between 0 and 1. Ideally, the variable t will compensate for the bias $P(y = 1|x) - \hat{P}(y = 1|x)$. Given a known distribution $P_{X,Y}$, Friedman gives examples using k-NN and shows how the variable t could be determined. He notes that in practice, the true distribution $P_{X,Y}$ is rarely known, but that t could be trained by cross-validation. Friedman leaves as future work applying the t compensation to problems where the true distribution is not considered known.

To judge the validity of Friedman's t compensation, a few simple simulated experiments were run. The value of t trained by cross-validation approached the optimal value. For a simulation with a consistent local bias (i.e. the local bias is of the same sign everywhere), the t compensation worked quite effectively.

The t compensation however adjusts for local bias with the same adjustment (t) everywhere. In practice, the local bias is likely to differ over the feature space. The t compensation could be a viable machine learning technique if the compensation was more local. Another limitation is that Friedman's work only considers the two-class problem, but a simple solution to that might be to compensate each class g's

conditional density by t_g, and require $\sum_g t_g = 1$. If the t compensation was a practical technique, it could be coupled with any distribution estimation technique (including the LIME algorithm proposed in this work).

Reducing the bias of estimation

Bias is often a local phenomena, and as such requires a localized solution. Using the contextual information when assigning weights in nonparametric estimation can help. The worth of a training sample depends on what other training samples are in a test point's neighborhood. Training samples with similar feature vectors can reduce the noise of estimates, but as seen in the above example, they can also cause a serious bias. In the next few sections, neighborhood methods and linear interpolation are reviewed and it is shown how they may be combined through the principle of maximum entropy to yield an algorithm that takes into account the spatial relationships of the training points and thereby reduces the effects of local bias.

1.3 K-NN and linear interpolation

Let us take a closer look at nearest neighbor and kernel methods and at a seemingly completely unrelated algorithm, linear interpolation. The limitations of both approaches are discussed and the insight gained is used to develop a new approach, termed LIME. The LIME algorithm is presented and explored through simulations in the next chapter.

1.3.1 Nearest neighbor and kernel methods

Perhaps the most famous non-parametric method is the k-NN rule, for which Fix and Hodges [34] provided the first consistency results. The method is simple but performs competitively on many real-world problems [54], [56].

The 1-NN estimate is equal to the class label (or value, for regression problems) of the closest training sample to a test point x,

$$f_{1-NN}(x) = y_1(x).$$

where, for any integer j, $x_j(X)$ is the jth closest neighbor to a test point x out of the set of training samples under a chosen metric, and $y_j(X)$ is the training observation corresponding to $x_j(X)$. To simplify notation, from this point on the notated dependence on x will be omitted unless useful for clarity.

More generally, the k-NN rule weights the k nearest neighbors equally, $w_j(x) = \frac{1}{k}$, for $j = 1$ to k. Then the class or value is chosen that minimizes the expected cost. Thus, the k-NN rule results in the estimate

$$f_{k-NN}(x) = \arg\min_{\hat{y}} \frac{1}{k} \sum_{j=1}^{k} C(\hat{y}, Y_j).$$

Variations on the k-NN rule abound. Weighted nearest-neighbor methods adaptively tailor the weights on training points, usually defining a monotonically decreasing kernel that weights neighbors less as a function of their distance to a test point.

Similarly, kernel estimates for regression were proposed (independently) in 1964 by Nadaraya and Watson [50]; kernel estimates had been proposed earlier for density estimation by Parzen [97], and related ideas can be found in papers by Grenander from the 1950's. Define a kernel function $K : \mathcal{R}^d \rightarrow \mathcal{R}$, where K is usually even, nonnegative, and monotonically decreasing along rays starting at the origin. The kernel K usually has only one parameter h (which may be a vector) which specifies the bandwidth. The bandwidth parameter h may be data dependent. As an example, the Epanechnikov kernel is defined as [54]

$$K_{epan}(x - x_j) = .75 \left(1 - \frac{\|x - x_j\|}{h}\right)^2$$

for $\frac{\|x-x_j\|}{h} \leq 1$, and $K_{epan}(x - x_j) = 0$ otherwise.

Another popular kernel is the tricube kernel [54], defined as

$$K_{tricube}(x - x_j) = \left(1 - \|x - x_j\|^3\right)^3$$

for $\|x - x_j\| \leq 1$ and $K_{tricube}(x - x_j) = 0$ otherwise.

For an estimation for feature value $x \in \mathcal{R}^d$, the kernel weight corresponding to a training point x_j is

$$w_j(x) = \frac{K(x - x_j)}{\sum_{j=1}^{n} K(x - x_j)}.$$

The kernel estimate is

$$f_K(x) = \arg\min_{\hat{y}} \sum_{j=1}^{n} w_j(x) C(\hat{y}, y_j)$$

Various kernels have been proposed and investigated, including kernels with negative sidelobes. Many variations and properties are reviewed in [30], [54], and [89].

As shown in the example in the last section, kernel and nearest-neighbor methods can be biased by the distribution of the training samples. The bias of kernels whose bandwidth overlaps the boundary of the training data can be of a larger order of magnitude than the bias in the interior [107]. To address this issue modification schemes for kernels have been proposed [107], [50]. The main cause of the estimation bias is asymmetry of training data near the boundary. If feature dimension size increases but the number of training points stays constant, the surface area of the boundary increases exponentially.

Depending on the distribution of the training data, similar bias problems can occur in the interior of the feature space as well. Variations of $k - NN$ and kernel algorithms allocate weight to neighborhood training points based on their distance to a test point [30], [54], [89]. These training points may be original samples or points that represent samples, as is the case with edited k-NN or clustered k-NN (see Section 2.12). This is not a full use of the information at hand, because one also knows where the training points lie in relation to each other. The information about the spatial distribution of the training points is not taken into account. The distribution P_X of the data affects the estimate of $P_{Y|X}$. Examples of the local bias this can cause were given in Section 1.2.2

Clearly, a training point's worth for estimating $P(y|x)$ depends on the distribution of training points in the neighborhood. Repeated training points are useful in characterizing and reducing the effect of noise, but it will be shown that these advantages can be retained while reducing bias effects.

In Section 1.3.3, the LIME algorithm is introduced. LIME allocates weight to training points by taking into account which other training points are available. More specifically, LIME uses information about the spatial distribution of the training points in the neighborhood to create linear estimates that are not as biased by the distribution of the training points. The bias is removed to a first order by constraining the weights with the equations of linear interpolation. First, let us review linear interpolation.

1.3.2 Linear interpolation

Linear interpolation is a regression method traditionally applied to a neighborhood of $d+1$ training samples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_{d+1}, y_{d+1})\}$ in d dimensions where $x_j \in \mathcal{R}^d$ for all j [66], [110]. The $d+1$ training samples must contain the test point within the closure of the convex hull spanned by the $d+1$ training samples. Over the $d+1$ samples, weights are solved to satisfy the d-dimensional constraint $\sum_{j=1}^{d+1} w_j(x)x_j = x$, and the one-dimensional constraint $\sum_{j=1}^{d+1} w_j = 1$. To form the estimate for a test point x, the weights are applied in the output domain $f(X) = \sum_j w_j(x)y_j$.

Contrasting linear interpolation and plane-fitting

Least-squares plane-fitting models data with the hyperplane $f(x) = a^T x + b$ that satisfies

$$\arg\min_{f(x)} \sum_j (f(X_j) - Y_j)^2.$$

When fitting a plane, there is a degree of freedom to specify the slope for each dimension, plus a degree of freedom for the offset. Thus for k training points in d dimensions, there are $d+1$ parameters to fit for a hyperplane. Clearly, when $k > d$, and the points do not lie on a plane, plane fitting can not fit a plane through all the points, and thus the plane with minimum distortion (e.g. minimum mean squared error) is fit. Thus one can say that plane-fitting has $d+1$ parameters to try to fit k constraints.

Estimation Method	Number of unknowns to solve for	Number of constraints
plane-fitting	d+1	k
linear interpolation	k	d+1

Table 1.1: Contrasting plane-fitting and the linear interpolation equations for k sample points in d dimensions

On the other hand, linear interpolation selects a weight for each of k training points. And there is an equation for each dimension, so there are d constraints to satisfy. An additional $d+1$th constraint guarantees that the weights sum to one. Thus linear interpolation is in some ways the opposite of plane fitting, as summarized in Table 1.1. However, in order to solve the linear interpolation equations exactly, the number of unknowns k must equal the number of constraints $d+1$, which in fact results in a planar surface estimated between vertices that are the $k = d+1$ training points. Least-squares plane-fitting gives for $d+1$ training points in d dimensions the same estimates as linear interpolation. However, in this work linear interpolation is generalized, and then the estimates are no longer planar.

Local regression [54] is a form of nonparametric regression in which a neighborhood is chosen (as in k-NN or kernel algorithms), and then a model is fit to the observations in that neighborhood. A popular version of this is to least-squares plane-fit over a local neighborhood, termed local linear regression. This technique can result in biased estimates and radical extrapolations at the boundaries of the training data set in feature space. A number of methods exist to improve the performance of local regression methods, see [126] for further discussion.

Limitation of linear interpolation

Linear interpolation is interesting because it succeeds where neighborhood algorithms fail - it takes into account the spatial relationships of the neighborhood training samples. The system of equations $\sum_j w_j x_j = x$ works in any number of dimensions and provides a way to compensate for the empirical distribution of the training samples.

Linear interpolation suffers from one basic problem - the number of unknowns k must equal the number of constraints $d+1$ in order to solve the linear interpolation system of equations exactly and arrive at the weightings of the k training samples. For example, solving the linear interpolation system of equations exactly for a one dimensional problem yields only two weights, restricting the neighborhood to two training samples. That may not sound too restrictive, but consider an application such as color management with three feature dimensions. Only four weights can be solved for, restricting all estimates to a neighborhood of only four training samples in three dimensions. It is likely that one will have more relevant data to use to form an estimate. In the following section, linear interpolation is extended to use as many of the data points as one thinks are relevant. This leads to the LIME estimation method which decreases training sample distribution bias and performs well with increasing feature dimensions.

1.3.3 Introduction to the LIME algorithm

Linear interpolation takes into account the spatial distribution of training samples, but is limited to only $d+1$ training samples for a d dimensional feature space. Often there are more than $d+1$ training samples considered relevant to estimate a test point x. Consider the effect of including more data into the $d+1$ linear interpolation equations:

$$\sum_{j=1}^{k} w_j x_j = x \tag{1.5}$$

$$\sum_{j} w_j = 1 \tag{1.6}$$

For $k > d+1$, there are more unknowns than constraints, and the system of linear interpolation equations is an underdetermined system with an infinite number of solutions. Having more data than equations, $k > d+1$ is not a problem, but rather an opportunity to add other criterion to result in only one solution. One good criterion would be to use the neighborhood training points as equally as possible

given the spatial constraints of the linear interpolation equations. The intuition is that without a reason to trust one neighborhood point more than another, the points should be weighted as equally as possible.

Different mathematical criteria could formalize this approach. One criterion is to maximize the entropy of the weights. Jaynes' principle of maximum entropy [60] says to solve for probability distributions by choosing the distribution with maximum Shannon entropy $H(w) = -\sum w_j \log(w_j)$.

To find an unique solution to an underdetermined system of linear interpolation equations, one can maximize the entropy of the weights. Maximizing the entropy chooses the weight solution that commits the least to any one training sample. This will ensure that sample points are used as diversely as possible. Maximizing entropy is a popular way to solve under-constrained systems, e.g., [93], [9], [45], [38], [68], [15], [58], [27], [25], [112]. Under certain assumptions it is the maximum likelihood solution [79], [127]. Conveniently, maximizing entropy often leads to closed form solutions. In Chapter 6, the maximum entropy principle is discussed in more depth.

A more general approach to solving linear interpolation equations is to minimize the relative entropy to a model instead of maximizing the entropy. Consider the case that one would like to put more weight on specific sample points. For instance, one might want to weight training points closer to the test point more heavily. Given a kernel or model of the weights (monotonically decreasing with distance or otherwise) one can solve the underdetermined system of linear interpolation equations by solving for the weight solution that minimizes the relative entropy to the model (or kernel). From this perspective, maximizing the entropy is simply assuming an equal weighting (k-NN) kernel over the neighborhood. Results are not presented for minimizing relative entropy to a model(kernel), but it is proposed as an extension in Chapter 7.

Unfortunately, it may not be possible to solve the system of linear interpolation equations exactly. The linear interpolation equations are infeasible if the test point x lies outside the convex hull of its neighborhood points. Let $D(w)$ be an alternate notation for the distortion function $D(\sum_{j=1}^{k} w_j x_j, x)$. For example, $D(w)$ could be the mean squared error between $\sum_{j=1}^{k} w_j x_j$ and x. Requiring weights to be chosen to

exactly solve the linear interpolation equations is equivalent to requiring $D(w) = 0$, and this may not be a feasible problem. A more general goal is to minimize $D(w)$ and maximize the entropy of the weights, $H(w)$. In the next chapter, the LIME algorithm is presented, which trades-off the reproduction distortion and the entropy of the solution.

Chapter 2

Linear interpolation with maximum entropy

Mitte, wie du aus allen
dich ziehst, auch noch aus Fliegenden dich
wiedergewinnst, Mitte, du Stärkste.

Center, how from all beings
you pull yourself, even from those that fly,
winning yourself back, irresistible center.

Rainer Rilke

The LIME algorithm requires that for each test point X there be chosen a set of relevant training points, termed the 'neighborhood.' A common choice for the neighborhood of a test point X is to use the k nearest training samples to X, but the choice of neighborhood rule is left to the discretion of the practitioner. Given a test point and its neighborhood, the LIME algorithm computes a weight for each neighborhood training point so that the weights form a probability mass function. The weights are chosen to both minimize the distortion between the weighted combination of the local training points and the point to be estimated, and to maximize the entropy of the weight distribution. The parameter λ controls the trade-off between the two objectives, $\min D(w) - \lambda H(w)$. The algorithm is presented in the next section.

Some key ideas and issues are explored in the rest of this chapter using simulations. The impact of neighborhood choice on LIME is considered. One simulation shows that LIME can exploit a large neighborhood of training sample points effectively even if the distribution of the training samples is asymmetric. It is seen that LIME's use of direction as well as distance becomes increasingly important with increasing dimension size. The effect of the parameter λ, which controls the trade-off between minimizing distortion and maximizing entropy, is illustrated. The LIME weights are shown to be exponential and examples of the resulting data-adaptive kernel are presented.

2.1 Linear interpolation with maximum entropy

Let (X, Y) be a pair of random variables such that $X \in \mathcal{R}^m$ and $Y \in \mathcal{R}$ or $Y \in \mathcal{G}$, a finite set of classes. Given independent, identically distributed training samples $\{(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), \ldots, (X_n, Y_n)\}$ drawn from a distribution $P_{X,Y}$, estimate the probability distribution $P_{Y|X}$, or a minimum cost estimate of Y given a value of X, $\hat{Y} = \min E[C_{(Y,X)}]$.

Step 1) For a test point X, let a neighborhood of training samples be chosen and let $j = 1, 2, \ldots, k$ index the neighborhood set.

Step 2) For each sample pair from the set of k neighborhood training samples, $\{(X_1(X), Y_1(X)), (X_2(X), Y_2(X)), \ldots, (X_k(X), Y_k(X))\}$, calculate the corresponding weight $w_j(X)$, such that $\sum w_j(X) = 1$ and $w_j(X) \geq 0$ for all $j = 1$ to k, and such that the weights solve

$$\arg\min_{w} \left[D\left(\sum_{j=1}^{k} w_j(X) X_j(X), X \right) - \lambda H(w) \right]$$

where $\lambda \geq 0$ is a chosen parameter, $D(r, s)$ is a distortion function, and $H(w)$ is the Shannon entropy: $H(w) = -\sum_{j=1}^{k} w_j \log w_j$.

Step 3) For classification applications, the LIME estimate of the probability that Y is a member of class $g \in \mathcal{G}$ is

$$\hat{P}_{Y|X}(g|x) = \sum_{j=1}^{k} w_j(x) I_{Y_j(X)=g}.$$

The expected minimum cost LIME estimate of Y is

$$f_{LIME}(x) = \arg\min_{\hat{y}} \sum_{j=1}^{k} w_j(X) C(\hat{y}, Y_j(X)).$$

Unless otherwise noted, the distortion $D(r, s)$ used in Step 2 is assumed to be the mean squared error, $D(r, s) = \frac{1}{m} \sum_{d=1}^{m} (r_d - s_d)^2$. However any desired distortion could be used with LIME. A number of interesting distortion functions have been developed for or used with k-NN, for instance in [116]. Distortion functions that have proved successful for compression may also be useful.

By the Weierstrass theorem (see Appendix), a minimizer w^* for Step 2 exists as long as the distortion function is continuous in w. A sufficient condition for the minimizer w^* to be unique is that the distortion function be convex with respect to w (for example, any l_p distance function or monotonic function thereof).

2.2 LIME assumptions

The algorithm approximates a test point X by a linear combination of training points $\{X_1, X_2, \ldots, X_j\}$ and uses the same linear weighting on the training observations $\{Y_1, Y_2, \ldots, Y_k\}$ to estimate the corresponding estimated observation $\hat{Y}$. This limits the estimated observation to be within the convex hull of the known neighborhood observations. If a test point X lies beyond the boundaries of the convex hull of the training features, the algorithm projects X onto the convex hull of the training features (by minimizing the distortion) and then approximates the projected test point. Thus test points beyond the boundaries yield estimates constrained to the range of the neighborhood training data; a conservative estimate, but one that avoids dot-com dangers of extrapolation. Surface area grows rapidly with dimension, and for high-dimensional problems many test points occur beyond the boundary of the training data. For example, on a standard test set for vowel recognition [54], over

80% of the test data fall outside of the convex hull formed by the entire set of training points.

Every estimation algorithm has to make some assumptions in order to estimate the unknown. Ideally, the information known about a particular dataset is used to determine which estimation algorithm's assumptions are most correct for that data set. For instance, if it is known to be a two class problem where the data represent the output of two independent Gaussian sources with equal covariance matrices, then modelling the classes as Gaussians and using linear discriminant analysis may be the best choice. However, if little information is known about the problem, then a non-parametric method such as LIME may provide robust estimates.

2.3 LIME is a vector algorithm

In physics, equations usually feature velocity, not speed. The difference is that velocity is a vector, and speed is merely a magnitude. Methods like k-NN or kernels only take advantage of the distance that a training point is from a test point, not the direction. By using the equations of linear interpolation, LIME exploits the vector nature of the data, taking into account direction as well as distance. This use of the joint vector information about the training samples, by minimizing the distortion between the weighted neighborhood training points and the test point itself $D(\sum_j w_j X_j, X)$, is the most important way in which LIME differs from k-NN. Nearest-neighbor methods weight each training point independently of the other training samples available. LIME weights each training point using the context of the other training data.

There are a large number of variations of kernels and weightings for neighborhood methods [30], [54], [108], [50]. In the following sections, simulations are presented comparing LIME to the basic k-NN algorithm and to the tricube kernel; both algorithms were detailed earlier in Section 1.3.1. The simple k-NN technique achieves competitive performance over a wide range of classification problems, even when compared to state-of-the-art classifiers [54], [56]. The tricube kernel is a popular kernel [54] and is representative of the general class of positive symmetric smoothing kernels. The simulations show that the difference in performance between k-NN and the tricube

kernel is generally small compared to the difference between these symmetric kernels and LIME. This implies that although one cannot compare to every symmetric kernel, the simulation results should generalize well to other symmetric kernels. It should be noted that in some of the simulations (as noted in the simulation description) the tricube kernel is applied over a neighborhood where the neighborhood consists of k nearest neighbors instead of a bandwidth. This makes it easier to compare the neighborhood techniques.

2.4 Kohonen's example

To see the differences between the LIME algorithm and other neighborhood methods, an example is used from Kohonen et al. [77]. The same or similar simulations have been used by other researchers, including [42] and [54]. The Kohonen simulation is used to show a variety of effects. Separate simulations show the effects of increasing the amount of training data, the effect of increasing the number of iid training dimensions, the effect of varying λ, and the effect of varying the size of the neighborhood. Comparisons are made with other neighborhood methods and the Bayes' classifier (which 'knows' the class conditional densities and thus achieves on average the Bayes' error).

In the Kohonen simulation, there are random feature vectors $X \sim f(x) = .5f^0(x) + .5f^1(x)$ where $f^g(x)$ is the conditional pdf given that the class label $Y = g$ and Y is equally likely to be 0 or 1. The misclassification costs C_{01} and C_{10} are taken to be equal. For Kohonen's simulation, the two class conditional densities are Gaussian random vectors with iid components, zero mean, and equal variance. The variances for each independent feature component are $\sigma_0^2 = 1$ for class 0 and $\sigma_1^2 = 4$ for class 1.

Figure 2.4 shows the results of Kohonen's simulation with 200 training data samples and two feature dimensions. Class 0 is represented by zeros and class 1 by crosses. The thick circle with radius 1.923 in the image is the Bayes' decision surface. Samples inside the circle are more likely to have been drawn from class 0, and samples outside the circle are more likely to have been drawn from class 1.

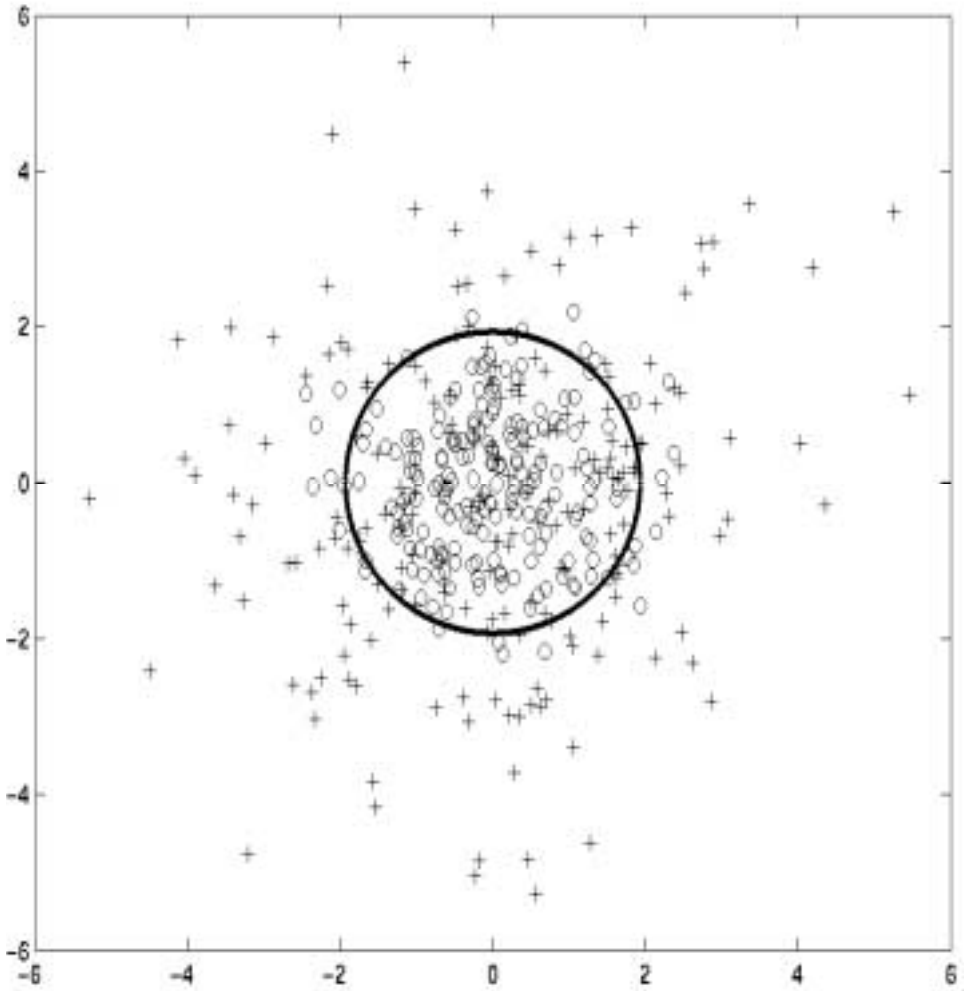

Figure 2.1: Plot of 200 training samples for Kohonen simulation with two dimensions

2.5 The importance of a good neighborhood

It is common wisdom in real estate that one should buy property in the nicest neighborhood one can afford. In statistical learning the relevance of a 'nice' neighborhood should also not be underestimated. Nonparametric algorithms often depend on defining a relevant neighborhood around a test point.

There are many approaches to defining a neighborhood. One of the oldest is the approach of k-NN, to pick the closest k neighbors, with k a parameter perhaps learned on the training set. Kernel algorithms define the relevant neighborhood implicity by the width and fall-off of the kernel.

Figure 2.2 shows simulation results for a twenty dimensional feature space corresponding to Kohonen's simulation. Here the neighborhood is defined to be the test point's k nearest neighbors, where the total number of training samples $n = 200$. The error rate is graphed as the neighborhood size k is increased. The λ parameter for LIME is held constant at 1. These results show that increasing the neighborhood size does not decrease the error for k-NN or the tricube kernel. However, for a large range of neighborhood sizes LIME is able to use the additional sample points effectively and the error rate is decreased. The difference arises in how the local bias of the distribution of training data is handled. Near the Bayes' decision boundary, more neighbors are likely to come from class 0 due to the local slope of the Gaussian sources P_X. Go back to Section 1.2.2 to understand more about how local bias can affect estimations. In particular, Figure 1.3 is a two-dimensional visual example of the bias effect that renders extra neighbors useless in the k-NN and tricube results shown in Figure 2.2.

Research into defining the right neighborhood for non-parametric learning algorithms may or may not be directly applicable to LIME. For instance, the work by Rice in bandwidth choice for kernels [107] chooses the bandwidth to minimize an unbiased estimate of the risk that only applies to kernel methods. However, recent work in this area has focused on using the sample observations to specify non-trivial neighborhoods [52], which should, at least in principle, be useful for defining good neighborhoods for LIME as well.

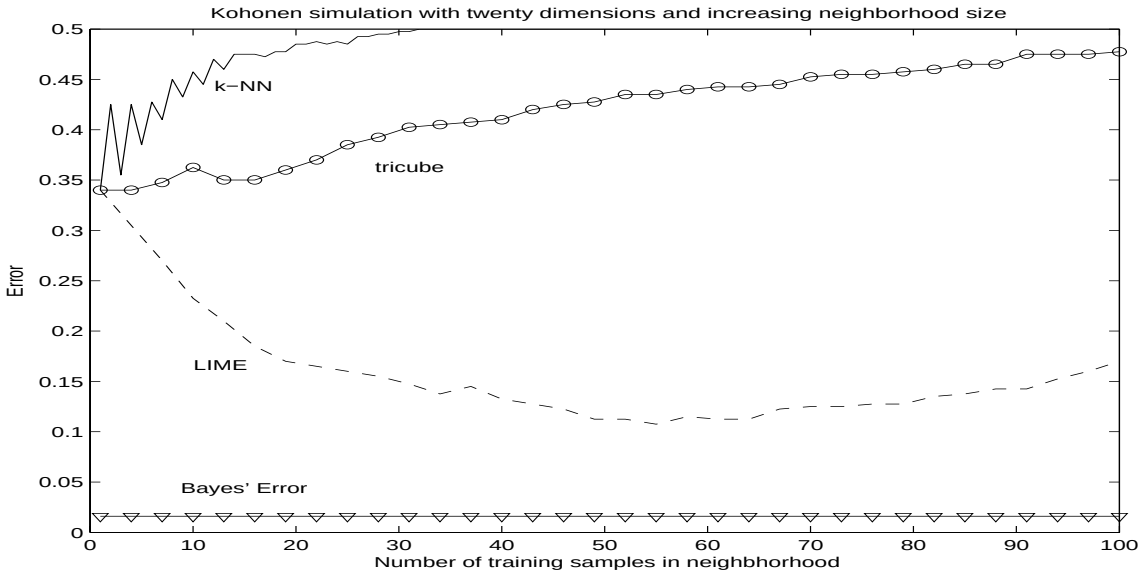

Figure 2.2: Kohonen simulation with increasing neighborhood size, twenty feature dimensions, and 200 training samples

For some applications, a good neighborhood might be one that adapts in size depending on how sparse the data are in an area. One definition of such a neighborhood would be all training points falling within a radius $(1+\alpha)$ times the distance from the point to be estimated to its nearest neighbor. This definition has the benefit of always including at least one neighbor. A value of $\alpha = .26$ was shown to work well for at least one experimental application [43]. Ideally, α would be trained on a training set of data, or trained by cross-validation, or selected based on some information about the sparsity of the data.

Another metric for a good neighborhood is whether the test point X is within the convex hull spanned by the neighborhood points. This is a requirement for traditional linear interpolation regression estimates. Being within a convex hull of data points can help decrease estimation bias and informs the estimation about how the observation space is changing in all directions.

2.6 LIME bridges k-NN and linear interpolation

The LIME algorithm includes a parameter λ to trade-off between maximizing the entropy and minimizing the distortion. Setting λ to a very high constant maximizes entropy at whatever cost in distortion, and at this extreme the algorithm disregards the mean constraint that is enforced by the distortion criteria. Maximizing entropy of a distribution with no other constraints leads to a uniform distribution for the weights, and thus LIME with a very high λ behaves like a k-NN estimator, where k is the number of points in the neighborhood.

On the other hand, if λ is set to a relatively low constant, then the algorithm is focused on achieving zero distortion $D(w)$ and uses the entropy part of the functional $H(w)$ to choose the most diverse of the set of non-unique solutions. These intuitive results about the limiting λ cases are proven in Section 3.1.

If the measurements of the training data are noisy, then working hard to achieve a perfect reconstruction is not so important as the accuracy of the reconstruction is limited by noise, and thus a higher λ should be used, leading to a more robust solution. This is exemplified in a simulation with additive noise comparing LIME to a least squares fitting of a hyperplane in Section 4.4.6

In Figures 2.3 and 2.4, the Kohonen simulation detailed in Section 2.4 is shown with λ varying from 2.5×10^{-9} to 10^{10}. In the simulation shown in Figure 2.3, there are two feature dimensions and, and in Figure 2.4 there are twenty feature dimensions. There were 200 training points and in an earlier simulation (where λ was held constant at 1) the error was found to be minimized with $k = 75$ neighbors for the two dimensional problem and $k = 87$ neighbors for the twenty dimensional problem.

The results of Figures 2.3 and 2.4 show how for small λ the maximizing entropy part of the functional is secondary to minimizing the distortion; the result is virtually zero distortion and a plateau in the classification accuracy. At the other extreme

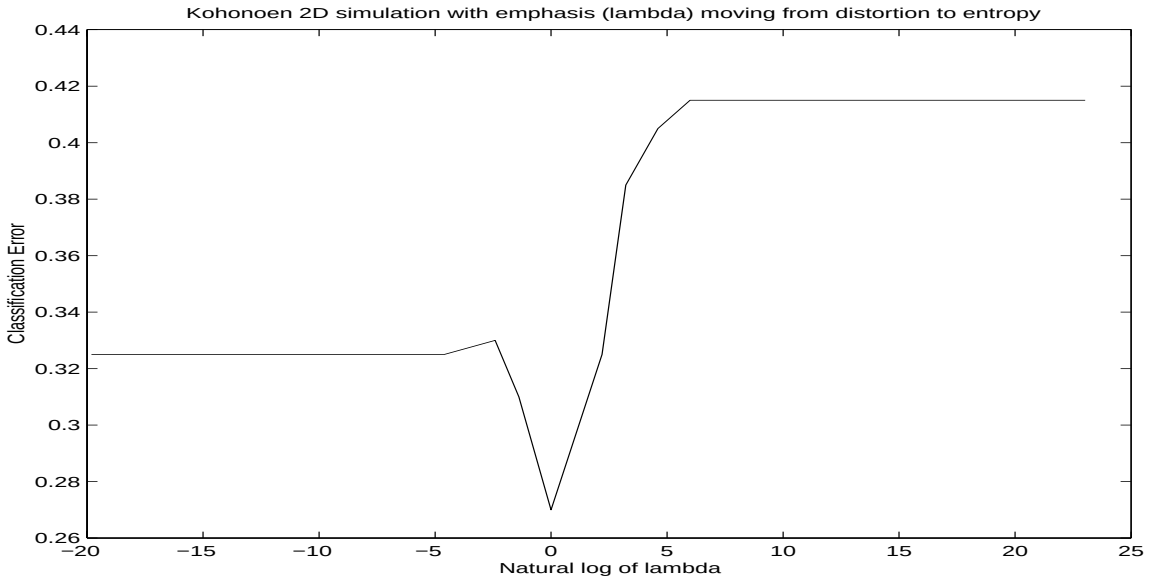

Figure 2.3: Kohonen simulation varying lambda with two feature dimensions and 200 training points

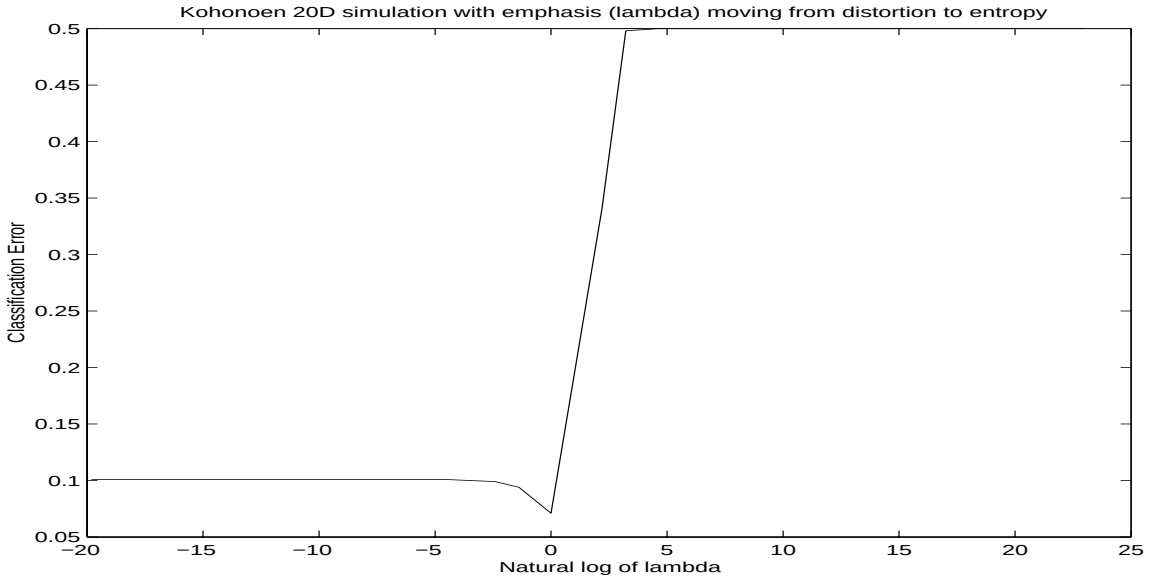

Figure 2.4: Kohonen simulation varying lambda with twenty feature dimensions and 200 training points

the entropy maximization is given preference over distortion minimization, and the weights become virtually uniform; the entropy, distortion, and classification accuracy again plateau. In the middle range of λ there is an optimal λ where the trade-off between distortion and entropy yields the lowest classification error.

2.7 As the number of training samples increase

The next simulation shows the effects of increasing the total number n of training samples. Consider the fairly simple problem of classifying with two feature dimensions. The Bayes' decision region for two dimensions is a circle with radius 1.923. The LIME algorithm has two parameters, the number of neighbors k and the trade-off between distortion and entropy, λ. The parameter λ was chosen by leave-one-out cross-validation [54] to be $\lambda = 1$ based on one initial run of 1000 test points and 200 training points and a two dimensional feature space. The parameter λ was then held constant and not tuned further. For the tricube weighting and the nearest neighbor algorithm the only parameter is the number of neighbors k. For each change in the number of training samples, an initial learning run with 1000 test points and the new number of training points was used to optimize the parameter k for each of the neighborhood methods. For each n number of samples, ten tests were run, with n new training samples and 1000 new test points generated for each test. Then the ten tests were averaged. The data representing the average over the ten tests (and thus 10,000 test points) are given in Table 2.1 and shown in Figure 2.5. Note that the scores for LIME, k-NN, and tricube are actually a little below the Bayes' error. This is possible because of the Bayes' error will be the minimum achievable when averaged with respect to the distributions, and hence only asymptotically when empirical distributions are used.

In Section 5.1, results are shown for increasing numbers of training samples for a different simulation. The performance is similar.

Num training samples for each class	LIME error	k-NN error	Tricube error
10	.344	.380	.371
50	.290	.298	.317
100	.279	.292	.300
1000	.265	.286	.274
10000	.260	.259	.261

Table 2.1: Kohonen simulation for two dimensions with increasing training data

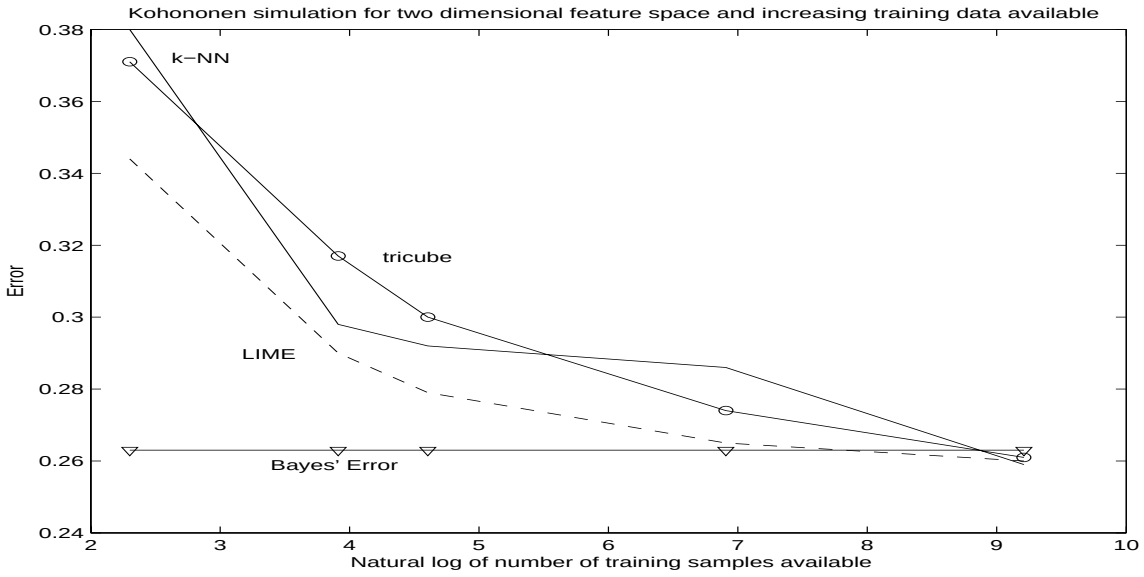

Figure 2.5: Kohonen simulation for two dimensions with increasing training data

2.8 Performance as the number of training dimensions increases

Table 2.2, and Figure 2.6, show the Kohonen simulation results as the number of feature dimensions increases from two to twenty. Each feature dimension is iid Gaussian as detailed earlier. Ten tests are run and averaged for each dimension. For each of the ten tests, 200 new training points are drawn, and 1000 new test points are drawn. The results in the table represent the average over the 10,000 test points from the ten independent tests.

The LIME algorithm has two parameters: k, the number of neighbors, and λ, the trade-off between distortion and entropycd. The parameter λ was chosen to be $\lambda = 1$ based on one initial run of 1000 test points and 200 training points and a two dimensional feature space. The parameter λ was then held constant and not further tuned. For the tricube weighting and the nearest neighbor algorithm the only parameter is the number of neighbors k. For each change in the number of feature dimensions, an initial learning run with 1000 test points and 200 training points was used to optimize the parameter k for each of the neighborhood methods. Particularly for the higher feature dimensions the optimal k for both the tricube weighting and the nearest neighbor method was $k = 1$, resulting in all the weight on the nearest neighbor and the same accuracy for these two methods.

The Bayes' decision region for two dimensions is a circle with radius 1.923, as discussed earlier. For the higher dimensions, the Bayes' decision region is a hypersphere with the radius noted in the Table 2.2. The table also notes the corresponding Bayes' error.

As the number of dimensions increases, the Bayes' error decreases, since each added feature carries information about which class a test point belongs to. In high dimensions there is a compounded asymmetry in the distribution of the test points such that it becomes increasingly likely that the nearest neighbors of a test point are from class 0, and not class 1. This causes severe bias in neighborhood methods that rely on distance.

Num of dim	Bayes' radius	Bayes' error	LIME error	k-NN error	Tricube error
2	1.923	.264	.279	.292	.300
3	2.355	.214	.219	.249	.257
5	3.041	.148	.163	.269	.254
10	4.300	.066	.102	.302	.302
15	5.266	.032	.093	.364	.364
20	6.081	.016	.075	.412	.412

Table 2.2: Kohonen simulation from two to twenty dimensions and 200 training points with Bayes' decision region radius and Bayes' error

The results suggest that as the number of dimensions increase and all points become roughly equally far away, it becomes more important to take into account the spatial relationships of the feature vectors.

2.9 LIME as an adaptive kernel

Kernel and weighted nearest neighbor techniques were reviewed in Section 1.3.1. LIME can be interpreted as a data-adaptive exponential kernel. The weights behave as if they were selected by an exponential kernel, but the slope of the exponential depends on the spatial relations of the training samples. In Section 3.2.1, it is shown that the LIME weights take on the exponential form

$$w_j(x, x_1, x_2, \ldots, x_k) = \frac{e^{-\alpha(x)^T(x_j - x)}}{\sum_j e^{-\alpha(x)^T(x_j - x)}}$$

where w_j denotes the weight corresponding to the training sample $\{x_j, y_j\}$.

An unsymmetrical kernel is not standard. The LIME kernel is adapting to the data points to control the local training sample distribution bias, and thus the lack of symmetry is purposeful and adaptive. This is a key point. The insidious effects of local bias can be reviewed in Section 1.2.2, and an example of LIME triumphing over that evil can be seen in the previous section in Figure 2.6. Four examples of the adaptive kernel of LIME weights are shown in Figure 2.7. In the plots, there is only one feature

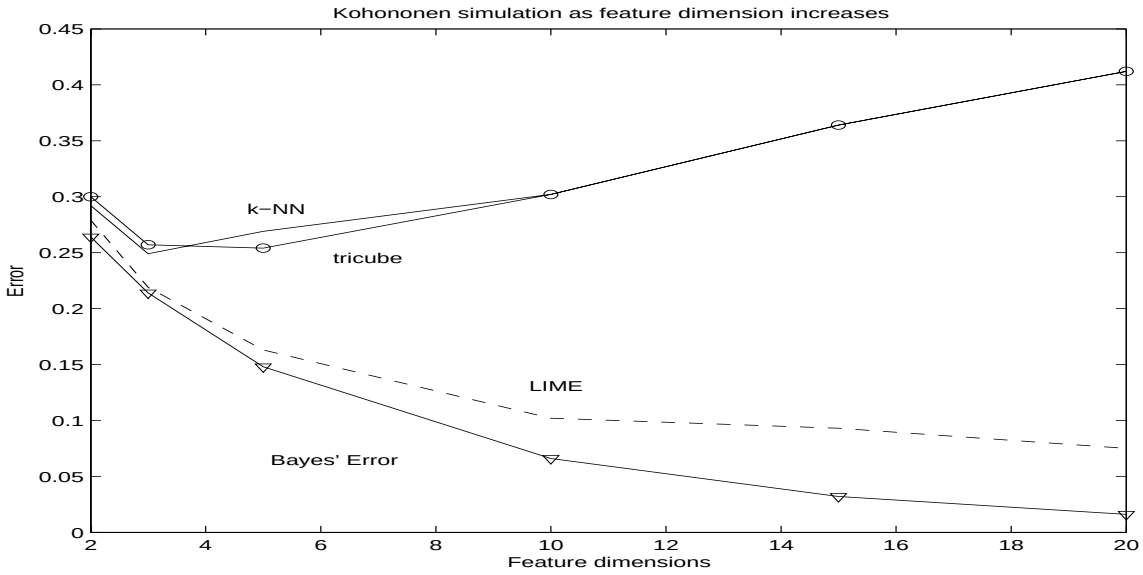

Figure 2.6: Kohonen simulation from two to twenty dimensions and 200 training points

dimension, the test point X is at the origin, and there are five neighborhood training samples as noted in the captions. The parameter λ is set at .001, which focuses on minimizing the distortion and then maximizing the entropy. In each plot the LIME weight is marked corresponding to each training sample. The weights trace out the adaptive exponential shape of the LIME weighting function. Note that in the fourth plot the test point X is outside the convex hull of the training points, and since the $\lambda = .001$ is relatively low, the weight solution focuses on minimizing the distortion yielding the weight solution $1, 0, 0, 0, 0$, a very fast decaying exponential.

Standard kernel algorithms run into bias at boundaries as discussed earlier in Section 1.3.1. These problems are due in part to the bias of the training sample distribution at boundaries (and due in part simply to the lack of nearby data at a boundary). The bias aspect of the boundary issue is controlled to a first order by the distortion criteria of the LIME algorithm, as is evidenced by the adaptive kernel.

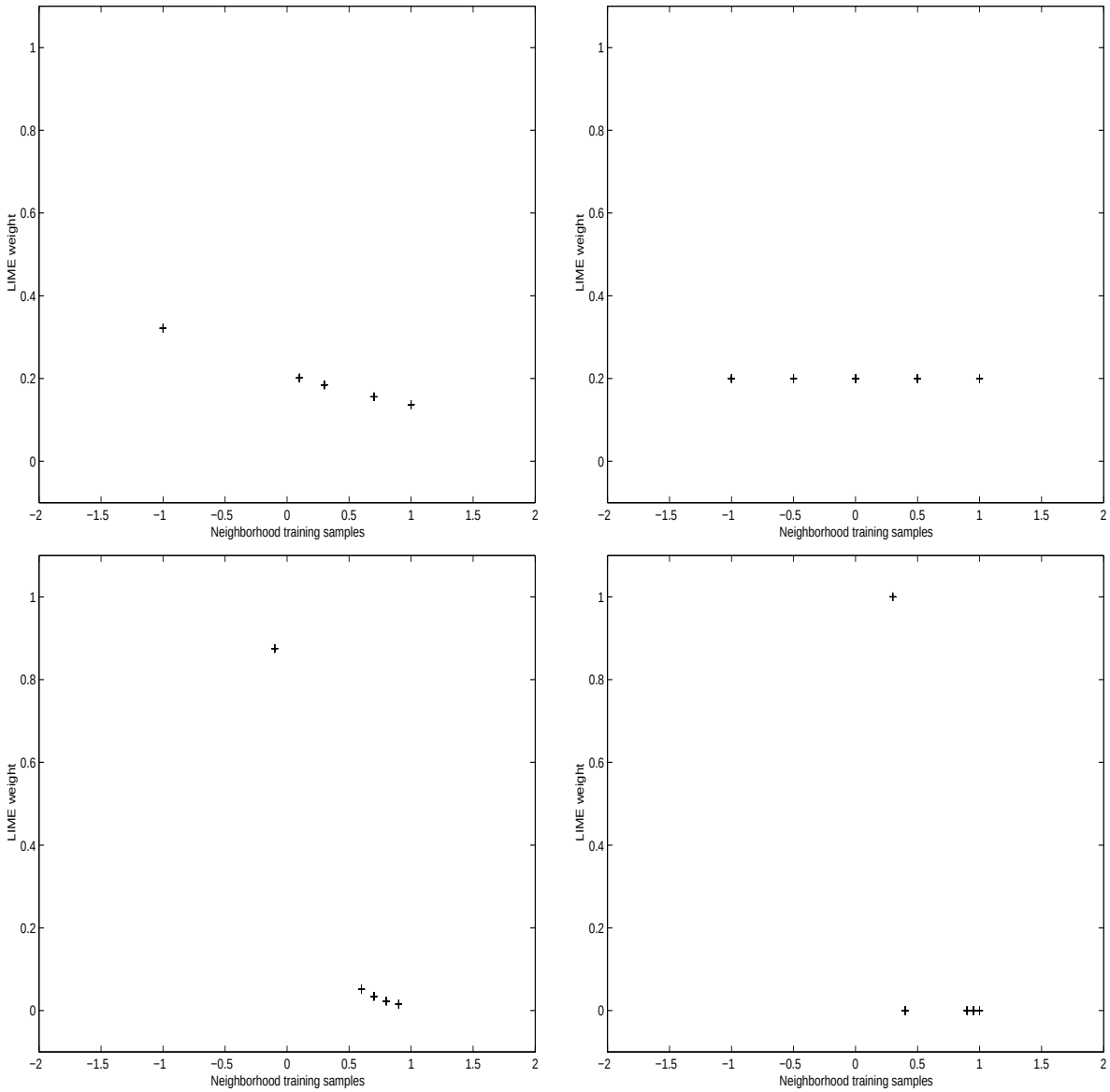

Figure 2.7: LIME weights form an adaptive kernel. Top left: neighborhood training samples at -1, .1, .3, .7 and 1. Top right: neighborhood training samples at -1, -.5, 0, .5 and 1. Bottom left: neighborhood training samples at -.1, .6, .7, .8 and .9. Bottom right: neighborhood training samples at .3, .4, .9, .95 and 1.

2.10 Analogs in source coding theory

There are a few similarities and differences between the LIME work and rate distortion theory.

2.10.1 LIME and variable rate coding

One approach to lossy compression is a variable rate coding of the source. In variable rate coding two common goals are to minimize the distortion caused by the lossy compression, and minimize the entropy of the compressed data source [39]. For example, entropy constrained vector quantization compresses by minimizing a functional of the form $D + \lambda H$ [22]. The LIME algorithm performs classification by minimizing a functional $D - \lambda H$. The LIME functional maximizes the bits needed to achieve a certain level of distortion. Which training samples are the most important ones is unknown, so LIME tries to give weight to all of them by maximizing the entropy of the weighting distribution. This is analogous to sending a message in many different languages to a stranger- safer to make it redundant because one does not know which language is important. Like telling the long version of the story instead of the short story. Like representing a message in the longest, most exhaustive, most descriptive, most diversified way possible...

2.10.2 LIME and fixed rate coding

In rate-distortion theory a source is coded in a way to minimize the average distortion between an original vector and its reproduction, $D(\hat{X}, X)$, traded-off with minimizing a rate R, where, for fixed rate, R represents the log of the cardinality of the set of reproduction codewords. This constrains R to be a discrete quantity; R cannot take on any value one would like. Thus, there are only certain achievable pairs of rate and distortion (R, D) for a source. The closure of the achievable (R, D) pairs is the rate distortion region, and its infimum of rates R for a given D is termed the operational rate distortion function $R(D)$. For optimal (R, D) pairs, minimizing $D + \lambda R$ is equivalent to minimizing R given a constraint that D be less than some D', where

$-\lambda$ is the slope at the minimizing point on the rate distortion function.

In LIME, a distortion D also measures the similarity between an original vector X and a reproduction $\hat{X} = \sum_j w_j X_j$. However, when LIME minimizes $D - \lambda H$, it minimizes the distortion D corresponding to a local case, and not the average distortion. The H in LIME represents the entropy of the weights of local training points. This differs from the rate R in two important ways. First, the weights can vary continuously, and thus H can take on any value within a range from 0 to $\log k$ nats, where k is the number of neighborhood samples. This means that the achievable (H, D) pairs vary continuously, and it is always possible to minimize $D - \lambda H$ and achieve pairs on the closure of the achievable region. The second important difference between the entropy H maximized in LIME and the rate R minimized in rate distortion theory is that R is a global property, and the H of interest is a local property.

In conclusion, there are fundamental differences between rate distortion theory and the LIME approach, including optimizing global versus local quantities, the entropy $H(w)$ over samples versus the cardinality of the reproduction set R, and optimizing discrete versus continuous random variables. Hence some of the mathematics developed for rate distortion theory may be applicable to LIME, and some may not.

2.11 Implementing the algorithm

The LIME algorithm can be implemented in different ways. Consider distortion functions that are convex with respect to w, such as mean squared error. For convex distortion functions $D(w)$, the functional $D(w) - \lambda H(w)$ is a sum of convex functions and is thus convex [13]. Then the weights w^* that minimize the functional can be found by convex optimization. It may not be necessary to use an iterative optimization algorithm. For l_1 distortion in a d dimensional feature space, such that $D(w) = \sum_{m=1}^{d} \| \sum_{j=1}^{k} w_j X_j[m] - X[m] \|)$, (where $Z[m]$ denotes the mth component of the vector Z), it is shown in Section 3.2.2 that the weight solution is known to within 2^d possibilities. A closed form solution also exists in the limit that $\lambda \to 0$ and the training samples lie on a regular grid with the neighborhood defined as the

vertices of a unit of the grid, as shown in Section 4.4.2.

A number of methods for solving for the maximum entropy distribution have been proposed, including hill-climbing, iterative projection, the damped Newton method, and iterative scaling [127].

In all of the numerical results presented in this work, LIME was implemented with an iterative convex optimization approach. Total squared error distortion $D(w) = \sum_{m=1}^{d}(\sum_{j=1}^{k} w_j X_j[m] - X[m])^2$ was used. The optimization of Step 2 was done with a primal-dual log-barrier solver implemented from Saunders [5]. This implementation belongs to the class of interior-point primal-dual methods, so called because all iterates remain in the interior of the inequality constraints (in this case, all iterates are strictly positive). Interior point methods require the Hessian matrix (the matrix of second derivatives, see Appendix for details) to be specified for the objective function $D - \lambda H$. For squared error distortion, the second derivatives all exist, and thus the Hessian matrix can be specified.

The interior-point implementation has several useful qualities. First, the iterates generated at each step of the algorithm remain strictly positive. Thus, the LIME objective function is guaranteed to always be well defined. Secondly, the algorithm makes explicit use of the positive definite, diagonal structure of the Hessian. Saunders' implementation is designed to handle large problems (e.g. 6000×6000 matrices). Further discussion of interior-point methods can be found in any standard book on optimization, such as [92].

2.12 Prototypes, editing, and clustering

Some neighborhood methods create and use a set of data prototypes that are more efficient than the original training samples; these methods are equally relevant for LIME. One approach is to shrink (edit) the original set of training samples in order to reduce storage requirements and decrease the time needed to find nearest neighbors. This approach dates back to Hart's work from 1968 [48]; recent work in this area includes [106] and [116].

Another set of approaches create prototype samples based on the original training samples. The prototype samples may be selected to speed up finding the nearest neighbor, to reduce storage requirements, or to reduce noise in the feature vectors. The Lloyd algorithm [39] (also known as k-means) or other clustering methods [49], [86] can be used to replace clusters of training samples with their centroid. This is advantageous in reducing the number of samples and the speed of searching for nearest neighbors. It may also compensate for some noise in the feature vectors. Depending on the clustering method and the underlying distribution of training samples, this approach may reduce some of the bias resulting from the probability distribution of the training features. However, a relatively large number of samples may be needed for a clustering method to obtain bias or noise reduction advantages without skewing the resulting decision boundary.

Learning vector quantization(LVQ) [76] uses a variation of the k-means algorithm to create prototypes for classification problems. LVQ uses information about the training sample observations Y to train a set of prototype samples.

Two simulations in [54] show that prototype methods can marginally outperform basic nearest neighbor methods. The LIME algorithm could be used with prototype samples instead of the original training samples. Prototype samples may also be used to achieve compression of the data set. Gray and Olshen propose and compare the classification and compression performance of prototype algorithms in [42].

Other research into speeding up the nearest neighbor search uses fast search methods or builds trees of prototype samples to decrease the average or worst case search time, including [87], [7], [74], and [82]. Similarly, creating and using a set of prototype training samples may be an efficient implementation for the LIME algorithm.

2.13 Related work

Research related to the LIME algorithm includes other generalizations of linear interpolation, other applications of information theory to statistical learning, and other uses of a functional that trades-off distortion and entropy. Very few generalizations

of linear interpolation were found. There are methods to generalize linear interpolation for points on a regular grid in two or three dimensions. That work is further discussed in Chapter 4 on regular grid applications. The next few paragraphs are a tour of other related work.

2.13.1 Maximum entropy and learning

A number of researchers have applied information theory to statistical learning, including the principle of maximum entropy.

The classic approach to using maximum entropy for regression has its roots in astronomical imaging [16]. The premise is that there is definite prior information available about the function (discrete or continuous) to be estimated. However, the data are in the form of known moments or integrals, or bounds of the desired function (for example, an image). The known information does not uniquely specify the desired function. The principle of maximum entropy is used to solve for a unique regression estimate of the desired function. [127]. A recent example of work in this classical vein is Csiszár et al.'s 1999 paper [27] which extends the maximum differential entropy approach for use with a range of prior information such as a known Sobolev norm for the desired function. Gamboa and Gassiat, in their work known as maximum entropy on the mean (MEM) [38], approach the specific classical problem of an ill-posed linear system of equations and instead of applying the maximum entropy principle directly, they apply the principle of minimum relative entropy (maximizing entropy with respect to a reference distribution) repeatedly with different stochastic reference distributions to arrive at an estimate.

The classical use of the principle of maximum entropy for regression and the LIME approach are quite different. The classical approach begins with information about global integrals or global constraints. On the other hand, LIME models a test point as the average of local training points. The classical approach maximizes the entropy over the distribution of the final estimated function. LIME maximizes the entropy of the weighting of local training points. Generally, the classic approach assumes that the constraints or prior information are definite. In 1999, Campbell proposed [18]

indefinite moment constraints for maximum entropy problems (see Section 2.13.2 and Section 3.2.2 for further discussion). The LIME algorithm also uses a soft moment constraint.

An unrelated use of the principle of maximum entropy in learning, is work by Jaakkola et al. [58]. They proposed using the maximum entropy principle to select a parametric model for two-class classification. Instead of finding a single parameter setting, they find a probability distribution over the parameters so that the convex combination of resulting discrimination functions has both the maximum entropy of the parameter distribution and minimizes the empirical error on the training set. This work is quite different from the LIME approach, most directly because it is parametric. Also, the Jaakkola et al. approach is a global approach, whereas the LIME is a local approach. Jaakkola's approach has the disadvantages and advantages of most parametric approaches.

Another use of the principle of maximum entropy for regression is fitting cubic splines to noisy data [45]. This is also a parametric approach.

2.13.2 The functional D - λ H

LIME minimizes the functional $D(w) - \lambda H(w)$. The same functional crops up in a few different places.

One relevant appearance is a 1998 paper by Campbell [18]. Campbell proposes the general problem of solving for a probability distribution based on a prior and uncertain side information about the mean. With his formulation, one would like to estimate a probability distribution w over a set of points $\{x_1, x_2, \ldots, x_k\} \in \mathcal{R}$ based on a prior distribution q and uncertain side information that $\sum_j w_j x_j = x$. Campbell proposes minimizing a functional that trades off minimizing the Kullback-Leibler distance ($\mathcal{D}$) to the prior and minimizing the absolute value difference to the uncertain mean,

$$F(w) = \mathcal{D}(w, q) + \lambda |\sum_{j=1}^{k} w_j x_j - x|.$$

Campbell's parameter λ, like the LIME λ, is chosen by the practitioner to represent

the desired trade-off between the matching the $\sum w_j x_j$ and x and matching the chosen distribution w to the prior q. To clarify the link to LIME, let the prior q be a uniform distribution such that $q_j = 1/k$ for all j. Since minimizing the relative entropy with respect to a uniform distribution is equivalent to maximizing the entropy, Campbell's functional solves the same problem as the LIME functional where the LIME distortion D is absolute value and the feature vectors are one dimensional. In the same work [18], Campbell presents a closed form solution for the optimal distribution. We discuss his closed form solution, and prove a multidimensional generalization in Section 3.2.2.

The functional $D - \lambda H$ also describes the Helmholtz free energy of a system (or Helmholtz thermodynamic potential) [17], $U - TS$, where U signifies the system's internal energy, T the temperature of the system, and S the Boltzmann entropy, $S = k \log P$, in which P is the number of microstates that could achieve the system's macrostate and k is a constant. Interpreting the relationship between the Helmholtz free energy of a system, and the LIME use of the functional, is left to the more poetic reader.

The functional $D - \lambda H$ also appears in the clustering literature for deterministic annealing [109]. In deterministic annealing, the functional $D - TH$ is minimized for progressively lower T. In this context, T is a lagrangian multiplier representing temperature, analogous to the Helmholtz free energy. The D is a distortion that measures the error of the clustering,

$$D(x, y) = \sum_x p(x) d(x, y(x))$$

where x denotes a discrete source vector; $y(x)$ denotes its best reproduction codevector from a codebook Y; $p(x)$ is the probability mass function of the source vector x; and $d(r, s)$ denotes the distortion between r and s. In practice, an empirical formulation of the distortion is used where the source vectors are the data. The entropy H in the deterministic annealing functional is taken as the Shannon entropy over the joint distribution of source vectors x and representation vectors y,

$$H(x, y) = -\sum_x \sum_y p(x, y) \log p(x, y).$$

A key problem in clustering is getting trapped in local minima when trying to minimize the distortion D directly. Deterministic annealing attempts to avoid local minima traps by valuing randomness (at least at the start) by a high T value, and then letting T fall to zero over multiple iterations.

Although both LIME and deterministic annealing attempt to reduce the distortion of a reproduction, deterministic annealing maximizes the conditional randomness of the reproductions given the source vectors, whereas LIME is maximizing the randomness of which training samples are used to create the reproduction. Also, deterministic annealing iteratively lowers T so that ultimately the functional is the classic clustering functional: minimize the reproduction distortion D, with no dependence on any entropy.

Rose [109] also explains how to use deterministic annealing for supervised learning, employing the functional $D - TH$ again. His implementation is parametric and lets the distortion D measure the error from some local parametric model (such as a decision tree) to the training data points. The classifier partitions the feature space. The entropy H is a function of the conditional probability distribution for the partitions given a test vector x. The Lagrangian multiplier T is driven to zero so that in the final iteration the TH term of the functional has basically disappeared. This technique is related to the expectation maximization algorithm [86] in that the partitioning is 'soft.' Rose emphasizes that H in this context measures the average level of uncertainty in the partition decisions. This approach is clearly different than LIME. Deterministic annealing is parametric, LIME is nonparametric. The distortion in LIME is measured in the feature domain, whereas in the deterministic annealing setting it is measured in the output domain (a flavor of empirical risk minimization). Deterministic annealing considers the entropy H as a measure of the uncertainty in partitioning decisions, whereas LIME uses the entropy H as a tool to measure the distribution of weighting to near neighbors.

Similar to deterministic annealing clustering, Karayiannis [68] proposes an unsupervised learning algorithm which transitions from a fuzzy maximum entropy clustering to a hard clustering. At the beginning of his algorithm, the entropy of which cluster each data point belongs to is maximum, and this entropy is slowly reduced.

He considers the functional $(1-\alpha)D - \alpha H$. This work is conceptually closer to our use of the maximum entropy principle: Karayiannis considers the entropy of the membership distribution over different clusters; LIME considers the entropy of the probability distribution over near-by training points.

2.13.3 Other applications of information theory to supervised learning

In this section we discuss some applications of information theory to supervised learning that do not utilize the maximum entropy concept but are relevant to the issues at hand.

Complexity regularization

A number of estimation algorithms trade-off between empirical accuracy and complexity. Such methods are willing to 'pay' in terms of empirical error for a simpler model, defended by Occam's razor, the minimum discrimination length principle, or other complexity-based costs. Work in this area includes [85], [8], and [123]. These methods can be motivated in part by a Vapnik-Chervonenkis analysis [78]. Najmi [91] proposed a parametric model-fitting principle that trades-off maximizing likelihood with minimizing entropy; given a data point x and a set of models $\mathcal{M}$, $\max_{\theta \in \mathcal{M}} P(x|\theta) - H(\theta)$.

Unlike the above regularization learning approaches, LIME does not trade-off empirical accuracy and complexity. Instead, the functional $D - \lambda H$ being minimized trades off distortion in the feature space (not in the output, or observation space) with diversity in the use of the training data.

The cost of features

Chou in [21] shows that decision trees can be seen as a variable rate source code. All classification algorithms that divide the feature space into regions, including LIME, can be implemented as a decision tree. Chou's insight was that by splitting the feature space carefully and with simple regions, the number of 'tests' that have to be done to classify a point can be minimized. This approach is particularly important in medical

applications where getting data is expensive and possibly dangerous. LIME is unlikely to identify regions that split nicely along dimensions, and thus from the point of view of this metric, LIME is likely to create too complex (and hence expensive) a decision tree.

The cost of data storage

Another application of information-theoretic ideas to pattern recognition is due to Pearl, who applied rate-distortion theory to the problem of data-storage versus error rate for supervised pattern recognition and classification [98]. Given a set of data, one can achieve a certain accuracy on queries about the same or related data. However, if there is not room to store the complete data set, then average accuracy is expected to decrease.

Pearl posed this trade-off between the amount of data stored and the accuracy on classification queries as a trade-off between rate (R) and distortion (D). In his framework, D is the average probability of answering a query wrong. R is the number of bits of data that must be stored. Given this framework he applies the insight and theorems of the Shannon $R-D$ literature [10]. Pearl considers the theoretical storage-error trade-off without respect to a particular algorithm or type of algorithm.

Obviously, if the underlying space can be well-characterized by a parametric model the necessary R may be quite low. For non-parametric models the R is necessarily much higher as these models attempt to characterize more complexity of the space. In the proposed LIME algorithm, the goal is the lowest possible D. There is no restriction on how much data can be used. The interesting problem of finding the minimal storage needed for a given distortion is not considered.

Chapter 3

Asymptotics, bounds, and robustness to noise

The success of any project really
depends on managing expectations.

Naomi Karten

In this chapter, one finds results about asymptotics, bounds, and robustness to noise. First, the limiting cases as λ goes to zero or to infinity are considered. In Section 3.2, the LIME weights are shown to have exponential form, and for l_1 distortion, the exponential form can be specified to within 2^d possibilities for a d-dimensional feature space (for a one-dimensional feature space, the exponential form can be specified exactly). Using the known d-dimensional exponential form of the LIME weights, it is shown in Section 3.3 that the LIME algorithm is weakly consistent, as per Stone's conditions.

Robustness to noise is treated in Section 3.4. In Section 3.4.3, a variation of the law of large numbers is shown to hold for LIME weights. That leads to nice theoretical properties for well-behaved additive noise on the training features or on the training observations.

The chapter ends with consideration of which functions are fit exactly by LIME regression. More results about functions that are fit exactly appear in Chapter 4 on

regular grids.

Readers may find it useful to review Section 1.0.2 on notation choices.

3.1 LIME weight distributions for extreme λ

For the LIME functional $D - \lambda H$, in the limit as λ grows to infinity the maximum entropy objective dominates. Similarly, as λ shrinks to zero, minimizing the distortion takes precedence. This intuition is formalized in the following results.

Lemma 1 *Let $w_j^u = \frac{1}{k}$ for all n and for all $j = 1, \ldots, k$, and let $F(w, \lambda) = D(w) - \lambda H(w)$. Then,*

$$\lim_{\lambda \to \infty} \frac{\inf_w F(w, \lambda) - F(w^u, \lambda)}{\lambda} = 0 \tag{3.1}$$

Proof: By definition,

$$\inf_w F(w, \lambda) \leq F(w^u, \lambda).$$

Then, since $\lambda \geq 0$,

$$\frac{F(w^u, \lambda)}{\lambda} - \inf_w \frac{F(w, \lambda)}{\lambda} \geq 0,$$

and thus,

$$\liminf_{\lambda \to \infty} \left(\frac{F(w^u, \lambda)}{\lambda} - \inf_w \frac{F(w, \lambda)}{\lambda} \right) \geq 0. \tag{3.2}$$

Also, coupling

$$\frac{D(w^u)}{\lambda} - H(w^u) - \inf_w \left(\frac{D(w)}{\lambda} - H(w) \right) \leq \frac{D(w^u)}{\lambda} + -H(w^u) + \sup_w H(w), \tag{3.3}$$

with

$$\begin{aligned} & \lim_{\lambda\to\infty}\left(\frac{D(w^u)}{\lambda}+\sup_w H(w)-H(w^u)\right) \\ = & \lim_{\lambda\to\infty}\left(\sup_w H(w)-H(w^u)\right) \\ = & \sup_w H(w)-H(w^u) \\ = & 0, \end{aligned}$$

establishes that

$$\limsup_{\lambda\to\infty}\left(\frac{F(w^u,\lambda)}{\lambda}-\inf_w\frac{F(w,\lambda)}{\lambda}\right)\leq 0 \tag{3.4}$$

Combining (3.2) and (3.4) yields the lemma. q.e.d.

Corollary 1 *Given the conditions of Lemma 1,*

$$\lim_{\lambda\to\infty}\|w^*-w^u\|_1\to 0$$

where $w^*=\arg\inf_w F(w,\lambda)$.

Proof: From Lemma 1, it can be concluded that

$$\lim_{\lambda\to\infty}(H(w^*)-H(w^u))=0,$$

or equivalently,

$$\lim_{\lambda\to\infty}H(w^*)=H(w^u)=\log k. \tag{3.5}$$

A result from information theory (pages 102-103, [41]) relates the l_1 distance and the relative entropy $\mathcal{D}$ of two pmfs p and q,

$$\|p-q\|_1\leq\sqrt{2\mathcal{D}(p\|q)} \tag{3.6}$$

Then

$$\begin{aligned}\lim_{\lambda\to\infty}\|w^* - w^u\|_1 &\leq \lim_{\lambda\to\infty}\sqrt{2\mathcal{D}(p\|q)} \\ &= \lim_{\lambda\to\infty}\sqrt{2(\log k - H(w^*))} \\ &= 0\end{aligned}$$

where the last line follows from (3.5). q.e.d.

A result can also be stated for the limit $\lambda \to 0$.

Lemma 2 *Let* $w^* = \arg\sup_w (H(w)|D(w) = \inf_a D(a))$. *Then*

$$\lim_{\lambda\to 0}\left(\inf_w F(w,\lambda) - F(w^*,\lambda)\right) = 0 \tag{3.7}$$

Proof: By definition,

$$F(w^*,\lambda) - \inf_w F(w,\lambda) \geq 0$$

and thus

$$\liminf_{\lambda\to 0}\left(F(w^*,\lambda) - \inf_w F(w,\lambda)\right) \geq 0. \tag{3.8}$$

Also

$$\begin{aligned}&\limsup_{\lambda\to 0}\left(D(w^*) - \lambda H(w^*) - \inf_w (D(w) - \lambda H(w))\right) \\ \leq\ &\limsup_{\lambda\to 0}\left(D(w^*) - \inf_w (D(w) - \lambda H(w))\right) \\ \leq\ &\limsup_{\lambda\to 0}\left(\sup_w \lambda H(w)\right) \\ =\ &\limsup_{\lambda\to 0}\lambda\log k \\ =\ &0,\end{aligned}$$

which establishes that

$$\limsup_{\lambda\to 0}\left(F(w^*,\lambda) - \inf_w F(w,\lambda)\right) \leq 0. \tag{3.9}$$

Combining (3.8) and (3.9) yields the lemma. q.e.d.

A related conjecture is that

$$\lim_{\lambda \to 0} \arg\inf_{w} F(w, \lambda) = \arg\sup_{w} \Big(H(w) | D(w) = \inf_{a} D(a) \Big) \tag{3.10}$$

In the special case that a test point is within the convex hull of its neighborhood points, and $D(w)$ is taken as the l_1 distance, then the conjecture can be shown to hold (see Theorem 3 and Corollary 3).

3.2 Exponential form for the optimal distribution

In this section it is shown that the LIME weights have an exponential form. All the results are for arbitrary d-dimensional feature spaces unless otherwise noted. For l_1 distortion, the exponential is shown to decay as a function of λ. The exponential form means that LIME can be treated as a data-adaptive kernel as was discussed in Section 2.9. The exponential form will be used to show consistency in Section 3.3.

For one-dimensional l_1 distortion (absolute value distortion), the closed form solution enables one to substitute and solve for the weights directly. For d-dimensional feature spaces and l_1 distortion, we show that the solution is known to be one of 2^d exponential possibilities, whose distortion and entropy can be calculated to determine the exact solution. Later, in Section 4.4.2, a closed form solution is derived for any distortion and $\lambda \to 0$ such that the distortion is constrained to zero and then the entropy maximized.

It is well known in the field of information theory that constrained maximum entropy problems result in exponential distributions. Two different proofs for general cases can be found in Kullback [79] and Cover and Thomas [25]. These proofs assume that the entropy is being maximized subject to some expectation constraint(s). Instead of a constraint $D = 0$, we are interested in a trade-off between minimizing the distortion D and maximizing the entropy H. Similar problems arise in rate distortion theory, and those solutions have also been shown to have an exponential form [25], [37]. In Section 3.2.1, it is shown that the LIME weights will have an

exponential form for any distortion function that is a norm $D(a,b) = \|a-b\|$. The simple proof presented provides an insight: the weights that minimize $D(w) - \lambda H(w)$ for a test point x are the same as the weights that maximize $H(w)$ with the mean constraint $D(w) = 0$ for a different test point x'.

Then, in Section 3.2.2 it is shown that if l_1 distortion is used, the weights have an exponential form with decay dependent on λ. Note that the weighted reconstruction $\sum_j w_j x_j$ is the expectation with respect to the distribution $\{w_1, w_2, \ldots, w_k\}$ on the points $\{x_1, x_2, \ldots, x_k\}$. Thus minimzing $D(w)$ can be seen as attempting to fit a mean constraint to the data set. Campbell, in his work [18], considered the problem of uncertain side information about the mean of an unknown pmf over a given set of scalar points. His formulation results in the same optimization problem as LIME. Campbell gives a closed form solution for the optimal probability mass function for the scalar case with absolute value distortion, with the assumption that the uncertain 'mean' $x \in \mathcal{R}$ lies within the range of the set of events $\min_j x_j \leq x \leq \max_j x_j$. We show an extension of this result for LIME for the l_1 distortion over a d-dimensional feature space and without a range constraint on $x \in \mathcal{R}^d$.

3.2.1 Exponential form of weights for any distortion

To show that the LIME weights have an exponential form, we first review a theorem for solutions of the standard maximum entropy problem with a mean constraint. Then we present a simple proof extending this result to the LIME weights.

The theorem given here is a special case of Theorem 11.1.1 from Cover and Thomas [25], pages 267-268. An older proof is due to Kullback [79], who proved the general case that minimizing relative entropy given a constraint yields an exponential distribution.

Theorem 1 (Cover and Thomas) *Consider the points x and $x_j \in \mathcal{R}^d$ for $j = 1, \ldots, k$. If $w^*(x)$ has the form $w_j^*(x) = \gamma e^{-\alpha^T x_j}$ for $j = 1, \ldots, k$, where α and γ satisfy*

C1) $\gamma = \frac{1}{\sum_j e^{-\alpha^T x_j}}$

C2) $\sum_j \gamma x_j e^{-\alpha^T x_j} = x$,

then $w^*(x)$ uniquely maximizes the Shannon entropy $H(w)$ subject to the mean constraint $\sum_j w_j^*(x)x_j = x$.

Conversely, if w^* maximizes $H(w)$ subject to the mean constraint, then it must have the form $w_j^*(x) = \gamma e^{-\alpha^T x_j}$, where γ and α satisfy C1 and C2.

Next, we establish that the LIME weights will have the same exponential form as solutions for the problem of maximizing entropy with a mean constraint as stated in Theorem 1.

Theorem 2 *Consider the points x and $x_j \in \mathcal{R}^d$ for $j = 1, \ldots, k$. If w^* is a pmf that minimizes the functional $F(w) = D(w) - \lambda H(w)$, where $D(w) = z(\| \sum_j w_j x_j - x\|)$, for any norm $\|\cdot\|$ (for example, an l_p norm) and any monotonic function z, then w^* has the form $w^* = \gamma e^{-\alpha^T x_j}$.*

Proof: Suppose that w^* is an optimal LIME weight distribution, and let $\hat{x} = \sum_j w_j^* x_j$.

Consider the new problem of solving for the maximum entropy weighting distribution $w^\sharp$ over the $\{x_1, x_2, \ldots, x_k\}$ with the strict mean constraint $\sum_j w_j^\sharp x_j = \hat{x}$. From Theorem 1, it is known that the minimizing distribution has the form $w_j^\sharp(x) = \gamma^\sharp e^{-\alpha^T x_j}$ for $j = 1, \ldots, k$.

Next, consider the distortion resulting from using the new weights $w^\sharp$ to reconstruct x (instead of $\hat{x}$),

$$D(w^\sharp) = z\left(\| \sum_j w_j^\sharp x_j - x\|\right).$$

Similarly, the optimal LIME weights w^* result in distortion

$$D(w^*) = z\left(\| \sum_j w_j^* x_j - x\|\right).$$

Since $\hat{x} = \sum_j w_j^* x_j$, and the weights $w^\sharp$ were chosen so that $\sum_j w_j^\sharp x_j = \hat{x}$,

$$D(w^\sharp) = z\left(\|\hat{x} - x\|\right) = D(w^*).$$

Thus the weights $w^\sharp$, which solve the maximum entropy with a strict mean constraint of $\hat{x}$, yield the same distortion as the LIME weights w^*. Then the weight

distributions w^* and $w^\sharp$ must be the same, because if one set of weights had a higher entropy it would have been the solution to the other's problem, since the distortions $D(w^\sharp)$ and $D(w^*)$ are equal, and the maximum entropy with mean constraint $w^\sharp$ is unique (see Theorem 1).

Hence we conclude that the LIME weights will have the same form as maximum entropy weights for a strict mean-constraint, that is, $w_j^*(x) = \gamma e^{-\alpha^T x_j}$ for all j. q.e.d.

3.2.2 Exponential weights for l_1 distortion and scalar feature space

Campbell [18] considered the problem of uncertain side information about the mean of an unknown distribution. We review Campbell's closed form solution for scalar feature spaces with absolute value distortion and then present a constructive proof of Campbell's solution. In Section 3.2.3, the theorem is extended to multi-dimensional random variables x with l_1 distortion. The section ends with a note on implementation.

In Campbell's formulation [18], one would like to estimate a pmf w over a finite set of points $\{x_1, x_2, \ldots, x_k\} \in \mathcal{R}$ given prior pmf q and uncertain side information that $\sum_j w_j x_j = x$. Specifically, he proposes to minimize a penalized distortion,

$$\lambda \mathcal{D}(w\|q) + |\sum_{j=1}^{k} w_j x_j - x| \tag{3.11}$$

where the minimum is over all pmfs w and the user chooses $\lambda > 0$, in order to trade-off Kullback-Leibler information ($\mathcal{D}$) with the absolute value of the accuracy of the approximation. Campbell further assumes that $\min_j x_j < x < \max_j x_j$. He presents a closed form solution for the minimizer.

For our purposes of maximizing entropy, the prior q is a uniform distribution such that $q_j = 1/k$ for all j. We will make this assumption from here on.

In this section, we present a constructive proof for Campbell's solution for one-dimensional feature spaces without any constraint on the range of x. The proof

involves two cases: the case where the distortion between x and the optimal reproduction $\sum_j w_j x_j$ is greater than zero, and the case where the distortion is zero. Then, we present an analogous theorem for d-dimensional vectors x. In the multi-dimensional case, the solution is not exactly known, but is narrowed to one of 2^d possibilities.

Before presenting the results, some necessary definitions and lemmas will be given, as well as a review of the exact penalty function theorem from optimization theory and a useful corollary. The minimization problems are defined in the style of optimization theory. These minimization problems could equivalently be considered problems of defining an infimum over a set and determining when it can be achieved.

After the one dimensional proof we consider the multi-dimensional case with $x \in \mathcal{R}^d$.

Problem 1 (l_1 LIME minimization problem) *Given $x \in \mathcal{R}^d$, and $x_j \in \mathcal{R}^d, j = 1, \ldots, k$, and $\lambda > 0$, the l_1 LIME minimization problem is to minimize the functional,*

$$F(w, \lambda) = \left(\| \sum_{j=1}^{k} w_j x_j - x \|_1 + \lambda \sum_{j=1}^{k} w_j \log w_j \right) \tag{3.12}$$

over all pmfs w (so that $\sum_{j=1}^{k} w_j = 1$ and $w_j \geq 0$ for all j), where $\| \circ \|_1$ denotes the l_1 norm.

Some observations about the l_1 LIME minimization problem defined above will be useful later. First, note that by the Weierstrass Theorem (see Appendix), a minimizer w^* for the problem exists since the objective function F is a continuous function of w and the constraint region ($\sum_j w_j = 1$) is compact and not empty. Moreover, note that the objective function is convex since it is a weighted sum of two convex functions (negative entropy and the l_1 norm).

To determine the one-dimensional solution (Theorem 4), the following two lemmas will prove useful. The first lemma shows that if the average of the set of neighborhood samples is greater than x, then the LIME weight distribution w^* will construct a reproduction of x that is also greater than x, that is, if $\sum_{j=1}^{k} x_j/k > x$ then $\sum_{j=1}^{k} w_j^* x_j > x$.

Lemma 3 *Given $x \in \mathcal{R}$, and $x_j \in \mathcal{R}, j = 1, \ldots, k$ such that $\sum_{j=1}^{k} x_j/k > x$, and $\lambda > 0$. Suppose that w^* is a minimizer of the l_1 LIME minimization problem, then*

$$\sum_{j=1}^{k} w_j^* x_j \geq x. \tag{3.13}$$

Proof: The proof is by contradiction. Consider any distribution of weights $w^\sharp$ for which (3.13) does not hold, so that $\sum_{j=1}^{k} w_j^\sharp x_j < x$. We show that $w^\sharp$ cannot be optimal.

Let $x^\sharp = \sum_{j=1}^{k} w_j^\sharp x_j$. Let $w_j^u = 1/k$ for $j = 1, \ldots, k$, and $x^u = \sum_{j=1}^{k} x_j/k$. Note that w^u has the maximum entropy of all possible weight distributions.

First, consider the case that $\|x^\sharp - x\| \geq \|x^u - x\|$. Then w^u is a weight distribution that results in equal or smaller distortion than $w^\sharp$, and has strictly greater entropy than $w^\sharp$. Therefore, $w^\sharp$ cannot be the minimizing distribution of the l_1 LIME minimization problem.

Conversely, consider the case $\|x^\sharp - x\| < \|x^u - x\|$. Define a point $\hat{x} = x^\sharp + 2\|x - x^\sharp\|$. In the next paragraphs we show that $\hat{x}$ is associated with a set of weights $\hat{w}$ such that $D(\hat{w}) - \lambda H(\hat{w}) < D(w^\sharp) - \lambda H(w^\sharp)$. Therefore, $w^\sharp$ cannot be the minimizing distribution of the l_1 LIME minimization problem.

Since $x^\sharp < \hat{x} < x^u$, there exists some $\beta \in [0, 1]$ such that

$$\hat{x} = \beta x^\sharp + (1 - \beta) x^u.$$

Substituting for $x^\sharp$ and x^u yields,

$$\begin{aligned} \hat{x} &= \beta \sum_{j=1}^{k} w_j^\sharp x_j + (1 - \beta) \sum_{j=1}^{k} w_j^u x_j \\ &= \sum_{j=1}^{k} (\beta w_j^\sharp + (1 - \beta) w_j^u) x_j. \end{aligned}$$

Then, there is a weight distribution with components $\hat{w}_j = \beta w_j^\sharp + (1 - \beta) w_j^u$ for which $\sum_j \hat{w}_j x_j = \hat{x}$, and thus $\hat{w}$ yields the same distortion as $w^\sharp$. The entropy of $\hat{w}$ is $H(\hat{w}) = H(\beta w^\sharp + (1 - \beta) w_u) \geq \beta H(w^\sharp) + (1 - \beta) H(w^u)$ (since entropy is a

concave function). Since $H(w^u) > H(w^\sharp)$, $H(\hat{w}) > H(w^\sharp)$. Therefore, $w^\sharp$ cannot be the minimizing distribution of the l_1 LIME minimization problem. Also, $\hat{w}$ satisfies (3.13).

In conclusion, for any $w^\sharp$ which does not satisfy (3.13), it has been shown that there exists another weight distribution, w^u or $\hat{w}$, that does satisfy (3.13) and results in a lower $D(w) - \lambda H(w)$ for any λ. Thus $w^\sharp$ cannot be optimal. q.e.d.

Next, we define the constrained l_1 LIME minimization problem, which restricts possible solutions to pmfs that result in zero distortion. Then, Lemma 4 specifies the conditions under which a minimizer exists for the constrained l_1 LIME minimization problem and gives a closed form solution for any such minimizer. The constrained l_1 LIME minimization problem (and associated Lemma 4) will be useful in deriving closed form solutions (Theorem 4 and Theorem 5) for the l_1 LIME minimization problem (problem 1).

Problem 2 (Constrained l_1 LIME minimization problem) *Given $x \in \mathcal{R}^d$, and $x_j \in \mathcal{R}^d, j = 1, \ldots, k$, the constrained l_1 LIME minimization problem is to minimize the functional,*

$$F(w) = \sum_{j=1}^{k} w_j \log w_j \tag{3.14}$$

over all pmfs w (such that $\sum_{j=1}^{k} w_j = 1$ and $w_j \geq 0$ for all j) with the constraints that $\sum_{j=1}^{k} w_j x_j[m] - x[m] = 0$ for all $m = 1, \ldots, d$.

Some observations about the constrained l_1 LIME minimization problem defined above will be useful. First, note that a minimizer for the problem might not exist because there may be no feasible vectors w, that is, there may be no w that satisfies the constraints. Also, note that the objective function is convex (negative entropy).

The next lemma, Lemma 4, specifies the conditions under which a minimizer exists for the constrained l_1 LIME minimization problem and gives a closed form solution for any such minimizer. The lemma uses the second order necessary and sufficiency conditions for a minimizer and the notion of a Lagrange multiplier; these terms are

reviewed in the Appendix and in the proof. The existence conditions and form of the constrained solution will determine solution for the more general unconstrained problem (Problem 1).

Recall from the preliminaries on notation(Section 1.0.2) that $z[m]$ denotes the mth component of the vector z.

Lemma 4 *Given $x \in \mathcal{R}^d$ and $x_j \in \mathcal{R}^d, j = 1, \ldots, k$, then a necessary and sufficient condition for a solution to exist for the constrained l_1 LIME minimization problem, is that there exist a vector $\kappa \in \mathcal{R}^d$ which solves*

$$\sum_{j=1}^{k}(x_j[m] - x[m]) \left(\frac{e^{\sum_{r=1}^{d} -\kappa[r](x_j[r]-x[r])}}{\sum_{i=1}^{k} e^{\sum_{p=1}^{d} -\kappa[p](x_i[p]-x[p])}} \right) = 0,$$

for $m = 1, \ldots, d$, in which case

$$w_j^*(\kappa) = \frac{e^{\sum_{r=1}^{d} -\kappa[r](x_j[r]-x[r])}}{\sum_{i=1}^{k} e^{\sum_{p=1}^{d} -\kappa[p](x_i[p]-x[p])}}$$

for $j = 1, \ldots, k$.

Proof: First, note that the objective $-H(w)$ and all the constraint functions, $\sum_{j=1}^{k} w_j x_j[m] - x[m]$ for $m = 1, \ldots, d$, and $\sum_j w_j - 1$, are twice continuously differentiable in w.

In this proof, we initially conjecture that the non-negativity constraint $w_j \geq 0$ for all j is non-binding (also known as inactive), which means that solving the constrained l_1 LIME minimization problem without this constraint will still lead to a solution which satisfies the constraint. To test the conjecture that the constraint is non-binding, we remove it explicitly from the problem, solve the constrained l_1 LIME minimization problem without this constraint, and then it is seen that indeed the constraint held.

To show the existence and closed form of the solution w^* given in the theorem, we review the second order sufficiency conditions for a constrained minimizer [84]. As applied to the constrained l_1 LIME minimization Problem 2, the necessary and sufficient optimality conditions are as follows (a more general review of the necessary and sufficient optimization conditions appears in the Appendix):

C1) For the first order conditions there must exist a w^* such that the problem's constraints hold, $\sum_{j=1}^{k} w_j^*(x_j[m] - x[m]) = 0$ for $m = 1, \ldots, d$ and $\sum_j w_j^* - 1 = 0$, and additionally there must exist a $\kappa \in \mathcal{R}^d$ and $\mu \in \mathcal{R}$ such that

$$\left[\nabla \sum_{j=1}^{k} w_j \log w_j + \sum_{m=1}^{d} \kappa[m] \nabla \left(\sum_{j=1}^{k} w_j (x_j[m] - x[m]) \right) + \mu \nabla (\sum_{i=1}^{k} w_i - 1) \right]_{w=w^*} = \mathbf{0} \tag{3.15}$$

where $\nabla z(w)$ denotes the gradient of $z(w)$ (see the Appendix for definition of gradient and Hessian).

C2) Given the first-order condition C1 (which specifies a κ and μ), the second-order condition for optimality is that the matrix

$$\left[\nabla^2 \sum_{j=1}^{k} w_j \log w_j + \nabla^2 \sum_{m=1}^{d} \kappa[m] \left(\sum_{j=1}^{k} w_j (x_j[m] - x[m]) \right) + \nabla^2 \mu \left(\sum_{i=1}^{k} w_i - 1 \right) \right]_{w=w^*} \tag{3.16}$$

must be strictly positive definite.

First consider the second-order optimality condition C2. Since $-H(w)$ is strictly convex, and the other terms in the above expression (3.16) are linear, the expression will be strictly positive definite. Thus the second order optimality condition C2 will be satisfied if condition C1 is satisfied.

One can solve the first order optimality condition C1 specified in (3.15) for a possible solution w^*,

$$w_j^*(\kappa, \mu) = e^{\mu - 1 + \sum_{m=1}^{d} \kappa[m](x_j[m] - x[m])} \tag{3.17}$$

for $j = 1, \ldots, k$.

To show that the possible solution w^* is in fact a minimizer, one must show that there exist Lagrange multipliers κ and μ such that the constraints of the constrained l_1 LIME minimization problem are satisfied. Solving the constraint $\sum w_j = 1$ for μ results in

$$\mu = 1 - \log\left(\sum_{i=1}^{k} e^{\sum_{p=1}^{d} \kappa[p](x_i[p]-x[p])}\right). \tag{3.18}$$

Substituting for the value of μ in (3.17), the possible solution $w_j^*(\kappa)$ becomes

$$w_j^*(\kappa) = \frac{e^{\sum_{m=1}^{d} -\kappa[m](x_j[m]-x[m])}}{\sum_{i=1}^{k} e^{\sum_{p=1}^{d} \kappa[p](x_i[p]-x[p])}} \tag{3.19}$$

for $j = 1, \ldots, k$.

Lastly, κ must satisfy the set of constraints, $\sum_{j=1}^{k} w_j(x_j[m] - x[m]) = 0$ for $m = 1, \ldots, d$. Substituting in the potential solution (3.19) for w^*, κ must satisfy

$$\sum_{j=1}^{k}(x_j[m] - x[m])\left(\frac{e^{\sum_{m=1}^{d} -\kappa[m](x_j[m]-x[m])}}{\sum_{i=1}^{k} e^{\sum_{p=1}^{d} -\kappa[p](x_i[p]-x[p])}}\right) = 0$$

for $m = 1, \ldots, d$.

If no κ satisfies the above equation then, by the second order necessary conditions [84], no solution w^* exists for the d-dimensional constrained l_1 LIME minimization problem. If κ satisfies the constraints then the solution w^* exists as stated in (3.19). q.e.d.

The last problem definition needed is the semi-constrained l_1 LIME minimization problem, which is the l_1 LIME minimization problem with some dimensions constrained to have zero error of reproduction, $(\sum_j w_j x_j)[m] = x[m]$ for some m. Lemma 5 specifies the conditions under which a minimizer exists for the semi-constrained l_1 LIME minimization problem and gives a closed form solution for any such minimizer. The semi-constrained l_1 LIME minimization problem (and associated Lemma 5) will be useful in proving Theorem 5) for the multidimensional l_1 LIME minimization problem (problem 1).

Problem 3 (Semi-constrained l_1 LIME minimization problem) *Given $x \in \mathcal{R}^d$, and $x_j \in \mathcal{R}^d, j = 1, \ldots, k$, and $\lambda > 0$, and a division of the dimensions into exhaustive and mutually exclusive sets $\mathcal{M}$ and $\mathcal{P}$, the semi-constrained l_1 LIME minimization problem is to minimize the functional,*

$$F(w) = \left(\sum_{m \in \mathcal{M}} \sum_{j=1}^{k} w_j x_j[m] - x[m] \right) + \lambda \sum_{j=1}^{k} w_j \log w_j \tag{3.20}$$

over all pmfs w *(such that* $\sum_{j=1}^{k} w_j = 1$ *and* $w_j \geq 0$ *for all* j*) subject to the set of constraints,* $\sum_{j=1}^{k} w_j x_j[p] - x[p] = 0$ *for all* $p \in \mathcal{P}$*, and* $\sum_{j=1}^{k} w_j x_j[m] - x[m] > 0$ *for all* $m \in \mathcal{M}$ *.*

Some observations about the semi-constrained l_1 LIME minimization problem defined above will be useful. First, note that a minimizer for the problem might not exist because there may be no feasible vectors w, that is, there may be no w that satisfies the constraints.

The next lemma, Lemma 5, specifies the conditions under which a minimizer exists for the semi-constrained l_1 LIME minimization problem and gives a closed form solution for any such minimizer.

Lemma 5 *Given* $x \in \mathcal{R}^d$ *and* $x_j \in \mathcal{R}^d, j = 1, \ldots, k$*, and* $\lambda > 0$*, and a division of the dimensions into exhaustive and mutually exclusive sets* $\mathcal{M}$ *and* $\mathcal{P}$*, then a necessary and sufficient condition for a solution to exist for the semi-constrained* l_1 *LIME minimization problem, is that there exist a vector* $\kappa \in \mathcal{R}^d$ *which solves*

$$\sum_{j=1}^{k} (x_j[p] - x[p]) \left(e^{\sum_{r \in \mathcal{M}} -(x_j[r] - x[r])/\lambda + \sum_{s \in \mathcal{P}} -\kappa[s](x_j[s] - x[s])/\lambda} \right) = 0,$$

for $p \in \mathcal{P}$*, in which case the minimizing pdf is*

$$w_j^*(\kappa) = \frac{\left(e^{\sum_{r \in \mathcal{M}} -S[r](x_j[r] - x[r])/\lambda + \sum_{s \in \mathcal{P}} -S[s]\kappa[s](x_j[s] - x[s])/\lambda} \right)}{\sum_{i=1}^{k} e^{\sum_{u \in \mathcal{M}} -S[u](x_i[u] - x[u])/\lambda + \sum_{v \in \mathcal{P}} -S[v]\kappa[v](x_i[v] - x[v])/\lambda}}$$

for $j = 1, \ldots, k$*, where* $S \in \{-1, 1\}^d$*.*

Proof: In this proof, we initially conjecture that the non-negativity constraint $w_j \geq 0$ for all j is non-binding (also known as inactive), which means that solving the constrained l_1 LIME minimization problem without this constraint will still lead to a solution which satisfies the constraint. To test the conjecture that the constraint is

non-binding, we remove it explicitly from the problem, solve the constrained l_1 LIME minimization problem without this constraint, and then it is seen that indeed the constraint held.

Assuming that a minimizing pdf w^* exists, the vector S affects the orientation of the positive and negative axis for each dimension. Without loss of generality, let the correct choice of $S[m] = 1$ for all $m \in \mathcal{M}$ and note that the correct choice of S yields $\sum_j w_j^* x[m] - x[m] > 0$ for all $m \in \mathcal{M}$. Then the set of constraints, $\sum_{j=1}^k w_j x_j[m] - x[m] > 0$ for all $m \in \mathcal{M}$, holds.

Note the objective function is a linear combination of linear functions and a convex function, and thus the objective function is twice continuously differentiable in w. Further, the constraint functions, $\sum_{j=1}^k w_j x_j[p] - x[p] = 0$ and $\sum_j w_j - 1$, are twice continuously differentiable in w.

To show the existence and closed form of the solution w^* given in the theorem, the second order sufficiency conditions for a constrained minimizer are applied [84] (see Lemma 4 for a more detailed application, and a more general review of the necessary and sufficient optimization conditions appears in the Appendix).

First, consider the second-order optimality condition. Since the objective function is strictly convex, the second-order optimality condition will be satisfied (assuming the first order optimality condition is satisfied).

One can solve the first order optimality condition for a possible solution w^*,

$$w_j^*(\kappa, \mu) = e^{\mu - \lambda + \sum_{m \in \mathcal{M}} (x_j[m] - x[m])/\lambda + \sum_{p \in \mathcal{P}} \kappa[p](x_j[p] - x[p])/\lambda} \tag{3.21}$$

for $j = 1, \ldots, k$.

To show that the possible solution w^* is in fact a minimizer, one must show that there exist Lagrange multipliers κ and μ such that the constraints of the constrained l_1 LIME minimization problem are satisfied. The μ variable ensures that $\sum_j w_j = 1$, accounting for this constraint, the possible solution $w_j^*(\kappa)$ becomes

$$w_j^*(\kappa) = \frac{e^{\sum_{m \in \mathcal{M}} (x_j[m] - x[m])/\lambda + \sum_{p \in \mathcal{P}} \kappa[p](x_j[p] - x[p])/\lambda}}{\sum_i e^{\sum_{r \in \mathcal{M}} (x_i[r] - x[r])/\lambda + \sum_{s \in \mathcal{P}} \kappa[s](x_i[s] - x[s])/\lambda}} \tag{3.22}$$

for $j = 1, \ldots, k$.

Lastly, κ must satisfy the set of constraints, $\sum_{j=1}^{k} w_j(x_j[p] - x[p]) = 0$ for $p \in \mathcal{P}$. Substituting in the potential solution (3.22) for w^*, κ must satisfy

$$\sum_{j=1}^{k}(x_j[p] - x[p]) \left(e^{\sum_{r\in\mathcal{M}} -(x_j[r]-x[r])/\lambda + \sum_{s\in\mathcal{P}} -\kappa[s](x_j[s]-x[s])/\lambda}\right) = 0,$$

for $p \in \mathcal{P}$.

If no κ satisfies the above equation then, by the second order necessary conditions [84], no solution w^* exists for the d-dimensional constrained l_1 LIME minimization problem. If κ satisfies the constraints then the solution w^* exists as stated in (3.22). In the statement of the lemma, the vector S captures the uncertainty of the orientation of the negative and positive axis in each dimensions and the correct choice of S will ensure that all the constraints hold.

q.e.d.

Next, we review the exact penalty function theorem, further discussion of which can be found in books by Nash and Sofer [92], pages 549-551, Luenberger [84], pages 387-391, and Conn et al. [23], pages 600-612. The theorem is useful because it says that for heavy weightings of an l_1 norm penalty, the minimizer of the l_1 penalty problem is the same as the minimizer for the corresponding constrained optimization problem.

Theorem 3 (Exact Penalty Function Theorem) *Consider the following general constrained optimization problem: minimize $f(w)$ such that $g_i(w) = 0$ for $i = 1, 2, \ldots, t$ and $w \in \mathcal{R}^k$. The objective function f and the constraint functions $g_i(w)$ are assumed to be twice continuously differentiable. Let π be the l_1 norm penalty function given by $\pi(w, \lambda) = \lambda f(w) + \|g(w)\|_1$.*

If w^* satisfies the second order sufficiency conditions for a local minimum of the constrained optimization problem, that is, for all vectors v such that $\nabla g(w^*)^T v = 0$, $\nabla f(w^*)^T v = 0$ and $v^T \nabla^2 f(w^*) v > 0$. If Λ is the reciprocal of the largest Lagrange multiplier in absolute value for the constraints g, and if $\lambda \leq \Lambda$, then w^* is also a minimizer of $\pi(w, \lambda)$.

We will also need the following corollary to the exact penalty function theorem. The corollary states that the unconstrained problem's minimizer will be different from the constrained problem's minimizer. Since the constrained LIME problem has a unique minimizer that yields the maximum entropy given that the distortion is equal to zero, the unconstrained LIME problem cannot have a different minimizer and distortion equal to zero. Then, for $\lambda > \Lambda$, the minimal distortion of the unconstrained LIME problem will be larger than zero. The corollary presented here is Theorem 14.5.2 from Conn et al. [23] (page 613) with different notation and assumptions made explicit.

Corollary 2 (Conn et al.) *Suppose that (w^*, κ^*) is a minimizer pair (minimizer and Lagrange multiplier) of the constrained minimization problem: minimize $f(w)$ such that $g(w) = \mathbf{0}$. Let Λ be the reciprocal of $\|\kappa^*\|_\infty$. Suppose also that f and g are twice continuously differentiable, and that the rows of the Jacobian matrix of $g(w^*)$ are linearly independent. Let π be the l_1 norm penalty function given by $\pi(w, \lambda) = \lambda f(w) + \|g(w)\|_1$. Suppose that*

$$\lambda > \Lambda.$$

Then w^* is not a local minimizer of π.

Finally, all the building blocks are in place to present Theorem 4 and its proof. The theorem shows that, like Campbell's result, there is a closed form solution for the l_1 LIME minimization problem for one-dimensional feature spaces. Theorem 4 adds to Campbell's result by removing the assumption that $\min_j x_j \leq x \leq \max_j x_j$.

Theorem 4 *Given $x, x_j \in \mathcal{R}, j = 1, \ldots, k$, and $\lambda > 0$, define Λ as the solution to (if a solution exists)*

$$\sum_{j=1}^{k} \frac{x_j e^{-(x_j - x)/\Lambda}}{\sum_{i=1}^{k} e^{-(x_i - x)/\Lambda}} = x$$

and 0 otherwise.

Then the l_1 LIME minimization problem for $\lambda > 0$ is solved by the pmf $w(\lambda) = \{w_1(\lambda), w_2(\lambda), \ldots, w_k(\lambda)\}$, given by,

$$w_j(\lambda) = \begin{cases} \frac{e^{-s(x_j - x)/\lambda}}{\sum_{i=1}^{k} e^{-s(x_i - x)/\lambda}} & \text{if } \lambda \geq \Lambda \\ \frac{e^{-s(x_j - x)/\Lambda}}{\sum_{i=1}^{k} e^{-s(x_i - x)/\Lambda}} & \text{otherwise} \end{cases}$$

for $j = 1, \ldots, k$, where

$$s = \begin{cases} +1 & \text{if } \sum_{j=1}^{k} x_j/k \geq x, \\ -1 & \text{otherwise.} \end{cases}$$

Proof: First, note that if $\sum_{j=1}^{k} x_j/k < x$ then $s = -1$ and the x and x_j have their sign flipped, so that it suffices to prove the theorem with the assumption that $\sum_{j=1}^{k} x_j/k > x$ and $s = 1$.

The proof is divided into two cases. The first case is that $\lambda \leq \Lambda$. The exact penalty function theorem(Theorem 3) states that the minimizer of the l_1 LIME minimization problem is the same as the minimizer of the *c*onstrained l_1 LIME minimization problem if $\lambda \leq \Lambda$, where Λ is the reciprocal of the Lagrange multiplier of the constrained problem. From Lemma 4, it is seen that if the one-dimensional constrained l_1 LIME minimization problem has a solution, then the Lagrange multiplier κ exists and solves,

$$\sum_{j=1}^{k} \frac{x_j e^{-\kappa(x_j - x)}}{\sum_{i=1}^{k} e^{-\kappa(x_i - x)}} = x.$$

However, if the constrained l_1 LIME minimization problem has no solution, then there is no solution to the above equation. Thus in the theorem, Λ is defined as $1/\kappa$ if the above equation has a solution, and 0 if not. Since by hypothesis $\lambda > 0$, the theorem never equates the minimizer of the l_1 LIME minimization problem to the minimizer of the *constrained* l_1 LIME minimization problem unless the constrained problem is solvable.

If the l_1 LIME minimization problem satisfies the exact penalty theorem conditions then it has the same minimizer w^* as the constrained l_1 LIME minimization problem (see Lemma 4). Substituting $\Lambda = 1/\kappa$ into (4), and noting that this is the solution for any $\lambda \leq \Lambda$, we conclude that for $\lambda \leq \Lambda$,

$$w_j^*(\lambda) = \frac{e^{-(x_j - x)/\Lambda}}{\sum_{i=1}^k e^{-(x_i - x)/\Lambda}} \tag{3.23}$$

for $j = 1, \ldots, k$.

The remaining case is that $\lambda > \Lambda$. Recall that $\sum_{j=1}^k x_j/k > x$ by assumption. Then from Lemma 3, it is known that the minimizing weights w^* satisfy $\sum_{j=1}^k w_j^* x_j \geq x$.

Further, it is known that $\sum_{j=1}^k w_j^* x_j > x$ because by Corollary 2, for $\lambda > \Lambda$, $D(w) \neq 0$, so that $\sum_{j=1}^k w_j^* x_j - x \neq 0$. (Note, to apply Corollary 2 the rows of the Jacobian of $D(w)$ must be linearly independent. For a one-dimensional feature space this is trivially satisfied.)

Thus there exists a minimizer w^* of the l_1 LIME objective function $F(w) = \| \sum_{j=1}^k w_j x_j - x \| + \lambda \sum_{j=1}^k w_j \log w_j$ such that $\sum_{j=1}^k w_j^* x_j > x$.

Consider a new objective function $F'(w)$ that is the same as $F(w)$ but without absolute value signs:

$$F'(w) = \sum_{j=1}^k w_j x_j - x + \lambda \sum_{j=1}^k w_j \log w_j.$$

Note that w^* is a minimizer to $F(w)$ such that $\sum_{j=1}^k w_j^* x_j > x$. Local to w^*, $F(w)$ and $F'(w)$ are equivalent. So w^* must also be a minimizer of $F'(w)$.

Since $F'(w)$ is strictly convex it has a unique minimizer. Therefore, the unique minimizer of $F'(w)$ must be w^*.

Because $F'(w)$ is differentiable everywhere, one can solve analytically for the minimizer of $F'(w)$.

Since the objective function $F'(w)$ is the positively weighted sum of a linear function(convex) and a strictly convex function, the matrix of second derivatives is positive definite, and the second order optimality condition will be satisfied. Next, we consider the first order optimality condition. Differentiating the objective function $F'(w)$ with respect to w, setting it to zero, and solving for the Lagrange multiplier (for the constraint $\sum_j w_j - 1 = 0$) leads to the solution $w^* = \{w_1, w_2, \ldots, w_k\}$ where for each $j = 1$ to k,

$$w_j^*(\lambda) = \frac{e^{-(x_j - x)/\lambda}}{\sum_{i=1}^{k} e^{-(x_i - x)/\lambda}}. \tag{3.24}$$

Thus we have shown that the Lagrangian multiplier Λ from the constrained problem divides the unconstrained problem into two cases. In the first case, $\lambda \leq \Lambda$ and the l_1 distortion for the unconstrained case is zero. By the exact penalty function theorem the minimizer is then equivalent to the constrained problem's minimizer, as shown in (3.23). On the other hand, for $\lambda > \Lambda$ the unconstrained problem can be re-written and solved analytically, yielding the solution shown in (3.24). q.e.d.

3.2.3 Exponential weights for l_1 distortion and multi-dimensional feature space

It would be delightful if one could simply generalize the closed form solution for l_1 distortion from one dimension (Theorem 4) to many dimensions. However, it is not clear that a simple generalization exists. At the least, one can prove that for a d-dimensional feature space, the optimal distribution will be one of 2^d specifiable possible solutions.

The problem is that in the multi-dimensional feature space case, it is not known in which quadrant the best reproduction $\sum_j w_j^* x_j$ lies. For the one dimensional case $x \in \mathcal{R}$, we showed in Lemma 3 that if $\sum_{j=1}^{k} x_j/k > x$, then the minimizer w^* satisfies $\sum_{j=1}^{k} w_j^* x_j \geq x$. However, in the multiple dimensional case with l_1 distortion, the corresponding assertion does not hold. Specifically, it is not true that if $\sum_{m=1}^{d} \sum_{j=1}^{k} x_j[m]/k > \sum_{m=1}^{d} \sum_{j=1}^{k} x[m]$, then the optimal weights satisfy $\sum_{m=1}^{d} \sum_{j=1}^{k} w_j^* x_j[m] > \sum_{m=1}^{d} \sum_{j=1}^{k} x[m]$. Counterexamples are easy to construct.

Without knowledge of which quadrant contains the optimal estimate, one can determine the closed form of the solution only to within a set of 2^d specifiable possibilities.

The multi-dimensional unconstrained l_1 LIME minimization problem has a solution that can be stated in closed form, but there are a number of cases. The following

procedure separates the cases. In the procedure, first the constrained l_1 LIME minimization problem is solved for its Lagrange multiplier Λ. Then, if $\lambda \leq \min_m \|\Lambda[m]\|$, the solution is known; by Theorem 3 the unconstrained solution is the constrained problem's solution. If $\lambda > \max_m \|\Lambda[m]\|$, then no constraints hold, and the solution can be analytically solved for. If $\min_m \|\Lambda[m]\| < \lambda < \max_m \|\Lambda[m]\|$, then some of the constraints might hold. To determine which constraints actually hold, a new semi-constrained l_1 LIME minimization problem is set-up and solved, resulting in a new Lagrange multiplier. Then, the logic repeats itself: how does λ compare to the new Lagrange multiplier? That will determine how many constraints actually do hold for the unconstrained solution, and that determines the analytic solution.

Procedure for solving the l_1 LIME minimization problem

Given $x \in \mathcal{R}^d$, a set of points $x_j \in \mathcal{R}^d, j = 1, \ldots, k$ which include a basis of the d-dimensional space, and $\lambda > 0$, follow the steps until a solution w^* for the l_1 LIME minimization problem is given.

Step 1) Define $\Lambda \in \mathcal{R}^d$ as the solution to (if a solution exists) the system of d equations,

$$\sum_{j=1}^{k} (x_j[m] - x[m]) \left(\frac{e^{\sum_{r=1}^{d} -(x_j[r]-x[r])/\Lambda[r]}}{\sum_{i=1}^{k} e^{\sum_{p=1}^{d} -(x_i[p]-x[p])/\Lambda[p]}} \right) = 0, \tag{3.25}$$

for $m = 1, \ldots, d$, and $\Lambda = 0$ otherwise.

Step 2) If $\lambda > \max_c \|\Lambda[c]\|$ for $c = 1, \ldots, d$, then the minimizer is

$$w_j^*(\lambda) = \frac{e^{\sum_{r=1}^{d} -S[r](x_j[r]-x[r])/\lambda}}{\sum_{i=1}^{k} e^{\sum_{p=1}^{d} -S[p](x_i[p]-x[p])/\lambda}}$$

for $j = 1, \ldots, k$, where $S \in \{-1, 1\}^d$.

Step 3) If $\lambda \leq \min_c \|\Lambda[c]\|$ for $c = 1, \ldots, d$, then the minimizer is

$$w_j^*(\lambda) = \frac{e^{\sum_{r=1}^{d} -(x_j[r]-x[r])/\Lambda[r]}}{\sum_{i=1}^{k} e^{\sum_{p=1}^{d} -(x_i[p]-x[p])/\Lambda[p]}}$$

for $j = 1, \ldots, k$.

Step 4) Let m be the smallest integer such that $\lambda > \|\Lambda[m]\|$.

Step 5) Let $\Lambda[c] = 0$ for $c \leq m$ and for $c = m+1, \ldots, d$, let $\Lambda[c]$ be the solution to the system of $d - m$ equations,

$$\sum_{j=1}^{k} (x_j[p] - x[p]) \left(e^{\sum_{r=1}^{m} -(x_j[r]-x[r])/\lambda + \sum_{s=m+1}^{d} -(x_j[s]-x[s])/\Lambda[s]} \right) = 0,$$

where $p = m+1, \ldots, d$.

Step 6) If $\lambda \geq \max_c \|\Lambda[c]\|$ for $c = m+1, \ldots, d$, then the minimizer is

$$w_j^*(\lambda) = \frac{e^{\sum_{r=1}^{d} -S[r](x_j[r]-x[r])/\lambda}}{\sum_{i=1}^{k} e^{\sum_{p=1}^{d} -S[p](x_i[p]-x[p])/\lambda}},$$

for $j = 1, \ldots, k$, where $S \in \{-1, 1\}^d$.

Step 7) If $\lambda \leq min_c \|\Lambda[c]\|$ for $c = m+1, \ldots, d$, then the minimizer is

$$w_j^*(\lambda) = \frac{\left(e^{\sum_{r=1}^{m} -S[r](x_j[r]-x[r])/\lambda + \sum_{s=m+1}^{d} -S[s](x_j[s]-x[s])/\Lambda[s]} \right)}{\sum_{i=1}^{k} e^{\sum_{u=1}^{m} -S[u](x_i[u]-x[u])/\lambda + \sum_{v+m+1}^{d} -S[v](x_i[v]-x[v])/\Lambda[v]}}$$

for $j = 1, \ldots, k$, where $S \in \{-1, 1\}^d$.

Step 8) Let c be the smallest integer such that $\lambda > \|\Lambda[c]\|$ for $c = m+1, \ldots, d$. Let $m = c$. Go to Step 5.

Note that every time Step 4 is reached a less-constrained problem is evaluated. This has two consequences. First, a solution must always exist for the less-constrained problem if a solution existed for a more constrained problem (such as that solved in Step 1). Second, for finite dimensions d, the procedure must end.

Theorem 5 *Given $x \in \mathcal{R}^d$, a set of points $x_j \in \mathcal{R}^d, j = 1, \ldots, k$ which include a basis of the d-dimensional space, and $\lambda > 0$, the* Procedure for solving the l_1 LIME minimization problem *will result in the correct analytical solution for the minimizing weight distribution.*

Proof: The vector S affects the orientation of the positive and negative axis for each dimension. Without loss of generality, let the correct choice of $S[m] = 1$ for all $m =$

$1, \ldots, d$ and let the definition of the correct choice of S be that $\sum_j w_j^* x[m] > x[m] \geq 0$ for all $m = 1, \ldots, d$, where w^* is the minimizing pdf.

Consider first the case of Step 2, such that $\lambda > \max_c \|\Lambda[c]\|$. Recall that the vector S was selected so that

$$\sum_j w_j^* x_j[p] - x[p] \geq 0$$

for all $p = 1, \ldots, d$.

For the constraint $\sum_j w_j^* x_j[m] - x[m] = 0$ to hold, it is necessary that (by first order optimality conditions) that $\lambda \leq \|\Lambda[c]\|$. Combining $\lambda > \|\Lambda[c]\|$ and (3.2.3) leads to,

$$\sum_j w_j^* x_j[p] - x[p] > 0.$$

for all $p = 1, \ldots, d$.

Since, $\lambda > \|\Lambda[p]\|$ for all $p = 1, \ldots, d$, there exists a minimizer w^* of the l_1 LIME objective function $F(w) = \| \sum_{j=1}^k w_j x_j - x\|_1 + \lambda \sum_{j=1}^k w_j \log w_j$ such that $\sum_{m=1}^d \sum_j w_j^* x_j[m] - x[m] > 0$.

Consider a new objective function $F'(w)$ that is the same as $F(w)$ except without absolute value signs:

$$F'(w) = \sum_{m=1}^{d} \sum_{j=1}^{k} w_j x_j[m] - x[m] + \lambda \sum_{j=1}^{k} w_j \log w_j. \tag{3.26}$$

As shown above, w^* is a minimizer to $F(w)$ such that $\sum_{m=1}^d \sum_j w_j^* x_j[m] - x[m] > 0$. Then, locally around w^*, $F(w)$ and $F'(w)$ are equivalent. So w^* must also be a minimizer of $F'(w)$.

Since $F'(w)$ is strictly convex it has a unique minimizer. Therefore, the unique minimizer of $F'(w)$ must be w^*. Because $F'(w)$ is differentiable everywhere, we can analytically solve for the minimizer of $F'(w)$.

Since the objective function $F'(w)$ is the positively weighted sum of linear functions and a strictly convex function, the matrix of second derivatives is positive definite, and the second order optimality condition will be satisfied. Next, we consider

the first order optimality condition. Differentiating the objective function $F'(w)$ with respect to w, setting it to zero, and solving for the Lagrange multiplier (for the constraint $\sum_j w_j - 1 = 0$) leads to the solution $w^* = \{w_1, w_2, \ldots, w_k\}$ where for $j = 1, \ldots, k$,

$$w_j^*(\lambda) = \frac{e^{\sum_{r=1}^{d} -(x_j[r]-x[r])/\lambda}}{\sum_{i=1}^{k} e^{\sum_{p=1}^{d} -(x_i[p]-x[p])/\lambda}}. \tag{3.27}$$

Next consider the case of Step 3 such that $\lambda \leq \min_c \|\Lambda[c]\|$, for $c = 1, \ldots, d$. Then the exact penalty function theorem (Theorem 3) can be applied. The *constrained* l_1 LIME minimization problem minimizes $-H(w)$ subject to the constraint functions $\sum_{j=1}^{k} w_j x_j[m] - x[m] = 0$ for $m = 1, \ldots, d$. The exact penalty function theorem can be applied to the problem at hand to conclude that the minimizer of the l_1 LIME problem is the same as the minimizer of the constrained l_1 LIME problem if $\lambda \leq \min_m \|\Lambda[m]\|$. The minimizer of the constrained l_1 LIME problem was shown in Lemma 4.

If the conditions in Step 2 and Step 3 did not hold, then it must be that

$$\min_m \|\Lambda[m]\| \leq \lambda \leq \max_p \|\Lambda[p]\|$$

for some m and some p. In this case some of the constraints might hold, that is, it could be that

$$\sum_j w_j^* x_j[p] - x[p] = 0,$$

for some $p \in \mathcal{P}$.

In Step 4, variable m is assigned to be the smallest integer such that $\lambda > \|\Lambda[m]\|$. To determine if (3.2.3) holds for any $p \geq m + 1$, solve the semi-constrained l_1 LIME minimization problem with constraints on the dimensions $p \geq m + 1$. Since the constrained l_1 LIME minimization problem had a solution (or else one would not have arrived at Step 4), the semi-constrained l_1 LIME minimization problem will also have a solution, with an associated reciprocal Lagrange multiplier Λ', where $\Lambda'[p] = \lambda/\kappa[p]$ if $p \in \mathcal{P}$ and 0 otherwise, and κ solves (3.2.2). In Step 5, Λ is re-defined to be zero for dimensions $p = 1, \ldots, m$, and $\Lambda[p] = \Lambda'[p]$ otherwise.

Now, one again faces three cases. If $\lambda[p] \leq \min_p \|\Lambda'[p]\|$ for all $p > m$, then the EPFT applies and the l_1 LIME minimization problem is solved by the same pdf as the semi-constrained l_1 LIME minimization problem. If $\lambda[p] > \max_p \|\Lambda'[p]\|$ for all $p > m$, then no constraints hold for the minimizing solution, and instead (3.2.3) holds for all dimensions, and the l_1 LIME minimization problem can be solved analytically as described above in the proof for Step 2. Lastly, if $\|\Lambda'[r]\| \leq \lambda \leq \|\Lambda'[s]\|$ for some $r, s > m, s > r$, the validity of (3.2.3) is not yet known. In this situation, a new semi-constrained l_1 LIME minimization problem must be formulated with the constraints on the dimensions s for all s such that $\lambda \leq \|\Lambda'[s]\|$. This new semi-constrained problem should be solved, the resultant new Lagrange multiplier is compared to λ, and once again, three cases are faced.

Thus we have shown that the Lagrangian multiplier Λ from the constrained l_1 LIME minimization problem divides the unconstrained problem into three cases. In the first case, $\lambda \leq \min_m \|\Lambda[m]\|$ and the unconstrained problem's l_1 distortion is zero. By the exact penalty function theorem the minimizer is then equivalent to the constrained problem's minimizer. In the second case, $\lambda > \max_m \|\Lambda[m]\|$, and the unconstrained problem can be re-written and solved analytically, yielding the solution shown in (3.27). In the third case, $\min_m \|\Lambda[m]\| \leq \lambda \leq max_p \|\Lambda[p]\|$, and a semi-constrained l_1 LIME minimization problem must be solved recursively until it can be determined which constraints hold for the unconstrained minimizer. Once it is known which constraints hold for the unconstrained minimizing solution, the analytic form for the minimizer is known (from logic presented earlier in the proof).

In the end, one may be left with uncertainty as to the true vector S (recall it was assumed that the correct vector S was all ones at the beginning of the proof). There are d components of S and each component can take on one of two values, -1 or 1, leaving 2^d possibilities to test for the final solution.

Notes on implementing with the l_1 distortion

All of the numerical results for LIME were obtained using an l_2 distortion implementation of LIME, discussed in Section 2.11.

One could implement the l_1 distortion LIME algorithm as follows:

Step 1) For a test point $X \in \mathcal{R}^d$, determine its k nearest neighbors which include a basis of the d-dimensional space.

Step 2) Solve (3.25) for Λ.

Step 3) Follow the *Procedure for solving the l_1 LIME minimization problem.*

Step 4) If the solution from Step 3 has an unknown S vector, compute $F(w)$ for each of the 2^d possible S solutions from Step 3. The minimum $F(w)$ determines the optimal solution, w^*.

Step 5) Apply the LIME weights w^* to estimate the class or value of interest, $f(X)$,

$$f_{LIME}(x) = \arg\min_{\hat{y}} \sum_{j=1}^{k} w_j^*(X) C(\hat{y}, Y_j(X)).$$

The most troublesome implementation issue is Step 2, which requires solving (3.25) for Λ. This problem should be amenable to convex optimization techniques.

3.3 Consistency

Many nonparametric supervised learning classification rules achieve the Bayes' risk in the limit of increasing training samples, see [30] for a review of consistency properties of a number of classes of algorithms. Guaranteeing that an algorithm will asymptotically achieve low risk validates the inductive method in some philosophers' eyes (see Section 1.0.1). Many parametric algorithms are not guaranteed to converge to the Bayes' risk. Algorithms that fit a model of the class densities, or constrain the decision boundaries, may prove too rigid to adapt to the true decision boundary even in the presence of infinite data.

In 1967, Cover and Hart [24] showed that for k-NN with $k = 1$ and the total number of training samples $n \to \infty$, the asymptotic error is no worse than twice the Bayes' risk. With a one neighbor neighborhood, LIME performs exactly the same as

k-NN (all the weight goes to the one neighbor) and thus the same result holds.

In this section it is shown that the LIME algorithm with l_1 distortion converges to the Bayes' risk under standard assumptions of double asymptopia ($k \to \infty$, $n \to \infty$, and $k/n \to 0$).

In 1977 Stone proved [118] that for nonparametric algorithms that assign weights to neighboring training samples and then apply those weights linearly in the output domain ($\hat{Y} = \sum_j w_j(X)Y_j$), there are five conditions that must be satisfied for the algorithm to converge; only three conditions are needed if the weights form a probability mass function.

First we review some definitions and Stone's theorem, and then present the theorem and proof for LIME.

Given iid training samples $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)\}$ and (X, Y), and real-valued Y with $E|Y|^p < \infty$, then an estimator is consistent if $E[\hat{Y}|X] \to E[Y|X]$ in L_p as $n \to \infty$.

Let w be a sequence of pmfs. Let n be the cardinality of the training sample set, $k(n)$ the cardinality of the neighborhood. Let w_{nj} be the probability corresponding to the sample point X_j in the neighborhood.

Stone's [118] Corollary 1 is the following:

Theorem 6 (Stone) *A sequence of pmfs w is universally consistent if and only if the following three conditions hold:*

C1) There is a constant $C \geq 1$ such that, for every nonnegative Borel function f on $\mathcal{R}^d$, $E\left[\sum_j \|w_{nj}(X)\| f(X_j)\right] \leq CE[f(X)]$ for all $n \geq 1$

C2) $\lim_{n\to\infty} \sum_j \|w_{nj}(X)\| I_{\{\|X_j - X\| > a\}} \to 0$ in probability for all $a > 0$

C3) $\lim_{n\to\infty} \max_j \|w_{nj}(X)\| \to 0$ in probability.

Next, LIME with l_1 distortion is shown to be consistent. The proof for Stone's first condition relies on the l_1 distortion function assumption. We conjecture that the LIME weights will be consistent for l_2 distortion as well, however, efforts to prove this were thwarted by the first condition.

Theorem 7 (LIME consistency) *Suppose points $X \in \mathcal{R}^d$ and $X_i \in \mathcal{R}^d$, $i = 1, \ldots, n$ are iid. Let X_j, $j = 1, \ldots, k(n)$ be the jth nearest neighbor to X from the set*

X_i, $i = 1, \ldots, n$. Suppose w_n^* is the pmf that solves the l_1 LIME minimization problem for X and X_j, $j = 1, \ldots, k(n)$. Then the sequence of weights w^* is universally consistent as $k(n) \to \infty$, $n \to \infty$, and $k(n)/n \to 0$.

Proof: Stone's theorem (Theorem 6) is applied. The three conditions are proved separately in the next three subsections.

3.3.1 Proof of Stone's first condition

For the first condition of Stone's corollary, it must be shown that for all $n \geq 1$,

$$E\left[\sum_i |w_{ni}(X)| f(X_i)\right] \leq CE[f(X)] \tag{3.28}$$

For this proof, an upper bound on the LIME weights is shown, which will lead to an upper bound of the left-side of (3.28) for the LIME weights. Then Stone's Proposition 11 can be applied to the new weights to show that his first condition holds for the new weights. Since the new set upperbounds the LIME weights, the condition also holds for the LIME weights.

The LIME weights $w_{ni}(X)$ are upperbounded by a new set of weights $\tilde{w}_{nj}(X)$, $j = 1, \ldots, k(n)$:

$$\tilde{w}_{nj}(X) = \max_i \left(w_{ni}(X)\right) \tag{3.29}$$

Each new weight $\tilde{w}_{nj}(X)$ is the maximum of the set of original weights, and thus is not less than the corresponding weight $w_{nj}(X)$.

To show that the new set is normalizable, it will be shown that the new weights can be summed to a finite constant,

$$\begin{aligned} \sum_{j=1}^{k(n)} \tilde{w}_{nj}(X) &= \sum_{j=1}^{k(n)} \max_i \left(w_{ni}(X)\right) \\ &= k(n) \max_i \left(w_{ni}(X)\right) \end{aligned}$$

Substitute the known form of the weights from Theorem 5. Let

$$\lambda^* = \begin{cases} \lambda & \text{if } \lambda \geq \min_m \Lambda[m] \text{ for } m = 1, \ldots, d \\ \Lambda & \text{otherwise.} \end{cases}$$

Note that $\lambda^* > 0$ since $\lambda > 0$. Then,

$$\begin{aligned} \sum_{j=1}^{k(n)} \tilde{w}_{nj}(X) &= k(n) \max_q \left(\frac{e^{\sum_{r=1}^d -S[r](x_q[r]-x[r])/\lambda^*}}{\sum_{i=1}^{k(n)} e^{\sum_{p=1}^d -S[p](x_i[p]-x[p])/\lambda^*}} \right) \\ &\leq k(n) \frac{\max_q \left(e^{\sum_{r=1}^d -S[r](x_q[r]-x[r])/\lambda^*} \right)}{\min_i \sum_{i=1}^{k(n)} e^{\sum_{p=1}^d -S[p](x_i[p]-x[p])/\lambda^*}} \\ &= k(n) \frac{\max_q \left(e^{\sum_{r=1}^d -S[r](x_q[r]-x[r])/\lambda^*} \right)}{k(n) \min_i e^{\sum_{p=1}^d -S[p](x_i[p]-x[p])/\lambda^*}} \\ &= \frac{\max_q \left(e^{\sum_{r=1}^d -S[r](x_q[r]-x[r])/\lambda^*} \right)}{\min_i e^{\sum_{p=1}^d -S[p](x_i[p]-x[p])/\lambda^*}} \end{aligned}$$

Let Δ be the distance from the test point x to its furthest neighbor,

$$\Delta = \max_j |X - X_j|$$

Then the sum of the tilde weights $\sum_{j=1}^{k(n)} \tilde{w}_{nj}(X)$ can be expressed,

$$\begin{aligned} B &= \sum_{j=1}^{k(n)} \tilde{w}_{nj}(X) \\ &\leq \frac{e^{d\Delta/\lambda^*}}{e^{-d\Delta/\lambda^*}} \\ &= e^{2d\Delta/\lambda^*} \end{aligned}$$

Since the tilde weights are summable, they can be normalized to sum to one. Normalize the tilde weights to create a new set of weights

$$\tilde{w}'_n(X) = \frac{\tilde{w}_n(X)}{B}.$$

Now, Stone's condition 1 is shown to hold for the normalized set of weights $\tilde{W}'_{ni}$, and then it will be shown that if the condition holds for the normalized tilde weights, it must hold for the original LIME weights.

Proposition 11 from Stone's paper [118] will be useful.

Proposition 11: Let w_n be a pmf monotonically non-increasing function along all rays centered at X. If f is a nonegative Borel function on R^d such that $E[f(X)] < \infty$, then

$$E\left[\sum_i w_{ni}(X)f(X_i)\right] \leq \beta(d)E[f(X)],$$

where $\beta(d)$ is the minimum number of cones needed to cover the $\mathcal{R}^d$ space.

For finite d, $\beta(d)$ is a finite [118] constant.

The normalized tilde weights $\tilde{w}'$ satisfy Proposition 11's requirements of a probability weight function that is non-increasing in all directions. Applying Proposition 11 to the normalized tilde weights results in

$$E\left[\sum_{i=1}^{k(n)} \tilde{w}'_{ni}(X)f(X_i)\right] \leq \beta(d)E[f(X)].$$

Then for the unnormalized tilde weights $\tilde{w}$,

$$E\left[\sum_{i=1}^{k(n)} \tilde{w}_{ni}(X)f(X_i)\right] \leq \beta(d)BE[f(X)],$$

and for the original LIME weights,

$$E\left[\sum_{i=1}^{k(n)} w_{ni}(X)f(X_i)\right] \leq \beta(d)BE[f(X)]$$

Since $\beta(d)B$ is a constant, Stone's first condition holds.

3.3.2 Proof of Stone's second condition

The second condition of Stone's corollary requires that the sum of the weights outside a shrinking neighborhood goes to zero, $\sum_{j=1}^{k} \|w_{nj}(X)\| I_{\{\|X_j - X\| > a\}} \to 0$ as $n \to \infty$,

$k \to \infty$, and $k/n \to 0$, in probability for all $a > 0$.

Since all of the LIME weights correspond to points inside the neighborhood, this condition is equivalent to requiring the neighborhood to shrink to zero,

$$P(\|x_{k(n)}(x) - x\| > a) \to 0,$$

where $x_{k(n)}(x)$ is the kth nearest training point to x. Lemma 5.1 from Devore, Gyorfi, and Lugosi [30] applies here.

Lemma 6 (Devore, Gyorfi and Lugosi) *If X is independent of the training data, then $\|X_k(X) - X\| \to 0$ with probability one whenever $k/n \to 0$ as $n \to \infty$.*

The LIME neighborhood is defined in the theorem with $k/n \to 0$. Further, X and the training data X_j, $j = 1, \ldots, k$ are assumed to be iid. Thus Lemma 6 holds, and therefore Stone's second condition holds.

3.3.3 Proof of Stone's third condition

Stone's third condition requires that $\max_j \|w_{nj}(X)\| \to 0$ in probability. To show Stone's third condition, it is sufficient to show that for any $\epsilon > 0$, there is some n_0 such that for every $n > n_0$, $\max_j w_{nj} < \epsilon$.

The proof is by contradiction. For a sequence of pmfs, $\{a_n; n = 1, 2, \ldots\}$, let the set $\mathcal{A}$ be such that $\{a_n \in \mathcal{A}$ iff $\max_j a_{nj} \geq \epsilon\}$. For any $\epsilon > 0$ and any weight sequence a such that $a_n \in \mathcal{A}$ infinitely often, it is shown that there is some n_0 for which

$$F(w_n^u) < F(a_n) \tag{3.30}$$

for $n > n_0$, and $a_n \in \mathcal{A}$, where $w_{nj}^u = \frac{1}{k(n)}$ for all n and for all $j = 1, \ldots, k(n)$.

Since the weight sequence chosen by the LIME algorithm must minimize the functional $F(w) = D(w) - \lambda H(w)$ for all n, (3.30) entails that no pmf sequence with $\max_j a_{nj} \geq \epsilon$ infinitely often for any $\epsilon > 0$ achieves the minimum functional for all n. Therefore no pmf sequence with $\max_j a_{nj}$ infinitely often will be the LIME pmf sequence chosen. Then the LIME pmf sequence must satisfy Stone's third condition.

To show (3.30), begin with the relation

$$\begin{aligned} F(a_n) &= D(a_n) - \lambda H(a_n) \\ F(a_n) &\geq -\lambda H(a_n) \end{aligned}$$

Fano's inequality [25] applies to the entropy of a pmf with $\max_j w_j \geq \epsilon$,

$$F(a_n) \geq -\lambda \left(-\epsilon \log(\epsilon) - (1-\epsilon) \log \left(\frac{1-\epsilon}{k(n)-1} \right) \right)$$

for $a_n \in \mathcal{A}$.

To compare with $F(w_n^u)$, first note that

$$\begin{aligned} D(w_n^u) &= \| \frac{1}{k(n)} \sum_{j=1}^{k(n)} X_j - X \|_1 \\ &\leq \| \frac{1}{k(n)} \sum_{j=1}^{k(n)} \| X_j - X \| \|_1. \end{aligned}$$

The conditions for Lemma 6 (stated in Section 3.3.2) hold, and thus with probability one

$$\| X_{k(n)}(X) - X \| \to 0.$$

By definition of a jth nearest neighbor, $\| X_j(X) - X \| \leq \| X_{k(n)}(X) - X \|$ for all $j \leq k(n)$, and since the sum of the absolute value differences is divided by $k(n)$, we can conclude that as $n \to \infty$,

$$D(w_n^u) \to 0.$$

Comparing the entropy sequences for $n \in \mathcal{A}$,

$$\begin{aligned} H(w_n^u) - H(a_n) &= \log(k(n)) + \epsilon \log(\epsilon) + (1-\epsilon) \log \left(\frac{1-\epsilon}{k(n)-1} \right) \\ &= \log \left(\frac{k(n)}{(k(n)-1)^{1-\epsilon}} \right) + \log \left(\epsilon^{\epsilon} (1-\epsilon)^{1-\epsilon} \right) \\ &= \epsilon \log(k(n)) + (1-\epsilon) \log(\frac{k(n)}{k(n)-1}) + \log \left(\epsilon^{\epsilon} (1-\epsilon)^{1-\epsilon} \right). \end{aligned}$$

for $a_n \in \mathcal{A}$.

For a given ϵ, as $n \to \infty$, the above difference is dominated by the $\epsilon \log k(n)$ term and grows without bound as $n \to \infty$. Thus,

$$H(w_n^u) - H(a_n) \to \infty.$$

for $a_n \in \mathcal{A}$.

Since $\lambda > 0, \epsilon > 0$ are fixed, there exists some n_0 for which

$$D(w_n^u) < \lambda(H(w_n^u) - H(a_n))$$

for $n > n_0$, and $a_n \in \mathcal{A}$.

Consequently,

$$\begin{aligned} F(w_n^u) - F(a_n) &\leq D(w_n^u) - \lambda(H(w_n^u) - H(a_n)) \\ &< 0 \end{aligned}$$

for $n > n_0$ and $a_n \in \mathcal{A}$.

Then (3.30) has been shown, and no sequence of pmfs a with $\max_j a_{nj} \geq \epsilon$ infinitely often will be the weight sequence chosen by the LIME algorithm.

Therefore the LIME weight sequence must have $\max_j w_{nj} \to 0$ in probability, satisfying Stone's third condition.

In conclusion, the LIME algorithm chooses a weight sequence that satisfies Stone's three conditions and thus the LIME algorithm achieves consistency. q.e.d.

3.3.4 Other asymptotic properties

A number of other asymptotic properties may be shown for a learning algorithm. Stone shows [118] that many of these results are obtainable 'for free' once the weights are shown to be consistent. As an example, in this section convergence of the second moments is discussed.

Let f and g be Borel functions on R^m such that $E[f^2(Y)] < \infty$ and $E[g^2(Y)] < \infty$. Then the covariance of $g(Y)$ and $h(Y)$ given X is defined by

$$Cov(g(Y), h(Y)|X) = E[g(Y)h(Y)|X] - E[g(Y)|X]E[h(Y)|X],$$

and the conditional variance is defined,

$$Var(g(Y)|X) = Cov(g(Y), g(Y)|X).$$

Let the LIME estimate of the conditional covariance and variance be formed by using the LIME weights with details as in Theorem 7,

$$\widehat{Cov}(g(Y), h(Y)|X) = \hat{E}_{LIME}[g(Y)h(Y)|X] - \hat{E}_{LIME}[g(Y)|X]\hat{E}_{LIME}[h(Y)|X],$$

and

$$\widehat{Var}(g(Y)|X) = \widehat{Cov}(g(Y), g(Y)|X).$$

Stone showed [118] that the second order conditional moments converge in L^1 for any consistent set of weights and corresponding linear estimator that applies the weights to form estimators of the following form for Borel functions g defined on R^d with finite expectation,

$$\hat{E}_n[g(Y)|X] = \sum_j w_{nj}(X)g(Y_j).$$

Since it was shown in Section 3.3 that the LIME weights are consistent, and since the LIME estimators are linear estimators of the form specified, Stone's results apply and the LIME estimates of the second order conditional moments converge.

3.4 Robustness to noise

Real measurements of real data are rarely noise-free. In this section it is shown that if the training samples are corrupted by iid zero-mean additive noise, the expected LIME estimate will be unaffected, and further, that it will converge to the clean LIME estimate as the number of neighborhood training samples $k \to \infty$.

First it is shown that the LIME estimate is unbiased for training data features $\{X_1, X_2, \ldots, X_k\}$ with iid zero-mean additive noise, and then it is shown that the LIME estimate is also unbiased for training data observations $\{Y_1, Y_2, \ldots, Y_k\}$ infected with iid zero-mean additive noise.

A variation of the law of large numbers is shown for LIME weights. That result is key for the subsequent proof that iid zero-mean additive noise on either the training features or on the training observations will result in a noisy LIME estimate that asymptotically converges to the clean LIME estimate.

3.4.1 LIME expectation unaffected by noise on training observations

Let ϵ_j, for $j = 1, \ldots, k$, be iid zero-mean noise added to the neighborhood training observations $\{Y_1, Y_2, \ldots, Y_k\}$. Then the noisy dependent training variables are

$$\tilde{Y}_j = Y_j + \epsilon_j$$

for $j = 1, \ldots, k$. The clean estimate is $\hat{Y} = \sum_j w_j Y_j$. The noisy estimate is,

$$\hat{\tilde{Y}} = \sum_j w_j (Y_j + \epsilon_j).$$

Then the expectation of the noisy estimate is, where E_ϵ represents the expectation taken with respect to the noise random variables,

$$\begin{aligned} E_\epsilon[\hat{\tilde{Y}}] &= E_\epsilon\left[\sum_j w_j Y_j\right] + E_\epsilon\left[\sum_j w_j \epsilon_j\right] \\ &= \hat{Y} + \sum_j w_j E_\epsilon[\epsilon_j]. \end{aligned}$$

Since $E[\epsilon] = 0$,

$$E_\epsilon[\hat{\tilde{Y}}] = \hat{Y}.$$

Thus the expectation of the noisy estimate is the clean estimate. The LIME solution is unbiased by the presence of additive zero-mean noise on the training observations.

3.4.2 LIME expectation unaffected by noise on training features

Consider noisy independent training variables $\{\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_k\}$, which are the original independent training variables $\{X_1, X_2, \ldots, X_k\}$ infected with iid zero mean additive noise ϵ_j, $j = 1, \ldots, k$.

LIME solves for the weights that minimize $D - \lambda H$. Let the distortion D be some norm $\|\cdot\|$ of the difference of the reconstruction and original test point,

$$D = \|\sum_j w_j X_j - x\|.$$

The noisy training data are,

$$\tilde{X}_j = X_j + \epsilon_j.$$

Then the noisy LIME estimate solves for the minimizing weights,

$$\begin{aligned} & \arg\min_w \|\sum_j w_j(X_j + \epsilon_j) - X\| - \lambda H(w) \\ \equiv \quad & \arg\min_w \|\sum_j w_j X_j + \sum_j w_j \epsilon_j - X\| - \lambda H(w) \\ \equiv \quad & \arg\min_w \|\sum_j w_j X_j - (X - \sum_j w_j \epsilon_j)\| - \lambda H(w). \end{aligned}$$

The last line specifies the LIME weight solution for estimating the test point $X - \sum_j w_j \epsilon_j$ with clean training data. Thus iid zero mean additive noise on the independent training data creates a LIME estimate for a different test point, moved by the noise. That is, additive noise on the training points acts as additive noise on the test point,

$$f_{LIME}(X, \tilde{X}_1, \tilde{X}_2, \ldots \tilde{X}_k) = f_{LIME}(\tilde{X}, X_1, X_2, \ldots, X_k).$$

where $\tilde{X} = X - \sum_{j=1}^{k} w_j \epsilon_j$.

The actual effect of estimating at the dirty test point depends heavily on the training data and the function we are estimating. However, we can see that the new weight solution is unbiased in the sense that the expectation of the noisy estimate with respect to the noise E_ϵ is equal to X,

$$\begin{aligned} E_\epsilon \left[X - \sum_j w_j \epsilon_j \right] &= X - \sum_j w_j E_\epsilon[\epsilon_j] \\ &= X, \end{aligned}$$

since the expectation of each noise random variable is zero.

Thus, iid zero mean additive noise on the training features $\{X_1, X_2, \ldots, X_k\}$ is mathematically equivalent to noise on the test point, $\tilde{X} = X - \sum_j w_j \epsilon_j$. Further, we have shown that the expectation of the noisy test point is equal to the true test point, $E[\tilde{X}] = X$.

3.4.3 Variation of the law of large numbers

To show that the noisy LIME estimates converge to the LIME estimate without noise, a variation of the law of large numbers must be shown to hold for the LIME weights. The variation of the law of large numbers shows that n iid random variables weighted by LIME weights will converge to the random variables' expectation in the limit $n \to \infty$. There are a number of generalizations of the law of large numbers [96] [121]. One such theorem from Taylor [121] shows that a law of large numbers holds for any weight sequence that forms a Toeplitz sequence.

First, we review the definition of a Toeplitz sequence, then show that the LIME weights form a Toeplitz sequence, and then apply Taylor's theorem to conclude that the LIME weights obey a law of large numbers.

Definition: A double array $\{w_{nj} : n, j = 1, 2, \ldots\}$ of real numbers is said to be a Toeplitz sequence if

C1) $\lim_{n \to \infty} w_{nj} = 0$ for each j

C2) $\sum_{j=1}^{\infty} \|w_{nj}\| \leq C$ for each n
where C is a finite constant.

Lemma 7 *A LIME weight sequence* $\{w_{nj} : j = 1, \ldots, k; n = 1, 2, \ldots\}$*, where* $n \to \infty$*,* $j = 1$ *to* k*,* $k \to \infty$*, and* $k/n \to 0$*, is a Toeplitz sequence.*

Proof: In Section 3.3.3, it was shown that $\max_j w_{nj} \to 0$, and since $0 \leq w_j \leq \max_j w_{nj}$ for all j, $w_{nj} \to 0$ for all j. Thus condition C1 of the definition is satisfied.

Secondly, since $w_{nj} \geq 0$ by definition and $\sum_{j=1}^{k}(n) w_{nj} = 1$ for all $k(n)$, the second condition is satisfied with $C = 1$. Therefore, the lemma holds. q.e.d.

A theorem from Taylor [121](which Taylor credits to Pruitt [101]) is also needed.

Theorem 8 (Taylor) *Let* $Z_1, Z_2, \ldots$ *be a sequence of real-valued scalar iid r.v.'s such that* $E\|Z_1\| < \infty$ *and let* $\{a_{nj} : n, j = 1, 2, \ldots\}$ *be a Toeplitz sequence such that*

$$\sum_{j=1}^{\infty} \|a_{nj}\| \leq 1.$$

A necessary and sufficient condition that

$$\sum_{j=1}^{\infty} a_{nj} Z_j \to E[Z_1]$$

in probability is that the $\max_j \|a_{nj}\| \to 0$*.*

Taylor's theorem can be applied to the LIME weights.

Lemma 8 (Law of large numbers for LIME) *For iid random variables* $\{\epsilon_j : j = 1, 2, \ldots\}$ *with finite mean and finite variance, and LIME weight sequence* $\{w_{nj} : j = 1, \ldots, k; n = 1, 2, \ldots\}$ *for a given test point, the weighted sum* $\sum_{j=1}^{k(n)} w_{nj}\epsilon_j$ *converges to* $E[\epsilon_1]$*,*

$$\sum_{j=1}^{k(n)} w_{nj}\epsilon_j \to E[\epsilon_1]$$

in probability as $n \to \infty$*,* $k \to \infty$*, and* $k/n \to \infty$*.*

Proof: We apply Taylor's theorem (Theorem 8) such that ϵ takes the place of Y, and the LIME weight sequence w takes the place of the theorem's a. By assumption, ϵ is a sequence of iid random variables with finite variance, and it is shown in Lemma 7 that the LIME weight sequence w forms a Toeplitz sequence. Furthermore, we showed in Section 3.3.3 that the maximum LIME weight converges to zero, $\max_j \|w_{nj}\| \to 0$ as $k \to \infty$, $n \to \infty$ and $k/n \to 0$.

Hence one can conclude from applying Taylor's Theorem 8 that

$$\sum_{j=1}^{k(n)} w_{nj}\epsilon_j \to E[\epsilon_1]$$

as $n \to \infty$, $k \to \infty$, and $k/n \to \infty$. q.e.d.

This variation of the law of large numbers will be useful to prove that LIME converges to the clean estimate in the presence of additive iid finite mean noise.

3.4.4 LIME solution converges for noisy training observations

Lemma 9 *A LIME estimate $\hat{\tilde{Y}}_n$ made in the presence of iid zero mean, finite variance, additive noise $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_k\}$ on the training observations $\{Y_1, Y_2, \ldots, Y_k\}$ will converge to the clean LIME estimate $\hat{Y}$ in probability as $k \to \infty$, $n \to \infty$, and $k/n \to 0$.*

Proof: Each noisy training observation is $\tilde{Y}_j = Y_j + \epsilon_j$, the clean estimate is $\hat{Y} = \sum_j w_j Y_j$, and the noisy LIME estimate is (where the sum is over the test point's neighborhood),

$$\begin{aligned}\hat{\hat{Y}}_n &= \sum_{j=1}^{k(n)} w_{nj}(Y_j + \epsilon_j) \\ &= \sum_{j=1}^{k(n)} w_{nj} Y_j + \sum_{j=1}^{k(n)} w_{nj}\epsilon_j \\ &= \hat{Y} + \sum_{j=1}^{k(n)} w_{nj}\epsilon_j\end{aligned}$$

Since, by the variation of the law of large numbers proved above, $\sum_{j=1}^{k(n)} w_{nj}\epsilon_j \to 0$ as $k \to \infty$, $n \to \infty$ and $k/n \to 0$, we conclude that under those asymptotic conditions, $\hat{\hat{Y}}_n \to \hat{Y}$. q.e.d.

3.4.5 LIME solution converges for noisy training features

Lemma 10 *A LIME estimate made in the presence of iid, zero mean, finite variance, additive noise on the training features will converge in probability to the clean LIME estimate asymptotically as $k \to \infty$, $n \to \infty$, and $k/n \to 0$.*

Proof: In Section 3.4.2 it was shown that additive noise on the independent training points causes the LIME algorithm to solve for weights that actually estimate a noisy test point,

$$f_n(\tilde{X}) = f_n(X - \sum_{j=1}^{k(n)} w_{nj}\epsilon_j).$$

From the variation of the law of large numbers shown in Theorem 8, it can be concluded that

$$\left(X - \sum_{j=1}^{k(n)} w_{nj}\epsilon_j\right) \to X$$

as $k \to \infty$, $n \to \infty$, and $k/n \to 0$. Hence, asymptotically the noisy LIME estimate converges in probability to the clean estimate,

$$f_n(\tilde{X}) = f_n(X - \sum_{j=1}^{k(n)} w_{nj}\epsilon_j) \to f(X).$$

q.e.d.

3.5 Functions that are fit exactly

What kind of surfaces are fit by LIME? In the extreme as $\lambda \to \infty$, maximizing the entropy of the weights will be the focus (see Section 3.1) regardless of distortion. Then all points within a neighborhood will receive similar weight, and one expects that LIME will fit piecewise constant surfaces over disjoint neighborhoods, like a k-NN regression.

The other extreme, as $\lambda \to 0$, is more interesting. Based on simulations, a conjecture is that as $\lambda \to 0$, the LIME weights will be equivalent to the weights that solve the problem: maximize $H(w)$ such that $D(w) = \min_v D(v)$. For this problem, it will be shown that hyperplanes are fit exactly. Also, an analytical form for the surfaces fit on rectangular grids is shown in Section 4.4.3 with two-dimensional examples in Section 4.4.3.

3.5.1 Fitting hyperplanes

Lemma 11 *Given a linear relationship between the feature space X and the observation space Y such that $Y = a^T X + b$, and a set of training data samples (X_j, Y_j), $j = 1, \ldots, k$, then the estimate $\hat{Y} = \sum_j w_j(X) Y_j$, where w solves:* $\max H(w)$ *such that $D(w) = \min_v D(v)$, is exact for all test points X that are in the closure of the convex hull spanned by the feature samples $\{X_1, X_2, \ldots, X_k\}$.*

Proof: Any test point in the closure of the convex hull of the training features can be reproduced with zero distortion by some pmf over the training features by definition.

Then the set of weight distributions satisfying the minimum distortion criterion must yield zero distortion, that is, $\sum_{j=1}^{k} w_j(X)X_j = X$.

The observations are from a hyperplane, and thus each $Y_j = a^T X_j + b$ for some vector a and scalar b. Substituting this form for Y_j, the estimate becomes

$$\begin{aligned} \hat{Y} &= \sum_{j=1}^{k} \big(w_j(X)a^T X_j + w_j(X)b\big) \\ &= a^T \sum_{j=1}^{k} w_j(X)X_j + b\sum_{j} w_j(X). \end{aligned}$$

Since $\sum_{j=1}^{k} w_j(X)X_j = X$, and since $\sum_j w_j(X) = 1$, this resolves to,

$$\hat{Y} = a^T X + b.$$

And thus the estimation of test points in the closure of the convex hull spanned by the training points exactly reconstructs the original hyperplane.

Chapter 4

LIME and regular grids

Andamos sobre un espejo sin azogue
sobre un cristal sin nubes

We walk on an unsilvered mirror
on a crystal without clouds

Federico García Lorca

In some learning applications, the training samples are taken at vertices of a regular grid. A grid may be efficient for data collection, data display, or look-up table (LUT) data interpolation. Examples include geological sampling and digital video. In some cases, the grid data are not original data samples, but are prototypes based on (perhaps learned from) original non-grid data. For example, color management conversions represent information about the color display characteristics via a grid of training vectors that must be learned from actual non-grid color samples. In applications such as computer tomography, a rectilinear grid of training vectors needs to be created based on a non-rectilinear grid of original data samples.

Regular grids and interpolation appear in a wide range of applications, from medical imaging [102] to the finite element method [11]. Many physical engineering fields and scientific explorations, such as oceanography or minerals prospecting, also use regular grid sampling and interpolation. The chapter contains an in-depth look at

creating and interpolating the 3D regular grids used in current color management standards to maintain consistency across color displays.

There are two basic problems for interpolations based on regular grids: how to create a regular grid from non-grid original samples, and given the grid, how to perform interpolations. The LIME method can yield robust solutions for both problems.

The LIME method can be used to form grids from original training samples. Experiments with color management data provide a comparison of LIME with local linear regression, ridge regression, radial basis neural net, and other common methods.

A common goal is to interpolate test points based on the grid of training vectors. A traditional family of methods for interpolating grids includes linear, bilinear, and trilinear interpolation. In this work, this family of methods is generalized to any dimension and term the family 'product linear interpolation' (PLI). PLI for any dimension is then shown to be a special case of the LIME method where the distortion is first minimized and then the entropy of the solution is maximized. Thus a link is formed between the general LIME method, which uses any number of training samples with any structure, and the traditional methods for interpolating grids.

The chapter ends with three different grid cell interpolation experiments: a color management dataset, a simulation with additive noise, and a functional approximation.

4.1 Color management basics

LIME can be applied to the three-dimensional regression problems that occur in color management. Experiments show that LIME can be useful for creating the three dimensional look-up-tables used in contemporary color management. Also, when used to interpolate test points off the grid, LIME affords possible advantages in decreasing average interpolation error and variance.

During its lifetime, a color image may pass through many devices: captured with a digital camera, edited on a monitor, edited later on a laptop with LCD screen, printed, copied, re-scanned, etc. It is disturbing to users when colors in an image change from device to device. Color image quality is preserved by device color transformations that

appropriately map the input device's colorspace to the output colorspace. Quality color transformations require quality characterizations of a device's color space.

There are three types of device characterization [33]. First, *physical modeling* uses a complex mathematical model to account for physical differences in devices. Few measurements are needed, but good models are difficult to build for nonlinear devices like printers. Second, *empirical modeling* fits a statistical model to the measured differences between two devices. Lastly, *exhaustive measurement* colorimetrically measures as many sample points of the transformation as possible. Points in-between the sampled points are interpolated. Printers, due to their nonlinearity, are often characterized in this manner.

To manage colors between possibly nonlinear devices, the International Color Consortium has created a color management standard called the ICC profile [2]. An ICC profile format allows for color transformations between devices using LUT's and interpolation. No algorithms for creating the LUT or performing the interpolation are specified. The ICC profile format is an important emerging international standard for color management between disparate devices, including digital cameras, printers, and projectors.

The goal is to characterize the output of the device in terms of a device-independent input space. Many color devices work with an 8 bit RGB representation. Take the color (255,255,255) RGB, which is the brightest white an RGB device can produce. Clearly, the actual white produced will differ greatly between devices. Thus the RGB representation is a device-dependent representation of colors.

For these experiments, the CIEL*a*b* space is used, which is a common device-independent input space. CIEL*a*b* has three components, the channel $L \in [0, 100]$ which correlates with perceived luminance, the channel $a \in [-100, 100]$ which ranges from a saturated green a saturated red, and the channel $b \in [-100, 100]$ which ranges from a saturated blue to a saturated yellow. CIEL*a*b* color values correspond to actual reflectance spectra under defined viewing conditions. Once an image is specified in CIEL*a*b* space, the ICC profile for a specific device (and specific lighting conditions, humidity, inks, and paper) will specify how to convert the image from CIEL*a*b* space to the device's RGB space.

Consider the 1 bit CMYK color printer, two species of which, the *inkus jetus* and the *common laserat printeri*, can be found in offices and homes everywhere. These printing devices accept 8 bit RGB input. Within the device, an undercolor removal process changes the 8 bit RGB input to 8 bit CMYK. Another internal step is halftoning, which may upsample the data and which converts the image to 1 bit CMYK, which is printed. Printed color patches can then be measured with a spectrometer and the calibrated reflectance spectra converted into CIEL*a*b* units.

Thus a printing device maps an RGB input to a CIEL*a*b* output. Given a desired color in CIEL*a*b*, a color management system determines, 'What is the RGB value to send to the printing device so that the desired CIEL*a*b* color is printed?'

The color management system must estimate the function f that maps a three-dimensional colorspace (RGB) to another three-dimensional colorspace (CIEL*a*b*). The brute force solution is to print patches of all $256 \times 256 \times 256$ possible 8 bit RGB input values. All 2^{24} patches could have their CIEL*a*b* value measured, and an exhaustive table of all this data could be stored in the device. However, no part of that solution is efficient. Instead, the ICC profile includes a three dimensional rectangular grid of samples of the mapping f between CIEL*a*b* and RGB. First an appropriate grid of samples of the function must be built and stored in the profile. Then, during printing, the profile's grid is accessed and interpolated to estimate the function f to convert the desired input CIEL*a*b* color to the appropriate device RGB.

4.2 Estimating a grid for color management

A key problem for ICC profiles is to build a regular grid of CIEL*a*b* values and their corresponding RGB values for a device. The grid will serve as an LUT from which to estimate the appropriate device RGB for any input CIEL*a*b* value. An ICC profile may also contain a gamma-correcting 1-d LUT before and after the 3-D LUT.

How does one obtain a rectangular grid of CIEL*a*b* values and corresponding

RGB values? Unfortunately, the printer accepts RGB, not CIEL*a*b*. The usual solution is to send a number of RGB patches (usually a regular sampling, with perhaps extra gray ramps) to the device and measure the output CIEL*a*b* values. Then, these training samples are used to estimate the correspondences for a regular grid of CIEL*a*b* values.

Once a grid has been estimated it cannot be directly evaluated, as the 'true grid' is practically impossible to obtain. The following experiment is proposed to evaluate the estimation of the grid by stepping further through the color management path:

Color Management Grid Estimation Experiment

For a given printing device,

Step 1) Print a large target of RGB test patches
Step 2) Print a small target of RGB test patches
Step 3) Measure the CIEL*a*b* values of both sets of test patches
Step 4) Create a matrix B which is a $17 \times 17 \times 17$ rectangular grid of CIEL*a*b* values, with equal sampling $([0, 100], [-100, 100], [-100, 100])$
Step 5) Based on the large (or small) target's RGB to CIEL*a*b* samples, estimate the RGB values for the grid B.
Step 6) Based on the estimated grid, use trilinear interpolation to estimate the small (or large) target's RGB value for each CIEL*a*b* value.
Step 7) Evaluate the grid estimation of Step 5 by measuring the mean RGB error length between the known small (or large) target's RGB values and the estimated RGB values from Step 6.

Many learning algorithms require training of some parameter. It is common to train learning algorithm parameters by cross-validation on the training data set [54]. However, leave-some-out cross-validation results in a grid with severely decreased accuracy, and preliminary experiments showed that the training neighborhood size for local linear regression that resulted was far from the optimal parameter for the test set.

Instead, if the experiment was to use the large target as a training set and the

small target as a test set, then each learning algorithm was trained by estimating the grid with the large target and then testing on the large target. The learning algorithm parameter was trained to deliver the best performance for testing on the large target. After this training, the optimal parameter was used to estimate the grid values and test (Step 6) on the small test set. The other advantage of learning this way is the increased speed compared with leave-some-out cross-validation.

The data for the experiment are a large and small test chart that are used in the company Chromix's [1] service to create ICC profiles. The large test chart consists of 918 color patches which span RGB space as well as some extra bright colors and gray ramps. The small test chart consists of 288 color patches which span RGB space as well as some extra bright colors and gray ramps. The test charts were printed on a 600 dpi error diffusing Savin laser printer with EFI E650 Fiery RIP. All of the printer's color correction and color management were turned off.

In the next few paragraphs, it is noted how the parameters were trained for each of the different algorithms used in the experiments:

Local linear regression: Local linear regression was applied to the training set to estimate the grid over a neighborhood of k nearest neighbors. The parameter k was trained by minimizing the empirical error of the trilinear interpolation of the grid in estimating the training set. Training on the large Savin dataset yielded $k = 52$. Training on the small Savin dataset yielded $k = 20$.

Local ridge regression: Local ridge regression [54] was applied to the training set to estimate the grid over a neighborhood of k nearest neighbors. The parameters k and the smoothness parameter s were trained by minimizing the empirical error of the trilinear interpolation of the grid in estimating the training set. The smoothness parameter s was trained with intervals of .1. For the large Savin dataset, $k = 35$ and $s = 3.4$. For the small Savin dataset, $k = 18$ and $s = .6$.

LIME: Training for LIME led to a grossly overfit neighborhood of 2 or three training points. The overfitting was due to using the training points to estimate the grid and also to judge the grid. To avoid the overfitting, we implemented the LIME method by using the neighborhood size chosen by the local linear regression method. Given k, the delta parameter was trained empirically to within multiplicative intervals

of 10 ($\lambda = .001, .01, .1, 1, 10$), and optimized to $\lambda = 1$.

k-NN: For comparison sake, k-NN regression was run with $k = 1$ and k set to the neighborhood optimized for local linear regression ($k = 52$, $k = 20$).

Radial basis neural net: A generalized regression neural network was designed using the 'newgrnn' program available in the neural net toolbox for Matlab 6.1 [3]. The network has two layers, the first layer is composed of radial basis neurons, and the second layer is purelin neurons. The code for the program can be obtained by typing 'type newgrnn' at a matlab prompt. More details on these kinds of neural network methods can be found in [125]. A smoothing parameter 'spread' is called for in the design. The spread parameter was trained by minimizing the empirical error of the trilinear interpolation of the grid in estimating the training set. Intervals of .1 were tested. For training on the large Savin dataset, $spread = 3.2$. For training on the small Savin dataset, $spread = 3.2$. To ensure that overfitting was not a problem, another net was built for each dataset with the smooth parameter set to the mean Euclidean distance between input vectors, $spread = 3.5722$ for training with the large Savin dataset and $spread = 5.3772$ for training with the small Savin dataset.

A limitation of the experiment is that the evaluation is mean-squared-error in the device RGB space. A device's RGB space tends not to be perceptually uniform, and equal sized errors may result in unequal color differences. A better metric would be to print the estimated test RGB values, measure the corresponding test CIEL*a*b* values, and compare these CIEL*a*b* values to the original test set CIEL*a*b*. That would directly measure how far the actual printed values differ for each desired CIEL*a*b* value. Preliminary experiments showed, however, that the average of RGB errors from RGB test values spread throughout the colorspace does correlate well with actual visual error. If instead CIEL*a*b* measurements were used as the metric, the experiment would also be prohibitively difficult due to the number of samples to be measured to train the learning algorithm parameters. An actual ICC profile building service (which will build an ICC profile for around $100 dollars [1]) would have to train on RGB values. Lastly, the experiment with the RGB metric relies only on the small and large target data, and the device is no longer needed. Thus other experimenters can compare using the same datasets without needing the

original device (and original printing conditions) to evaluate the comparison.

Regression method	Trained on large dataset	Trained on small dataset
1-Nearest neighbor regression	38.89	46.43
k-Nearest neighbor regression	45.37	66.65
Local linear regression	34.34	41.33
Local ridge regression	33.94	41.23
Radial basis net with trained spread	35.04	43.73
Radial basis net with mean dist. spread	34.75	42.80
LIME	32.54	42.52

Table 4.1: Mean RGB error lengths for color management grid estimation

The results are shown in Table 4.1. The first column shows results for training on the larger test chart, and testing on the small test chart. The second column shows the results for training on the smaller test chart and testing on the large test chart.

For both experiments LIME performed competitively. To estimate the grid using the small target, LIME was 3% worse than the best method, local ridge regression. To estimate the grid using the large target, LIME performed 4.3% better than the next best method, local ridge regression.

The LIME algorithm took the longest time to run of all the algorithms tested; approximately ten times longer to run than the local linear regression method. This is not expected to be a problem though as building an ICC profile is not a real-time operation and other steps in the process (such as paper handling and measuring the patches) dwarf the computational run-time. The algorithm of choice for a particular company to estimate the grid is generally a trade-secret.

4.3 Interpolating a grid

Interpolating a grid is necessary in many applications, from computer graphics to weather estimation. Anisotropic (irregular) grids can yield lower error rates than regular grids and are engendering some research [29]. The LIME method can interpolate

any arbitrary grid. However, regular rectangular grids are computationally efficient and in this section we focus on regular grids. A common way to interpolate grids is traditional linear interpolation, using a convex hull around the test point of $d+1$ grid points in d dimensions [66], [110]. However, traditional linear interpolation rarely leads to the best possible estimate. Traditional linear interpolation may also be difficult to implement because a convex hull around the test point must be determined, a difficult task for three or higher feature dimensions. Fast methods to discover a convex hull exist [110], but do not guarantee that the best convex hull will be found.

A number of three-dimensional grid interpolation algorithms have been proposed in the color management literature. Tetrahedral interpolation [69] is the linear interpolation equations applied to four points forming a convex hull around a test point. Trilinear interpolation [66] is a linear interpolation applied to all 8 vertices of the grid cell that contains the test point. In the next section, we generalize trilinear interpolation to any number of dimensions and show that it relates to the LIME method. Other methods proposed in the color management literature interpolate a test point using five or six of the surrounding grid vertices [66].

In Kasson et al. [70], several linear interpolation methods are compared and their experiments show that tetrahedral interpolation provides the best accuracy (and at low computational cost). A wavelet-based nonlinear color transformation method was shown to be less accurate than trilinear interpolation [64]. Neural net methods have been shown to give good results in a number of papers, including [120, 65]. However, neural nets require parameter-tuning and training, may become overfit, and it has not been clearly demonstrated that they will perform better than simple trilinear interpolation.

In the next section we generalize trilinear interpolation and call the generalization 'product linear interpolation' (PLI). A formula for the PLI weights is given, and it is shown that the PLI weights are a special case of the LIME weights. A closed form is given for the surfaces fit using PLI weights, with some two-dimensional examples.

The chapter ends with three experiments. The first compares grid interpolation methods for a color management dataset from Apple. The second is a simulation that considers the effects on estimation of additive noise. The third experiment considers

functional approximation over a grid cell.

4.4 Product linear interpolation

The popular methods of bilinear and trilinear interpolation [66] can be directly generalized to d dimensions. We term this generalization 'product linear interpolation' (PLI) because it works by forming independent weights for each sample point for each dimension, and then multiplying the weights over the dimensions. However, like bilinear and trilinear interpolation, PLI only works on rectangular grids. In Section 4.4.2, we will show that PLI is a special subcase of LIME.

A method for creating regular grids of training points is sequential linear interpolation (SLI) [20]. Once the SLI algorithm has selected the training samples, Chang et al. suggest using bilinear or trilinear interpolation to actually create the estimates. They propose that SLI can be generalized to higher dimensions. PLI is the straightforward multi-dimensional generalization of bilinear and trilinear interpolation.

4.4.1 Formula for product linear interpolation

Product linear interpolation (PLI) is a generalization of linear interpolation that works with any d-dimensional regular rectangular grid. Training samples must be given for the vertices of the rectangular grid. A test point's neighborhood is defined to be all the 2^d vertices of its surrounding rectangular convex hull.

The PLI weights do not depend on shifts, and thus without loss of generality, we assume that the rectangular hull that forms the local neighborhood around X has been shifted so that the entire neighborhood lies in the positive quadrant with one vertex at the origin. Furthermore, since the weights only depend on relative distances in each dimension, we can without loss of generality assume that the neighborhood and test point have been scaled so that the neighborhood points are at the corners of the d-dimensional unit cube. The training features take values at the corners of the unit hypercube, $X_j \in \{0,1\}^d$ for all j. The test point is contained in the closure of the unit hypercube, $X \in [0,1]^d$. For d dimensions the neighborhood size is $k = 2^d$.

Then the PLI weights are as follows, where $X[m]$ denotes the mth component of the d dimensional vector X and absolute value is represented $\|\cdot\|$,

$$w_j(X, X_1, X_2, \ldots, X_k) = \prod_{m=1}^{d} \|1 - X_j[m] - X[m]\| \tag{4.1}$$

The PLI estimate is then formed by applying the weights to the training observations,

$$\hat{Y} = \sum_j w_j Y_j$$

In the following theorem, it is established that the PLI weights as given in (4.1) do generalize linear interpolation by satisfying the linear interpolation equations.

Theorem 9 *Let $X_j \in 0, 1^d$ for all $j = 1$ to 2^d, and $X \in [0, 1]^d$.*

Then the PLI weights,

$$w_j(X, X_1, X_2, \ldots, X_k) = \prod_{m=1}^{d} \|1 - X_j[m] - X[m]\| \tag{4.2}$$

satisfy the linear interpolation constraints,

C1) $w_j \geq 0$ for all j

C2) $\sum_{j=1}^{k} w_j = 1$

C3) $\sum_{j=1}^{k} w_j X_j = X$

Proof: C1 clearly holds because each weight is formed as a product of absolute values. Next condition C2 and condition C3 are shown by an inductive proof. First, it is seen that C2 and C3 hold for a base case, a rectangular grid of dimension $d = 1$. Then, it is shown that if conditions C2 and C3 hold for a rectangular grid of dimension m, then they will also hold for a dimension of $m + 1$.

Base Case: For $d = 1$, the training samples are $(0, Y_1)$ and $(1, Y_2)$, and the PLI weights are $w_1 = 1 - X$ and $w_2 = X$. Then $1 - X + X = 1$, so condition C2 is satisfied. Also, $(1 - X)0 + X(1) = X$, so condition C3 is satisfied.

Inductive Step: Assume that the PLI weights satisfy conditions C2 and C3 for d dimensions, then we show that they will also satisfy conditions C2 and C3 for $d+1$ dimensions.

Specifically, assume that conditions C2 and C3 hold as follows,

$$\begin{aligned}\sum_{i=1}^{k}\prod_{m=1}^{d}\|1-X_i[m]-X[m]\| &= 1\\ \sum_{i=1}^{k}\prod_{m=1}^{d}\|1-X_i[m]-X[m]\|X_i &= X\end{aligned}$$

Next, consider the corresponding $d+1$-dimensional situation. Now let the $2k$ training samples be denoted $V_j = [X_i\, 0]$ for $j = 1$ to k and $V_j = [X_i\, 1]$ for $j = k+1$ to $2k$. The test point $V \in [0,1]^{m+1}$ is defined $V[m] = X[m]$ for $m = 1$ to d, and the $d+1$th component of the vector V can be any point in the closed set, $V[d+1] \in [0,1]$.

It must be shown that condition C2 holds for the $d+1$-dimensional case given that C2 holds for the d-dimensional case.

$$\sum_{j=1}^{2k} w_j(V, V_1, V_2, \ldots, V_{2k}) = \sum_{j=1}^{2k}\prod_{m=1}^{d+1}\|1-V_j[m]-V[m]\|$$

$$\begin{aligned}&= \sum_{j=1}^{k}\left(\prod_{m=1}^{d}\|1-V_j[m]-V[m]\|\right)(\|1-V_j[d+1]-V[d+1]\|)\\ &\quad+\sum_{j=k+1}^{2k}\left(\prod_{m=1}^{d}\|1-V_j[m]-V[m]\|\right)(\|1-V_j[d+1]-V[d+1]\|)\end{aligned}$$

Now, using the relations between V_j and X_i, and between V and X, one can rewrite the above statement as,

$$\sum_{j=1}^{2k} w_j(V, V_1, V_2, \ldots, V_{2k}) =$$
$$\sum_{i=1}^{k} \left(\prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| \right) (\|1 - 0 - V[d+1]\|)$$
$$+ \sum_{i=1}^{k} \left(\prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| \right) (\|1 - 1 - V[d+1]\|)$$

$$\begin{aligned} &= [(1 - V[d+1]) + V[d+1]] \left[\sum_{i=1}^{k} \prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| \right] \\ &= \sum_{i=1}^{k} \prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| \\ &= 1 \end{aligned}$$

where the last step holds because condition C2 was known to hold for d dimensions. Thus C2 also holds for the $d+1$ dimensional case.

Next, consider condition C3.

$$\sum_{j=1}^{2k} w_j(V, V_1, V_2, \ldots, V_{2k}) V_j = \sum_{j=1}^{2k} \prod_{m=1}^{d+1} \|1 - V_j[m] - V[m]\| V_j$$

$$= \sum_{j=1}^{k} \left(\prod_{m=1}^{d} \|1 - V_j[m] - V[m]\| \right) (1 - V_j[d+1] - V[d+1])\, V_j$$
$$+ \sum_{j=k+1}^{2k} \left(\prod_{m=1}^{d} \|1 - V_j[m] - V[m]\| \right) (1 - V_j[d+1] - V[d+1])\, V_j$$

Substitute X_i for V_j as appropriate, and split the equation into two parts, one equation for the first d components, and another equation for the $d+1$th component.

The first d components of the above equation can be written

$$
\begin{aligned}
& \sum_{i=1}^{k} \left(\prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| \right) (1 - V[d+1]) X_i \\
& + \sum_{i=1}^{2k} \left(\prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| \right) V[d+1] X_i \\
= \; & [(1 - V[d+1]) + V[d+1]] \left[\sum_{i=1}^{k} \prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| \right] X_i \\
= \; & \sum_{i=1}^{k} \prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| X_i \\
= \; & \sum_{i=1}^{k} w_i X_i \\
= \; & X
\end{aligned}
$$

Thus one sees that the first d components of the $d+1$ vector expression $\sum_{j=1}^{2k} w_j V_j$ are equal to the d-dimensional test variable X, and thus also equal to the first d dimensions of the $d+1$ dimension test variable V.

Now consider the last ($d+1$th) vector component of $\sum_{j=1}^{2k} w_j V_j$,

$$
\begin{aligned}
& \left(\sum_{j=1}^{2k} w_j V_j \right) [d+1] \\
= \; & \sum_{i=1}^{k} \left(\prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| \right) (1 - V[d+1]) V_i[d+1] \\
& + \sum_{i=1}^{k} \left(\prod_{m=1}^{d} \|1 - X_i[m] - X[m]\| \right) V[d+1] V_{i+k}[d+1]
\end{aligned}
$$

The jth training sample's $d+1$th component value $V_j[d+1]$ is defined to be $V_j[d+1] = 0$ for $j = 1$ to k and $V_j[d+1] = 1$ for $j = k+1$ to $2k$. Substituting in these values yields,

$$\begin{aligned}\left(\sum_{j=1}^{2k} w_j V_j\right)[d+1] &= \sum_{i=1}^{k}\left(\prod_{m=1}^{d}\|1 - X_i[m] - X[m]\|\right) V[d+1] \\ &= V[d+1]\sum_{i=1}^{k}\left(\prod_{m=1}^{d}\|1 - X_i[m] - X[m]\|\right) \\ &= V[d+1]\sum_{j=1}^{k} w_i \\ &= V[d+1]\end{aligned}$$

The result is that $\sum_{j=1}^{2k} w_j V_j$ yields a vector whose first d components are equivalent to X and whose $d+1$th component is equal to $V[d+1]$.

In summary,

$$\sum_{j=1}^{2k} w_j V_j = |X\ V[d+1]|^T = V.$$

Therefore, condition C3 holds.

Then, it has been shown that the PLI weights satisfy the linear interpolation conditions for the one dimension, and that given that the conditions are satisfied for d dimensions, then the conditions will remain satisfied for the corresponding $d+1$ dimensional problem. In conclusion, the PLI weights satisfy the linear interpolation equations. q.e.d.

4.4.2 PLI and LIME

In Section 4.4, it was proposed that product linear interpolation (PLI) extends linear interpolation to a neighborhood of 2^d training points that form a rectangular hull around a test point X in d dimensions. PLI is convenient for interpolating LUT grids. As previously discussed, for PLI the neighborhood points are at the corners of a d-dimensional unit cube and the test point $X \in [0,1]^d$.

The following theorem states that of all the weight distributions that exactly satisfy $\sum_j w_j X_j = X$, the PLI weights have the maximum entropy. A corollary after

the theorem establishes that for l_1 distortion and $\lambda \leq$ some Λ, the LIME weights are the same as the PLI weights.

Theorem 10 *Let $X_j \in \{0,1\}^d$ for all $j = 1$ to 2^d, and $X \in [0,1]^d$. Then the PLI weight distribution*

$$w_j(X, X_1, X_2, \ldots, X_k) = \prod_{m=1}^{d} \|1 - X_j[m] - X[m]\|$$

is the weight distribution that solves $\arg\max_w (H(w)|D(w) = 0)$

Proof: Recall Theorem 1 discussed previously in Section 3.2.1, which shows that the maximum entropy weights given a moment constraint $D(w) = 0$ have an exponential form. According to the theorem, the weight distribution $w_j^*(X) = \gamma e^{-\alpha^T X_j}$, where α and γ satisfy conditions C1-C3, uniquely maximizes the entropy given the mean condition. We will present a γ and α that satisfy these requirements and show that $w_j^*(X) = \gamma e^{-\alpha^T X_j}$ for all j is equivalent to the PLI weights. Thus, the PLI weights are the maximum entropy weights given the mean constraint $D(\sum_j w_j X_j, X) = 0$.

Let

$$\gamma = \prod_{m=1}^{d} (1 - X[m])$$

and let the mth component of the vector α be as follows,

$$\alpha[m] = \ln\left(\frac{X[m]}{1 - X[m]}\right)$$

Substituting γ and α into the equation for the maximum entropy weight distribution $w_j^*(X) = \gamma e^{-\alpha^T X_j}$ yields,

$$\begin{aligned} w_j^*(X) &= \left[\prod_{m=1}^{d} (1 - X[m])\right] e^{\sum_{m=1}^{d}\left[\ln\left(\frac{X[m]}{1-X[m]}\right)\right] X_j[m]} \\ &= \left[\prod_{m=1}^{d} (1 - X[m])\right] \left[\prod_{m=1}^{d} e^{\left[\ln\left(\frac{X[m]}{1-X[m]}\right)\right] X_j[m]}\right] \\ &= \prod_{m=1}^{d} (1 - X[m]) \left(\frac{X[m]}{1 - X[m]}\right)^{X_j[m]} \end{aligned}$$

The neighborhood of training samples X_j is restricted to the vertices of a unit cube, $X_j[m] = \{0, 1\}$ for all m. Then $X_j[m]$ acts like a selector in the above weight equation. If $X_j[m] = 1$, then that vector component selects $X[m]$. Or if $X_j[m] = 0$, the vector component selects $1 - X[m]$. Thus this relation can be rewritten as,

$$w_j^*(X) = \prod_{m=1}^{d} \|1 - X_j[m] - X[m]\|$$

Note that the above equation is exactly the formula for the PLI weights. Thus, a γ and α have been shown for the exponential form that result in the PLI weights. Thus the PLI weights must be the unique maximum entropy weights given the mean constraint. q.e.d.

Corollary 3 *Let $X_j \in 0, 1^d$ for all $j = 1$ to 2^d, and $X \in [0, 1]^d$. Then the constrained l_1 LIME minimization problem (see Section 3.2.2) has a solution, and let Λ be the reciprocal of the largest Lagrangian multiplier for its solution. If $\lambda \leq \Lambda$, then a solution of the l_1 LIME minimization problem is equivalent to the PLI weight distribution.*

Proof: The constrained l_1 solution solves: maximize $H(w)$ such that $D(w) = 0$, where $D(w)$ is the l_1 distortion. By Theorem 10, the PLI weight distribution is shown to also solve: maximize $H(w)$ such that $D(w) = 0$, for any distortion function. Thus the constrained l_1 solution and the PLI distribution must be equivalent.

By Theorem 3, for $\lambda \leq \Lambda$ the solution of the l_1 LIME minimization problem is the same as the solution to the constrained l_1 LIME problem.

Then by the transitive property, for $\lambda \leq \Lambda$, the solution of the l_1 LIME minimization problem must be equivalent to the PLI weight distribution.

4.4.3 Surfaces fit for regular grids

In this section it is proven that PLI fits a very simple functional form to test points within a hypercube. Recall that PLI assumes training points at the vertices of a rectangular polytope, and that without loss of generality for calculating the weights

the feature space can be scaled in each dimension so that the rectangular polytope becomes a unit hypercube. First, we introduce some notation, then the theorem and proof, then we show some examples for the two-dimensional case.

Theorem 11 *Let the d components of a test point $X \in [0,1]^d$ be members of a set $\Upsilon = \{X[1], X[2], \ldots, X[d]\}$. Consider the power set P(Υ), which is the set of all 2^d subsets of Υ (including the empty set). Let there be a lexicographical indexing of the subsets of P(Υ). Let Z_i to be the ordinary arithmetic product of the elements in the ith subset of P(Υ), where the product of the elements of the empty set is defined to be 1.*

Given any test point $X \in [0,1]^d$ and training samples (X_j, Y_j) for $j = 1$ to 2^d with $X_j \in 0, 1^d$, the surface estimated by PLI has the functional form,

$$f_{PLI}(X) = \sum_i a_i Z_i$$

where Z_i is as defined above and a_i is some linear combination of the training observations $\{Y_1, Y_2, \ldots, Y_k\}$.

Proof: Recall that by definition the PLI weight for the jth training sample X_j is

$$w_j(X) = \prod_{m=1}^{d} \|1 - X_j[m] - X[m]\|$$

Since the PLI weights do not depend on shifts, without loss of generality, shift the feature space so that training point X_1 is at the origin. Then the weight on X_1 is

$$w_1(X) = \prod_{m=1}^{d} \|1 - X[m]\|$$

Since $X \in [0,1]^d$,

$$w_1(X) = \prod_{m=1}^{d} (1 - X[m])$$

Expanding the product and re-writing in terms of the random variable Z yields,

$$w_1(X) = \sum_{i=1}^{2^d} b_i Z_i$$

where b_i are nonzero coefficients that depend on some subset of the components of X and Z_i are as defined in the theorem statement.

Note that the other weights can be expanded to yield similar expressions,

$$w_j(X) = \sum_{i=1}^{2^d} c_{ij} Z_i$$

where some of the coefficients c_{ij} may be zero.

The estimation algorithm then applies the weights to the training observations to form the estimate

$$\begin{aligned}
\hat{Y} &= \sum_j w_j Y_j \\
&= \sum_{i=1}^{2^d} b_i Z_i Y_1 + \sum_{j=2}^{2^d} \sum_{i=1}^{2^d} c_{ij} Z_i Y_j \\
&= \sum_{i=1}^{2^d} \left(b_i Y_1 + \sum_{j=2}^{2^d} c_{ij} Y_j \right) Z_i
\end{aligned}$$

and thus we can conclude that the surface fit has the functional form

$$f_{PLI}(X) = \sum_{i=1}^{2^d} a_i Z_i$$

where $a_i = b_i Y_1 + \sum_{j=2}^{2^d} c_{ij} Y_j$. q.e.d.

Example surfaces fit to a 2D rectangular grid

In this section some examples of surfaces fit to a unit square are given.

The training samples are pairs $(X_1, Y_1), (X_2, Y_2), (X_3, Y_3), (X_4, Y_4)$ with $X_1 = [0, 0], X_2 = [1, 0], X_3 = [0, 1], X_4 = [1, 1]$.

The test point is $X \in [0, 1]^2$.

Then, $Z_1 = 1, Z_2 = X[1], Z_3 = X[2], Z_4 = X[1]X[2]$.

The PLI weights are $w_1 = (1 - X[1])(1 - X[2])$, $w_2 = X[1](1 - X[2])$, $w_3 = (1 - X[1])X[2]$, and $w_4 = X[1]X[2]$.

Then the functional form fit by PLI is

$$f_{PLI}(X) = \sum_i a_i Z_i = a_1 + a_2 X[1] + a_3 X[2] + a_4 X[1]X[2]$$

However we know that the estimate at a point X is $\hat{Y} = \sum_j w_j(X) Y_j$. Substituting in for the weights and equating $f_{PLI}(X)$ and $\hat{Y}$, we can solve for a_1, a_2, a_3, and a_4 to find that,

$$f_{PLI}(X) = Y_1 + (Y_2 - Y_1)X[1] + (Y_3 - Y_1)X[2] + (Y_4 + Y_1 - Y_2 - Y_3)X[1]X[2]$$

In Figure 4.1, four examples are shown of surfaces fit given different training observations Y_1, Y_2, Y_3 and Y_4 for the corresponding vertices of the unit square.

4.4.4 LIME and PLI for interpolating grids

LIME or PLI may be useful for interpolating grids. PLI is a robust and computationally efficient way to interpolate points within a grid cell. PLI uses all the grid vertices, so no searching is needed for a convex hull or near neighbors. The PLI weights are available in closed form, as shown in Section 4.4.1. PLI will also give robust estimates for nonlinear data since all the test points are used, and using more test points means more robustness to additive noise. LIME may offer performance advantages by relaxing the constraint $D(w) = 0$, and by allowing smaller, more local sets of training samples to be used. For example, a ten-dimensional grid has 1024 vertices to each cell. Traditional linear interpolation would use 11 vertices which form a convex hull around the test point. PLI uses all 1024 vertices. LIME allows any number of vertices to be used. The LIME computation time may be prohibitive, or for large grids with small cells, it may be possible to compute all the weights ahead of time and apply them as appropriate.

An experiment for color management grid interpolation compares tetrahedral interpolation, LIME, and PLI(trilinear). In Section 4.4.6 a simulation shows varying

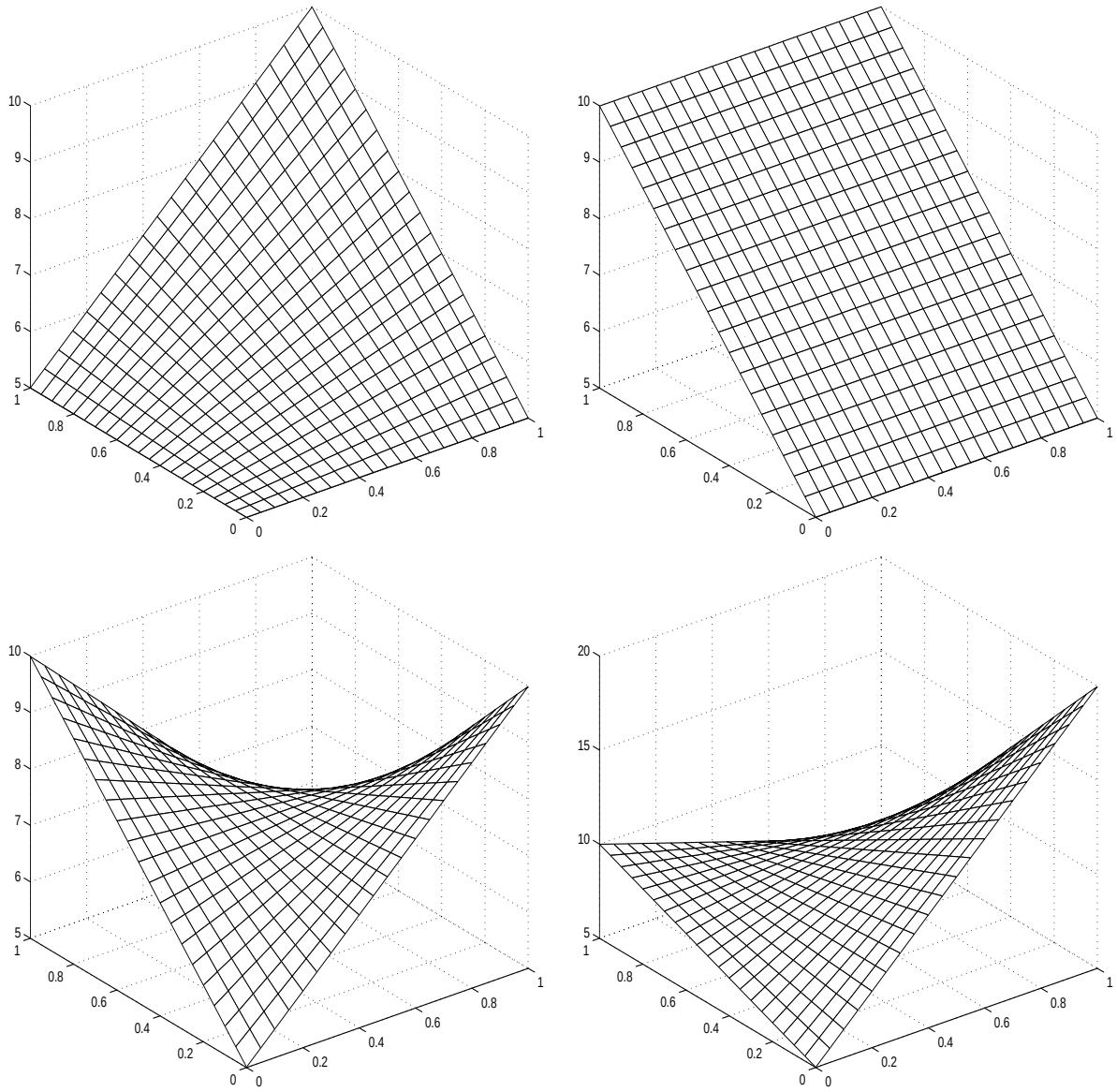

Figure 4.1: Example surfaces fit by PLI. Top left: training observations are 5,5,10,5. Top right: training observations are 5,10, 10,5. Bottom left: training observations are 5, 10, 5, 10. Bottom right: training observations are 5, 10, 5, 20.

robustness to additive noise. We end with a function approximation over a ten-dimensional grid cell.

4.4.5 Color grid experiment

We used a data set, courtesy of Dr. Gabriel Marcu of Apple Computer, of 12 × 12 × 12 almost-uniformly sampled data points spanning RGB space sent to an ink jet printer and printed on plain paper. The CIELAB values of the resultant printer color patches were then measured under a D50 illuminant. With the exception of the 8 corners of the RGB color cube, we ran a cross-validation experiment where each point was removed from the set in turn and estimated from the remaining sample points.

We implemented LIME defining the neighborhood based on the number of closest points to the target point. The LIME implementation used l_2 distortion and is further described in Section 2.11. We experimented with using five through eight nearest neighbors. The parameter λ was set very low to .001 to make a closer comparison to tetrahedral interpolation and PLI. Tetrahedral interpolation used the convex hull made of the four nearest neighbors to form a convex hull. PLI (trilinear interpolation) uses all 8 vertices.

A standard measure for color error is the l_2 norm in CIELAB space, which we refer to as the 'error length.' However, in a visual application such as color transformations, it is important that the maximum error over the color planes be small. Hence we also consider the l_∞ norm in CIELAB space.

Table 4.2 shows the mean error length and the variance of the error lengths over the 1720 cross-validated test set. The mean tetrahedral performance is 18% worse than using LIME with five or six nearest neighbors. Important in this application is that the variance of the LIME errors is almost half the variance of the tetrahedral errors, limiting the probability of disturbingly large errors.

Table 4.3 shows the average maximum (over L^*, a^*, b^*) error and the variance of the maximum error. The tetrahedral average maximum error is 19% worse than the

	Mean Error Length	Var
tetrahedral	1.50	1.92
LIME w/ 5 nearest	1.28	1.16
LIME w/ 6 nearest	1.27	1.03
LIME w/ 7 nearest	1.34	1.09
PLI	1.38	1.14

Table 4.2: Mean and variance of CIELAB error lengths

	Mean l_∞ Error	Var
tetrahedral	1.29	1.50
LIME w/ 5 nearest	1.08	.86
LIME w/ 6 nearest	1.08	.77
LIME w/ 7 nearest	1.13	.81
PLI	1.16	.86

Table 4.3: Mean and variance of CIELAB l_∞ errors

LIME with five or six nearest neighbors. The variance of the tetrahedral interpolation is almost twice as large, resulting in more large errors with tetrahedral interpolation.

In general, the LIME algorithm is more accurate than tetrahedral interpolation when relatively large errors occur. This is because LIME is using more information over the spatial domain, and if the colorspace is changing quickly tetrahedral interpolation may not be able to capture the change due to its limited use of the sample data.

4.4.6 Simulation with additive noise

In practice, noise is always a problem. Theoretical results showing noise robustness were presented in Section 3.4. In this section, a numerical simulation compares LIME's robustness to additive noise with least squares fitting for a hyperplane. In the simulation, the training features are the 8 vertices of a three dimensional unit cube. The corresponding observations are a linear function of the three dimensional features: $Y = AX$, where $A = [1\ 1\ 1]$. The 1000 test samples were sampled uniformly throughout the cube. The simulation was run with ten different amounts of

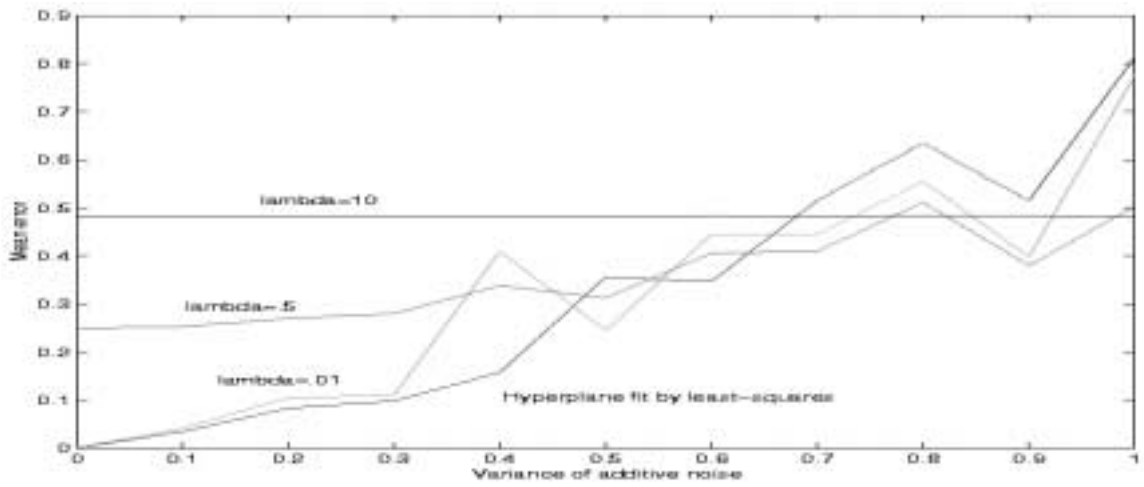

Figure 4.2: Simulation of linear surface with additive noise

independent additive white noise added to the training features, $\tilde{X} = X + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, and σ varied from 0 to 1. The corresponding observations were not infected by noise, $Y_j = AX_j$ for $j = 1$ to 8. Estimations were produced by a least-squares fit of a hyperplane to the data, and by LIME with three different values for λ, .01, .5, and 10. The results were measures in terms of mean squared error and plotted in Figure 4.2. Note that no noise was ever added to any test point or training observation.

Theoretically, one expects noise on the training features to create noisy estimates that are equivalent to the clean estimates for some point equal to the test point plus the noise. The simulation bears out these expectations. In particular, the LIME estimates at high λ weight all of the training points equally, no matter where the test point is. Thus additive noise on the features does not change the weighting, and hence the error for high λ stays constant at around .5 despite increasing additive noise.

The least squares plane fit is the correct model for the underlying relation between features and observations in this simulation. However, at a high level of added noise

	d=5 dimensions	d=10 dimensions
LIME w/ d+1 nearest neighbors	.0656	.0732
LIME w/ minimum convex hull	.0445	.0220
PLI	.0443	.0220
Linear regression over grid cell	.0519	.0251
Ridge regression over grid cell, k= 3.5(best k)	.0471	.0247

Table 4.4: Mean absolute value errors for approximating the square root of the sum of the feature dimensions

even using the correct model does not perform as well as LIME with a relatively large λ (or equivalently in the limit as $\lambda \to \infty$, k-NN).

4.4.7 Functional approximation over a grid cell

Simulations were run to compare using LIME with the nearest $d+1$ neighbors (which may or may not form a convex hull), LIME with the nearest k neighbors such that the test point is contained with the convex hull of the k neighbors, and PLI (2^d training points). For the two variations of LIME, $\lambda = .00001$. Traditional linear interpolation with the smallest convex hull would be an additionally interesting point to compare with, but it was prohibitively complex to compute the correct training points.

There were 5000 test points for the five-dimensional problem and 1000 test points for the ten-dimensional problem. The average k to achieve a convex hull was 9.9 nearest neighbors for five dimensions and 59.51 nearest neighbors for ten dimensions.

For ridge regression (over the total grid cell), an additional 1000 test points was used to train the ridge regression smoothness parameter.

The first function to approximate is

$$f(x) = \left(\sum_{m=1}^{d} x[m] \right)^{.5}.$$

The results are shown in Table 4.4. The maximum of the observations $f(x)$ was 2.7894, and the minimum was 1.3134. The mean was 2.2283. The 10-dimensional PLI approximations were on average only 1% from the truth. For both 5 and 10 dimensions, PLI performs around 10% better than ridge regression.

	d=5 dimensions	d=10 dimensions
LIME w/ d+1 nearest neighbors	.0559	.0569
LIME w/ minimum convex hull	.0420	.0271
PLI	.0424	.0276
Linear regression over grid cell	.0484	.0304
Ridge regression over grid cell	.0469	.0303

Table 4.5: Mean absolute value errors for approximating the log of the sum of the feature dimensions

For the next functional approximation, the function to approximate is

$$f(x) = \log\left(1 + \sum_{m=1}^{d} x[m]\right).$$

The observations $f(x)$ ranged from 1.2157 to 2.1478, with a mean of 1.7799. As shown in Table 4.5, ten-dimensional PLI approximations were on average within 2% of the truth. Again, the PLI estimates are around 10% better than ridge regression.

These results show that using LIME or PLI, good results for grid interpolations can be obtained without searching for a minimum convex hull.

Chapter 5

More experiments and simulations

Data is always a powerful
impetus for making decisions.

Richard Moran

The 'No Free Lunch' theorem [31] of machine learning states that when classifying or estimating the unknown, no one method will always work best. Certainly though, some methods work better than others. In this chapter we consider a few more experiments and simulations to evaluate and understand the performance of the LIME algorithm. First, there is a Gauss mixture simulation that looks at rate of convergence. Then we present a start-to-finish classification application for pipeline integrity verification [94]. There are also two standard datasets included, a vowel recognition dataset and a medical diagnosis dataset.

The LIME algorithm performs competitively throughout, and often makes rather different decisions than other comparable algorithms. Hybrid classification systems which use multiple classifiers have been shown to be useful [100], [71], [80], [46], [95], and LIME may be able to add to performance when used in conjunction with other classifiers.

5.1 Rate of convergence

In practice, the actual rate of convergence as a function of training data is important. Practitioners are often interested in the most 'bang for the buck' in terms of an algorithm that provides the lowest error rates for a given amount of training data. Further, the expected marginal value of obtaining more training data may help determine how much more training data is obtained. The rate of convergence may provide insight into these issues. A number of authors have looked at rates of convergence based on large-deviations theory or other techniques; the book [30] provides a good review of results. Some research has gone into understanding how sample size affects accuracy, including [103] and [104]. For the nearest neighbor algorithm, Flunking and Hummers [36] decompose the deviation from the asymptotic error into functions of sample size, metric, dimensionality, and underlying distribution.

In Section 2.7 we compared rates of convergence on a two class Kohonen simulation. The next Gauss mixture simulation also compares rates of convergence.

Consider a two class problem in ten dimensions. Each class is equally likely and each class is a mixture of five equally likely Gaussians. For each dimension of each Gaussian, its mean μ and its standard deviation σ were independently from a uniform distribution on [0,1]. Training and test points were independently drawn from the Gauss mixture distribution. In Figure 5.1, classification accuracy is plotted against an increasing number of training samples, from 50 to 150,000 samples. The classification accuracy is the empirical accuracy of the same two thousand test points classified with the increasing number of training samples. Cost of classification error was one for both cases.

For each training sample size n, the size k of the neighborhood was optimized for each algorithm independently. The optimization for the neighborhood size k was done using the n training samples and minimizing the empirical error on an additional 2000 points sampled from the Gauss mixture. For LIME, the λ parameter was optimized to be $\lambda = .5$ by minimizing the empirical error on the 2000 points with the neighborhood size was set at $k = 50$ and using 1000 training samples. The parameter λ was held constant at .5 throughout the simulation. Thus, only the parameter k was tuned for

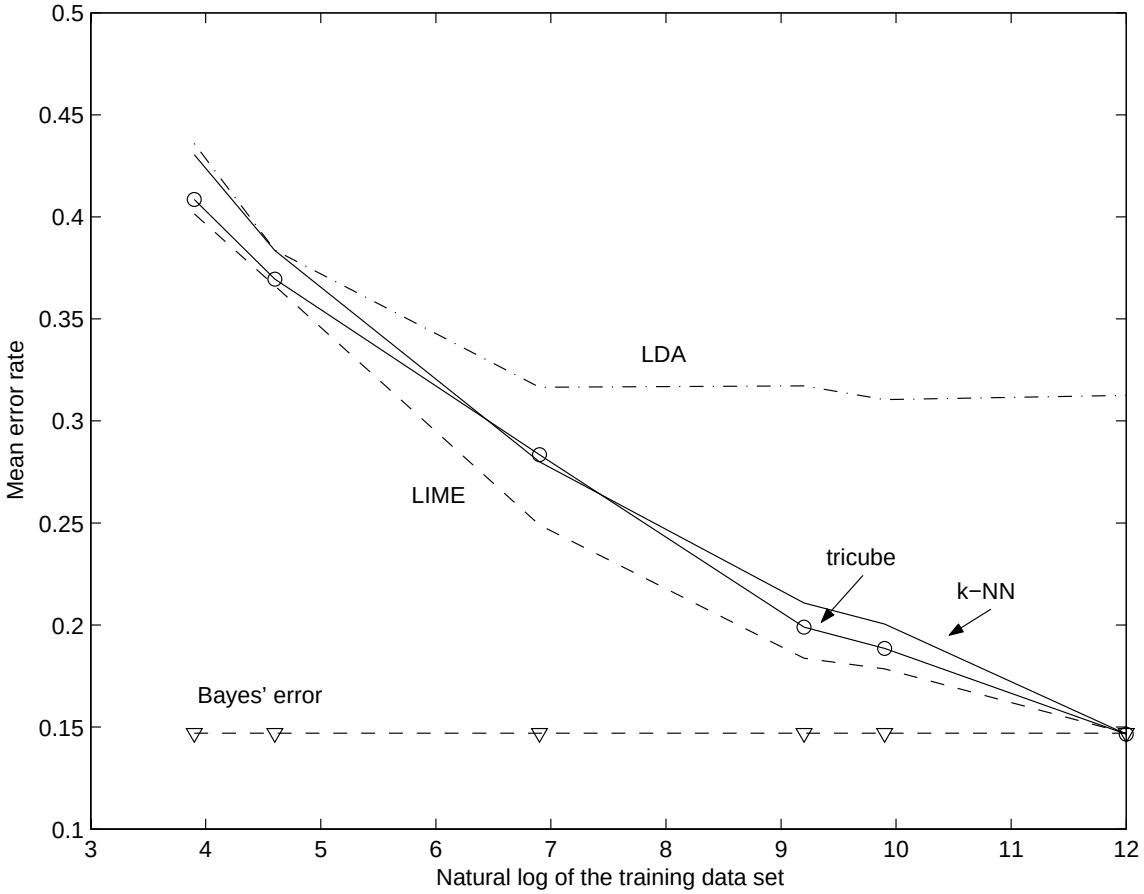

Figure 5.1: Gauss mixture simulation with increasing training data

each algorithm as n changed.

5.2 Pipeline damage detection

Natural gas pipelines in the North Sea and on land in continental Europe are aging and are beginning to show signs of corrosion and decay. Monitoring the pipeline integrity can avert costly leaks and speed repairs in the event of an accident. Currently, the primary technology for inspecting these pipelines is magnetic flux detectors [6]. Such inspections are expensive, difficult, and at times inaccurate and dangerous.

A new optical inspection technology has been developed by Norsk Elektro Optikk (NEO) which shuttles a laser camera through the pipelines storing images of the inner walls of the entire pipeline. It is inefficient to manually analyze the resulting kilometers of data. A goal is to develop an automatic system for analyzing the images and identifying anomalous events, thus providing information on the overall pipeline integrity and areas of the pipeline in need of repair.

Normal	43
Osmosis blisters	20
Scratches	14
Corrosion dots	17
Vertical welds	20
Weld cavity	19
Welds Too Close	16
Weld	20
Grinder marks	20
MFL marks	13
Corrosion blisters	11
Single dots	15

Table 5.1: Relative frequencies of the twelve pipeline event classes

We designed an image processing and classification method to identify abnormal events. Non-overlapping image blocks are classified into twelve different categories: normal, black line, grinder marks, magnetic flux leakage inspector marks, single dots, small black corrosion blisters, osmosis blisters, corrosion dots, longitudinal welds, field joint, cavity at a weld and longitudinal weld too close to field joints. Results compare different types of statistical classifiers. The features extracted from the pipeline image are designed to mimic the features humans use to identify the different classes. Difficulties include the large number of classes, the uneven costs associated with different errors, and training on a limited amount of expert classified data. The classification results show that this can be a useful tool for pipeline monitoring.

This project was joint work with Deirdre O'Brien, and the features, results, and details reported are due in part to her. NEO engineers were, of course, instrumental in providing data, expertise, and financial support. Further information on the project can be found in [94].

The original dataset is kilometers of JPEG compressed image data. These are broken into 96×128 pixel images, each representing sections of pipeline approximately 96 mm $\times$ 128 mm. Examples of the images are shown in Figure 5.2, the background vertical stripes in the images are due primarily to variations on the laser intensity across the line imaged by the camera.

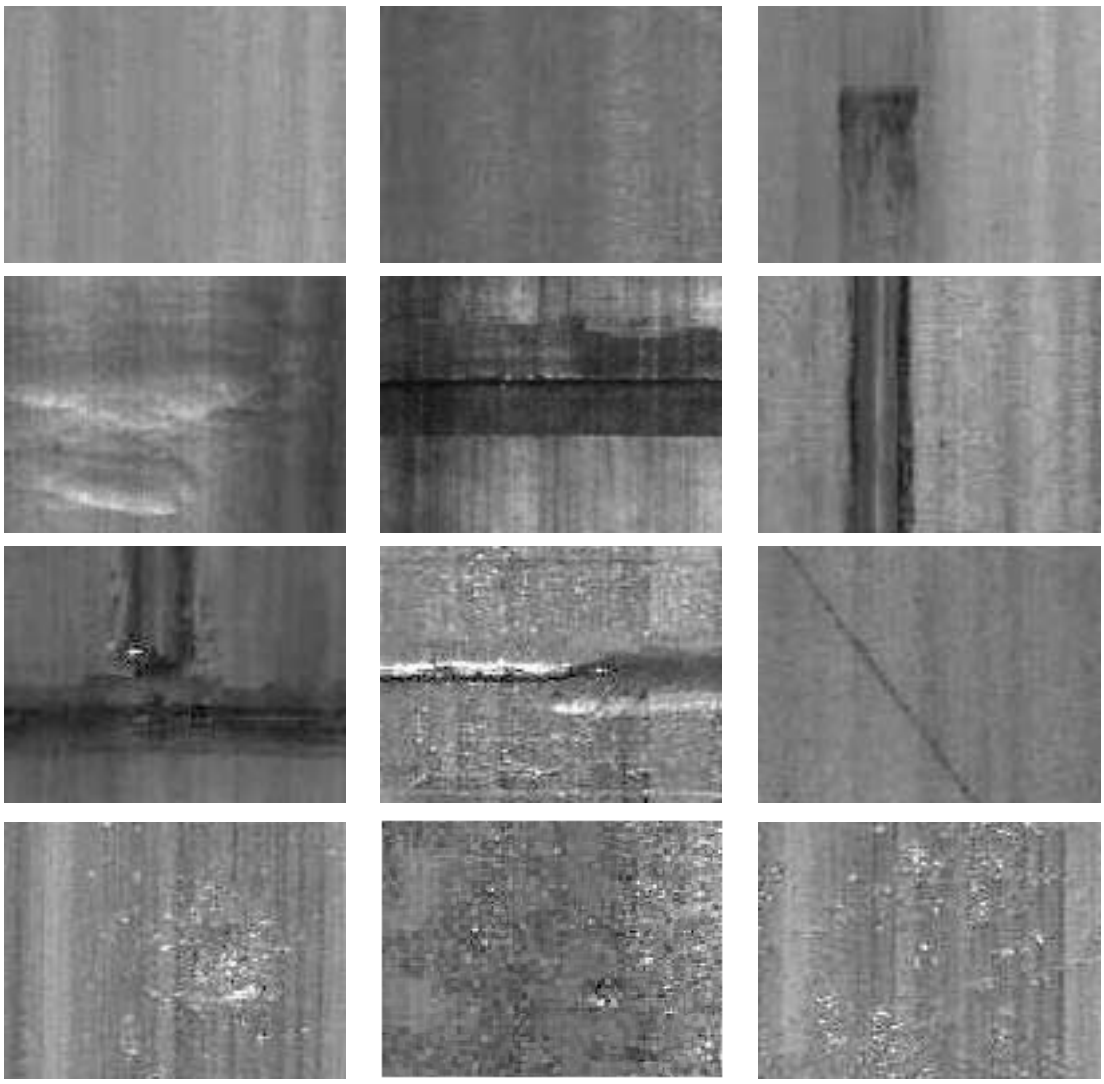

Figure 5.2: Example 96 × 128 pipeline images. Left to right, top to bottom: normal, normal, MFL mark, grinder mark, field joint, longitudinal weld, welds too close, weld cavity, black line, single dot, corrosion blisters, osmosis blisters.

A hand-labelled database was created of twelve classes that were of interest to NEO. As shown in Table 5.1, the database contains an uneven spread of 228 images, ranging from 11 corrosion blister images to 43 normal images. For many events, the table represents the maximum number of examples of those events found.

5.2.1 Features

Features based on raw pixel values, DCT coefficients, and wavelet coefficients for 8 × 8 blocks were explored early in the project, but did not capture enough information to achieve useful classification results. The events of interest generally occupied areas significantly larger than 8 × 8 blocks. Also, the artifacts due to laser variation and those resulting from the JPEG compression detracted from the information that could be extracted using this small blocksize.

Nonlinear, statistical, and morphological features were found to yield better differentiation between classes. Designing specific features for this particular application allowed prior knowledge of the artifact appearance to be incorporated into the features. This reduced the expected overfitting due to the small dataset.

The features used fall into two categories – those designed using the constrast feature of gray-level co-occurrence matrices (GLCM) [47] and those which we term Human Visual Discriminant (HVD) features.

Six GLCM contrast features were designed. First, local contrast was measured as the squared differences of pixels vertically separated by 4 mm or 8 mm, averaged over 4 × 4 mm^2 image blocks. Then, regions of high contrast were identified, such that the local contrast is greater than a constant multiple of the image variance. Six features were extracted which describe the total area and number of these connected high contrast areas as well as the size of the largest such area.

The 16 HVD features seek to mimic the features used in manually distinguishing between classes. The first feature measured the standard deviation of image pixels. Nine features were extracted which capture information about the size, shape and mean graylevel of relatively dark or bright areas. Dark and bright pixels were identified as those with graylevels of a number of standard deviations from the image

mean.

Other features were designed to specifically differentiate certain classes. To recognize blisters, a feature calculates the number of 8×8 blocks within the image with a range larger than a given threshold. To identify horizontal artifacts such as field joints, the average difference between the graylevel averages of each side of a horizontal boundary that sweeps down the image was included. To help find corrosion dots, one feature counts how many 4×4 blocks have means that are significantly darker than the column mean. Other features based on vertical strips through the image measure the mean of the 4 mm wide column standard deviations and the standard deviation of the mean of 2 mm wide columns as well as the maximum value of the image mean minus the mean of a 2 mm wide column. These final features are used to identify vertical artifacts such as longitudinal welds.

Each of these features was normalized to have unit variance. Unit variance feature scaling may not be the optimum relative scaling for the features. It is likely that certain features are better discriminants while others are noisy or less informative. One way to generate improved feature scalings is to use the feature importance metric incorporated in the CART (classification and regression trees [14]) algorithm. CART seeks to build a tree that minimizes the misclassification error. The importance of a particular feature can be measured by how closely a tree built using only that feature matches the best tree found by CART. Scaling the univariate features by CART's feature importance measures significantly improved the classification performance.

5.2.2 Classification algorithms compared

Four classifiers were compared, linear discriminant analysis (LDA) [54], regularized quadratic discriminant analysis (QDA) [54], multiple additive regression trees (MART) [54], and LIME. MART is a boosted version of a classification tree [1]. To incorporate a cost in LDA and regularized QDA, the methods were considered as model fitting algorithms. LDA fits a Gaussian to each class with each class having the same covariance, regularized QDA allows this covariance to be class dependent.

[1]MART was implemented using code available at http://www-stat.stanford.edu/ jhf/

Using this model, the probability that a test sample belonged X to class j was estimated, $\hat{P}_j(X)$. Given a cost matrix C, where $C(i,j)$ is the cost of assigning a sample to class j when it belongs to class i, the assigned class was then

$$\arg\min_{\hat{y}} \sum_j P_j C(\hat{y}, j)$$

Each image was classified into one of the twelve classes by each classification algorithm. Due to the small ratio of labelled data (228 images) to number of classes (12), leave-one-out cross-validation was chosen to compare classification algorithms. For each sample X, any classifier parameters (including the feature scalings calculated from CART) were estimated based on the other 227 sample points and the estimated class of X was determined using these parameters. The results shown in Table 5.2 and Table 5.3 are the average performance over all 228 images.

Images from the pipeline are overwhelmingly 'normal'. The hand-labelled database is more evenly distributed over the classes. Using either density as a prior resulted in classifications skewed towards normal. The error rates reported in this paper are based instead on a uniform prior.

The consequences of different class mislabellings vary significantly. Failing to detect cavities may seriously compromise the future integrity of pipeline, however the ramifications of confusing osmosis and corrosion blisters are much less significant. The misclassification costs were estimated by the researchers at NEO and this cost matrix was used in all classifiers except for the column titled LIME (0-1 cost) which was run with equal (0-1) costs for all classes to highlight the effect of NEO's cost matrix. Misclassifying images of classes A (normal), E (longitudinal welds), H (field joints), I (grinder marks) and J (MLF marks) generally had small costs, whereas misclassifying images of classes F (weld cavity) and G (welds too close) generally had high costs. For the other classes (which we call medium cost classes) the cost varies significantly dependent on how the image is misclassified.

In Table 5.2 the average costs per class (calculated using NEO's cost matrix) are listed for each algorithm. This information tells us that if there is an event of a particular class, the average cost is how much cost we expect to incur by the algorithm's estimation. Given the small dataset and misclassification cost ranging

from 5 to 600, a small number of serious errors causes a significant increase in average cost.

The different algorithms perform quite differently. MART and LIME were more strongly influenced by the cost matrix than the Gaussian methods (LDA and regularized QDA). In the low cost classes, listed above, LDA, regularized QDA and LIME (0-1 cost) generally perform better than MART and LIME. Conversely, for the medium and high cost classes MART and LIME outperformed, often significantly, the other methods. On average, the performance of the LIME is better than that of regularized QDA but for a particular task or goal, one classification algorithm may be more appropriate than another. Lower error rates may be obtainable with a hybrid classifier incorporating the opinions of a number of different classification methods [100].

	MART	LDA	Reg. QDA	LIME	LIME (0-1 cost)
A	20.93	0.23	0.70	9.65	0.81
B	11.50	14.75	5.00	5.75	0.00
C	16.79	28.57	37.50	11.79	37.14
D	6.18	117.65	52.94	29.41	47.06
E	12.00	2.50	0.00	1.50	1.25
F	7.89	99.47	17.37	9.47	15.79
G	37.50	161.88	50.00	25.00	74.69
H	27.50	0.00	2.50	25.00	5.00
I	32.00	0.75	0.25	20.75	1.25
J	23.85	3.08	1.54	20.77	1.92
K	20.45	27.27	72.73	20.91	18.18
L	18.00	78.00	41.33	6.00	46.67
Mean cost	19.55	44.51	23.49	15.50	20.81

Table 5.2: Mean expected cost for an event of a given class

In Table 5.3 the recall per class is listed for each algorithm. Recall for class Y is the number of images that belong to class Y that were correctly labelled. This table clearly shows the recall dropping for low cost events when NEO's cost matrix is used in the LIME estimation instead of LIME with 0-1 costs. LIME classification

using 0-1 cost has higher recall than LIME on all but two events (and significantly higher average recall). The misclassifications by LIME with 0-1 cost were expensive according to the NEO cost matrix, however. Both MART and LIME misclassify all samples from class J, but given the low cost of these misclassifications these algorithms still maintain a low average cost.

	MART	LDA	Reg. QDA	LIME	LIME (0-1 cost)
A	0.05	0.98	0.91	0.44	0.91
B	0.40	0.90	0.95	0.65	1.00
C	0.07	0.64	0.50	0.21	0.50
D	0.76	0.53	0.82	0.88	0.82
E	0.60	0.90	1.00	0.85	0.95
F	0.74	0.74	0.89	0.68	0.95
G	0.75	0.63	0.69	0.75	0.75
H	0.45	1.00	0.95	0.50	0.90
I	0.10	0.90	0.95	0.15	0.85
J	0.00	0.69	0.85	0.00	0.77
K	0.45	0.82	0.36	0.73	0.82
L	0.07	0.00	0.33	0.47	0.33
Mean Recall	0.37	0.73	0.77	0.53	0.80

Table 5.3: Mean recall for an event of a given class

More data of critical events such as welds too close could alter the balance of the results. More useful results might also be had by changing the problem from a twelve class problem to a 'normal' vs. 'abnormal' problem. The abnormal images could be fed to a human discriminator.

Due to the different abilities of each classifier, a majority rule or class-expertise hybrid classifier might achieve lower expected costs. Other researchers have found combining classifiers to be useful [71], [80], [46], [95]. For example, such a system might use Reg. QDA, LIME (0-1 cost) and LIME. If two of the classifiers agree to the class then the majority class is chosen. If no two classifiers agreed, then the LIME result could be used (with the assumption that although it might be wrong, it will provide on average the least costly misclassification).

5.3 Vowel data set

The next experiment is a vowel classification test based on a benchmark dataset available from the Information and Computer Science Department at the University of California, Irvine [12]. The training data consists of 528 data points from eight mixed-gender speakers saying eleven different words and six data points taken from the steady-state vowel of each word. The test data is composed of 462 data points from seven speakers. The eleven words (classes) are the steady-state vowels of British English: hid, hId, hEd, hAd, hYd, had, hOd, hod, hUd, hud, hed.

Classification method	Error rate
Single-layer Neural Network	67%
Linear Discriminant Analysis	54%
Multi-layer Neural Network	49%
CART (decision tree)	46%
K-Nearest Neighbor	44%
Flexible Discriminant Analysis	39%
LIME	38%
Equal weight on all points in LIME neighborhood	39%

Table 5.4: Error rates for classifiers on the vowel dataset.

The speech signals were low pass filtered at 4.7 kHz and then passed through a 12 bit ADC with a 10 kHz sampling rate. Six 512 sample Hamming windowed segments were taken from the steady part of each word's vowel and then analyzed with twelfth order linear predictive analysis. Reflection coefficients were used to calculate ten log area parameters which are entered as a ten dimensional data point to the classifier.

For each test point, its neighborhood was defined as the subset of the training data that falls within a radius of

$$(1+\alpha)\left(\min_j \|X_j - X\|\right),$$

where the parameter α was chosen to provide the best classification rate on the training data via cross-validation. The cross-validation was eight-fold with each speaker removed from the training set in turn and the remaining data used as the test set. We conjecture that using an adaptive α neighborhood may be more appropriate for

datasets like this on in which over 80% of the test points fall outside the convex hull of the training data.

For the results described in Table 5.4, $\alpha = .24$ and λ was found to be 10. The results for the other algorithms come from [54], and exact implementation details are unknown.

LIME with $\lambda = 10$ heavily emphasizes maximizing the entropy with little regard for the distortion. In the table we include the results for equal weightings over the LIME neighborhood, and find the results are very close. Then the low classification error of LIME on this test set is very much a function of neighborhood.

5.4 Pima Indians and diabetes

The Pima Indian diabetes dataset is another benchmark dataset available from the Information and Computer Science Department at the University of California, Irvine [12]. The Pima Indian dataset has 768 instances of eight features (none missing) and two categories denoting whether each person had diabetes or not. All patients were females at least 21 years old and of Pima Indian heritage. The eight features are:

1. Number of times pregnant
2. Plasma glucose concentration at 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)

An expert in diabetes might have some insight into how to scale the features with respect to each other. Or, one might try cross-validation on a training set to scale the features. However, the object of considering this dataset is to show the rough-and-ready application of LIME. Each feature was modelled by a Gaussian distribution with the data's mean μ and standard deviation σ of each feature calculated over the

entire data set, and scaled each feature accordingly:

$$x_i'[m] = (x_i[m] - \mu[m]) / \sigma[m]$$

A full-blown machine learning study was done in 1988 for this dataset [117]. The researchers developed a neural-net style algorithm called ADAP, specifically for this feature set.

The first 576 instances formed the training set, the last 192 instances the test set. In the training set there were 198 positive cases (34.38 %) and in the test set there were 70 positive cases (36.46%).

A two-dimensional joint search was performed for the λ and k nearest-neighbors neighborhood parameters using leave-one-out cross-validation for LIME. We also used leave-one-out cross-validation to choose the k parameter for the k-NN algorithm and for determining the extent of the tricube neighborhood (k nearest-neighbors neighborhood). All algorithms (except ADAP) were evaluated on the normalized features. For LIME, cross-validation gave a maximum performance on the training set of 72.22% for $\lambda = 1$ and $k = 34$. Training set cross-validation for k-NN yielded a maximum accuracy of 71.70% for $k = 35$. The tricube kernel yielded maximum accuracy of 69.27% at $k = 24$ on the training data.

The cost of misdiagnosis was considered equal for false positive and false negative. However, as in many medical experiments, the community may feel that different costs are more appropriate.

Classification method	Error rate
Linear Discriminant Analysis	30%
K-Nearest Neighbor	34%
1-Nearest Neighbor	29%
Tricube Kernel	23%
ADAP	24%
LIME	21%

Table 5.5: Comparison of classifiers on the Pima Indian Diabetes dataset

Chapter 6

Principles of inference

As a rule, probable inference is more like measuring smoke than counting children...

Richard Cox

A practical engineering problem is to assign a pmf q over mutually exclusive events without enough information. There may be a prior estimate p for q, or some information about q in the form of constraints or known moments. For example, in the LIME algorithm we seek a probability distribution over the training samples that minimizes distortion; this constraint may lead to non-unique solutions and we use the principle of maximum entropy (or minimum relative entropy) to select one solution. The general problem of estimating a distribution q given prior information or a model does not have a unique solution, and requires extra-situational principles to guide us. Some thinkers claim that the problem is in certain cases unresolvable. In practice, one can judge inductive estimates by their probabilistic success or behavior in limiting cases.

In this chapter, we first review the simple but controversial principle of insufficient reason. Next comes the more general principle of minimum relative entropy, of which the principle of maximum entropy is a special case. Lastly, we consider a 'principle of minimum expected risk,' which may yield robust estimates.

Some researchers have been concerned that assigning probabilities to events without full knowledge is undefined or ill-advised. Cox balks [26], *'As a rule, probable inference is more like measuring smoke than counting children, in that the probabilities themselves are not well-defined.'* Lucas asks [83], *'How can one base any conclusions on ignorance?'*

The perspective taken here is that the estimate q is a current best working guess, guided by principles of inference. Probability distributions may be profitably used to represent our uncertain understanding of a situation without making any claim to truly represent an underlying random variable. In this work, we consider a probability distribution assigned to events to represent the relative bets that one would make; this view is fairly traditional in information theory [25], and acceptable given that so much of the practical application of probability is by gamblers, investors, and insurers. From this viewpoint, those thinkers who claim that it is not possible to know the true distribution (unarguable) or that the distribution may be, in some sense ill-defined, are simply not very sporting.

6.1 Principle of insufficient reason

The oldest principle of inference is the principle of insufficient reason, which suggests that one assign equal probabilities to events that, to one's knowledge, are equal:

If k mutually exclusive and exhaustive events are possible, but there is no evidence to expect one event more than another, then the a priori probability distribution should represent the symmetry of ignorance, and the a priori probability should be 1/k for each event.

Keynes [73] called it the principle of indifference. Lucas [83] refers to it as the principle of equiprobability, and argues for its validity, *'The principle of equiprobability is not a principle but a presumption; not a principle of indifference, but a demand that differences, if there are any, shall be accounted for.'*

Although Lucas's defense of the principle is cogent, the difficulty of applying the principle has led to controversy and paradoxes.

The principle of insufficient reason is sometimes attributed to Laplace, or cited as a

Laplacian viewpoint. However, in Laplace's *A Philosophical Essay on Probability* [81], he lays out ten general principles of probability without explicitly stating a principle of equiprobability. In fact, he seems to take the idea for granted, clearly applying it throughout the work in examples, and early in the work (page 7 of the fifth edition) Laplace states the notion as a key part of his theory of chance:

La théorie des hasards consiste à réduire tous les évènements du même gare, à un certain nombre de cas également possibles, c'est-à-dire, tels que nous soyons également indécis sur leur existence; et à déterminer le nombre de cas favorables à l'evenement dont on cherche la probabilité.

An almost literal translation was done in 1902 [122]:

The theory of chance consists in reducing all the events of the same kind to a certain number of cases equally possible, that is to say, to such as we may be equally undecided about in regard to their existence, and in determining the number of cases favorable to the event whose probability is sought.

The relevant phrase here is, *'de cas egalement possibles, c'est-a-dire, tels que nous soyons egalement indecis sur leur existence'*. In English, *'of cases equally possible, that is to say, to such as we may be equally undecided about in regard to their existence'*.

Thus Laplace equates *equally possible events* and *events about which we are equally undecided*. Equal indecision implies equal probability, and equal probability represents equal indecision.

The principle of insufficient reason has caused a fair amount of controversy over the years, mostly due to the difficulty in defining 'equal events' for a given problem. For example, a famous reasoning against the principle of insufficient reason comes from Carnap [19]. Suppose there is an urn with an unknown number of blue, red, and yellow balls. What is the probability of drawing a blue ball? Carnap presents a paradox, should the probability of drawing a blue ball should be equal to the probability of not drawing a blue ball, thus the probability equals 1/2? Then the probability of drawing a red ball or yellow ball can be similarly reasoned to be 1/2. Clearly, this reasoning ends in a contradiction.

However, since there is knowledge about three colors of balls, 'blue' and 'not blue' are not the most equal events known. Equal colors are more equal events and should

get equal probabilities, resulting in a probability of 1/3 for each color. Properly enumerating the sample space should decrease the risk of misapplication.

For continuous sample spaces, the continuous analogue of the principle of insufficient reason suffers also a lack of invariance to nonlinear transformations. Again, defining the 'equal events' clearly should dispel most confusion. A classic example of the need for well-defined equal events is Bertrand's paradox (a recent presentation is found in [31]. In the paradox, a chord is drawn at random in a circle, and the question is *'What is the probability that the chord is longer than a side of the inscribed equilateral triangle?'* Attempting to solve the problem by applying the equiprobability principle it quickly becomes clear that the problem is not well-defined enough to use the principle; one must know or guess how the chord is drawn at random, whether a midpoint is chosen randomly, or the ends are chosen randomly, etc.

For continuous distributions, a uniform prior may be nonsensical. Consider, for example, a parametric distribution with an unknown parameter $\theta \in \mathcal{R}$. Jeffreys' proposed [63] that one construct a prior which is flat for a function $\phi(\theta)$ whose Fisher Information is constant. This leads to a prior distribution for θ proportional to I_θ^5, where I is the Fisher Information. The prior for θ then depends on the parameterized form of the distribution being estimated.

In conclusion, to apply the principle of insufficient reason one must consciously and carefully define events considered equal.

6.2 Principle of minimum relative entropy

The principle of minimum relative entropy was introduced by Kullback [79] and states that, given a prior p over a set of events $x \in \Omega$, one should choose the pmf that minimizes the relative information

$$\mathcal{D}(q\|p) = \sum_{x\in\Omega} q(x)\log(\frac{q(x)}{p(x)})$$

over all feasible pmfs q such that any other constraints concerning q are satisfied.

A prior p with probablity 0 over any event will lead to difficulties, this can be

avoided by not using priors that have zero probability for any event. In cases where there is no prior p, the prior is often assumed to be uniform, in which case the principle is called the principle of maximum entropy. Consideration of distributions with maximum entropy for some class or some constraint reportedly has its roots in physics as early as 1937 [113], but it was Jaynes who in 1957 proposed maximizing entropy as a principle for solving inference problems. The rationale for the principle of maximum entropy springs from the principle of insufficient reason, that events should be treated equally if there is no information. When some information is available (e.g. a moment constraint) then the principle of maximum entropy recommends that events should be treated as equally as possible after considering the given information.

The principles of minimum relative entropy and maximum entropy have a number of nice properties, explored in such works as [79], [25], [114], [61], [127]. For coding applications, the principle can be interpreted as minimizing the extra bits needed to code a source p given an optimal lossless code for a source q.

The minimum relative entropy solution is the exponential solution with maximum likelihood [79](pg. 94). Moreover, the maximum entropy solution is the maximum likelihood solution over all distribution families if it is assumed that no event is known to be more probable than any another event [127](see the Appendix for a sketch of this result).

Uniqueness of the solution

Maximizing entropy or minimizing relative entropy over a closed compact set is a convex optimization problem and thus yields a unique solution.

However, the solutions are not unique in the sense that the maximization (or minimization) of other measures may result in the same distribution. In fact, it has been proven that the 'generalized entropies' proposed by Havrda-Charvat, Renyi, Behara-Chawla, and Sharma-Mittal, are equivalent to Shannon's entropy in terms of the probabilistic distribution obtained by maximizing those measures under a given set of fairly general constraints[67].

Likewise, there are cases where the minimum variance or least-norm solution will be equivalent to the maximum entropy solution.

Not a panacea

The principle of minimum relative entropy may not always lead to the best estimate of q. The principle assumes that there is one prior or model that one has significant faith in. What if there is more than one model one thinks likely? Or what if there is a prior, but the new evidence is much more important?

The strength of the principle of minimum relative-entropy is precariously dependent on the strength of the prior, yet provides no means to mathematically communicate the amount of faith we have in our prior. This defect is also pointed out by Jaynes in his unfinished work [62].

The principle or minimum relative entropy has been developed axiomatically, including work by Shore and Johnson [113], and Csiszár [28]. However, when those axiomatic assumptions fail to hold, the principle may not be appropriate.

One of Csiszár's axioms [28] is consistency. Csiszár suggests that if a selection rule (such as the principle of minimum relative entropy) chooses a solution s^* from a set of possible solutions S, and then, due to new information, the feasible set shrinks to S', that if the original solution $s^* \in S'$ then it should remain the solution.

Yet there are times when new information may suggest that we change our solution even if it is still feasible.

For example, consider Carnap's problem discussed earlier. Let the original information be that there is a bag with blue balls and some not-blue balls. Based on this information, one might estimate that the probability of drawing a blue ball is 1/2. New information arrives that there are three colors of balls in the bag: blue, yellow, and red. Does Csiszár's axiom of consistency make sense? Even though the original estimate is feasible, it no longer seems like a good bet.

Consider another example. A coin is found. Based on good faith in the government, one estimates the probability of heads to be, *a priori*, 1/2. After 100 flips have yielded 90 heads and 10 tails, what is the new estimate? The original estimate is still feasible, and a policy satisfying Csiszár's aim of consistency would retain the original estimate.

In cases such as the above two examples, a different principle of inference may be in order.

6.3 The principle of minimum expected risk

In this section it is proposed that in some cases a better principle for estimating an unknown pmf is to minimize the expected risk. In particular, minimizing the expected *coding* risk is a relevant goal for some applications.

Estimate an unknown pmf as to be that pmf $\hat{q}$ which minimizes the expected distortion over all the feasible distributions p, where each feasible distribution has probability $f(p)$:

$$\hat{q} = \arg\min_q E_p[D(p,q)] \tag{6.1}$$

If information is known about the relative probability of each feasible distribution p, then that information forms the probability distribution over the feasible distributions, $f(p)$. For example, given observed data x drawn from the pmf to be estimated, one might use the likelihood $f(p) = P(x|p)$. Or if there was a set of priors, the posterior probability could be used, $f(p) = P(p|x)$.

If there is no information to favor one possible pmf over another, we propose invoking the principle of insufficient reason, and supposing that all of the feasible distributions p are equally likely *a priori*. It might be appropriate to minimize any of a variety of distortion measures (such as mean-squared error) for general functions, but for probability distributions minimizing the relative entropy leads to straightforward interpretations:

$$\hat{q} = \arg\min_q E_p[\mathcal{D}(p\|q)] \tag{6.2}$$

Then the estimated pmf $\hat{q}$ minimizes the expected inefficiency of assuming the distribution is $\hat{q}$ when it was really some distribution p. For instance, if the probability distribution is being used to build a code, then note that $D(p||\hat{q})$ is the number of extra bits needed to code a random variable actually drawn from distribution p. Then the principle leads to the distribution $\hat{q}$ which minimizes the expected extra bits needed for coding.

The principle of minimum expected risk, estimates the unknown pmf to be a pmf that is a sort of 'center' of all the possible pmfs, weighted by the probability of each

pmf. In quantization terminology, one can say that the estimated pmf forms a Lloyd centroid for the pmf space.

Consider again the example in which one would like to estimate the probability of flipping heads or tails of a newfound coin. With unshaken faith, one takes the prior p to be $\{1/2, 1/2)\}$. Information becomes available that 'the coin is biased towards heads.' This is a constraint that the probability of heads must be greater than $1/2$. The true pmf must fall in the open set defined by the endpoints: $(\{1/2, 1/2\}, \{1, 0\}]$.

Applying the principle of minimum relative entropy yields $\{1/2+\epsilon, 1/2-\epsilon\}$ where $\epsilon \to 0$. Since the prior is uniform, the principle of maximum entropy yields the same result. Yet given that the information is 'the coin is biased towards heads', the solution proposed by the principle of minimum relative entropy rings false. Instead, one has no information other than to assume that all feasible distributions are equally likely, satisfying the assumptions of the principle of minimal expected risk. The principle of minimum expected risk yields the estimate $\{3/4, 1/4\}$.

In his unfinished work [62], Jaynes notes that it is difficult to communicate mathematically how strongly one believes a prior. With this principle it is possible to express the strength of belief in a prior via the weighting over the priors $f(p)$.

The Bayesian community uses a version of this 'minimum risk principle' for estimating parameters for pmfs [72]. For a pmf $q(x, \theta)$, the Bayesian Mean Square Error Estimator is defined as [72] (pages 310-316, 342-350)

$$\begin{aligned} \hat{\theta}_{BMMSE} &= \int \theta f(\theta|x) d\theta \\ &= \arg\min_{\hat{\theta}} \int (\theta - \hat{\theta})^2 f(\theta|x) d\theta \end{aligned}$$

where a prior distribution over the θ parameter, $f(\theta)$ has yielded a posterior pdf $f(\theta|x)$ based on a new state of knowledge x.

As a general principle, estimating pmfs by minimizing the expected risk may be a useful in a variety of contexts and applications. Analytical solutions are be available for simple pmfs (such as Bernoulli random variables). For more complicated estimations, convex optimization techniques should be usable.

In comparison, the principle of maximum entropy assumes that all training points are *a priori* equally likely to be contributors, and thus should be weighted as equally as possible. The principle of minimum risk instead assumes (given a uniform prior $f(p)$) that all feasible pmfs are equally likely.

For LIME, the intuition was that all training points were equally likely to be contributors, and thus the principle of minimum relative entropy to the uniform prior was used. The principle of minimum expected risk could be used instead. In that case, the linear interpolation equations could be generalized to have as their solution the pmf $\hat{q}$ which solves

$$\hat{q} = \arg\min_{q} D(\sum_{j} q_j x_j, x) + \lambda E_p[\mathcal{D}(p\|q)]. \tag{6.3}$$

6.4 Policy should suit needs

A principle should be applied with full awareness of its assumptions and what the principle aims to do. In the case of the principle of insufficient reason, there must be equally natured events and no further information. In the case of the principle of minimum relative-entropy, the prior should be believed to still be relevant in the light of the new information. In the case of the principle of minimal expected risk, the assumption should be satisfied that all the feasible distributions are equally likely, or that one can specify how they are not equally likely through the distribution over pmfs $f(p)$.

The literature on maximum entropy, and the related notion of minimal relative entropy (also known as relative-entropy) is enormous. In the 80's a conference began that is dedicated to theory and applications of maximum entropy and Bayesian methods, called, *Maximum Entropy and Bayesian Methods in Science and Engineering* [32]. The principle of maximum entropy has been applied to a wide range of problems, from astronomical imaging to the distribution of particles among energy levels [127].

Some applications may be better served by resolving uncertainty with the principle of minimum expected risk. However, no one policy will be appropriate for all

estimations, and one ought to apply a principle of inference with eyes wide open.

Chapter 7

Conclusions

If you persist, you will soon
solve anything at all...

James Thurber

In this work, linear interpolation was extended by using the maximum entropy principle. Relaxing the linear interpolation equations resulted in a flexible algorithm for classification or regression, LIME. In Chapter 1, the issues of bias and high dimensionality in learning problems were explored. In Chapter 2, LIME was shown to address these problems and perform significantly better than other standard non-parametric methods. Theoretical properties of LIME were explored in Chapter 3.

Regression over grids was considered in Chapter 4. The popular interpolation methods, bilinear and trilinear interpolation, were generalized to any dimensional grid. The resulting method, termed PLI, was shown to be related to LIME in that the non-uniqueness of the PLI linear interpolation equations is resolved by the maximum entropy criteria. Experiments and simulations showed that PLI could be a computationally fast and efficient method for multi-dimensional grids.

In Chapter 5, a multi-class pipeline integrity classification application, benchmark data sets, and a rate of convergence simulation provided more data for evaluating the LIME algorithm.

Maximizing entropy is not the only way to resolve uncertainty. In Chapter 6 featured a closer look at principles of inference, and the principle of maximum entropy in specific.

This thesis now ends with some summary of the advantages and disadvantages of LIME and how it may be useful in practice. Lastly, research is by definition never quite finished, and the last section of this thesis ponders extensions and related ideas.

LIME

LIME has been shown to give different and competitive performance on a number of datasets. However, LIME is not computationally simple, and may not result in easily visualized classification boundaries, or easily understood linear surfaces. LIME has been shown to deal well with unsymmetric distributions of training data and high dimensional feature spaces. LIME may yield superior error performance in some applications, but may also be prohibitively difficult, expensive, or ill-suited for real-time applications. However, processing power and speed is ever on the increase, making LIME's complexity a decreasingly important issue.

Hybrid classifiers using a number of different classification algorithms is an active area of research [100], [71] [80] [46] [95]. LIME approaches statistical learning rather differently than the discriminant analysis, neural nets, decision trees, or traditional weighted nearest-neighbor families of methods, and thus may be a valuable algorithm to incorporate into a hybrid system.

PLI

PLI is related to LIME but restricted to 2^d training points at the vertex of a grid and must solve the linear interpolation equations exactly. On the other hand, there is a closed form solution for the PLI weights and no neighborhood search must be done. Also, the surface fit by PLI was shown to have an analytic description. PLI was shown to work well for interpolating functions over grids. It is an useful and efficient method for high-dimensional grid applications.

7.1 Extensions

In this work, a 'vanilla' version of an algorithm was proposed and developed . Many of the advanced techniques of supervised learning could be advantageously applied to the algorithm at hand.

Accuracy would probably be improved by applying state of the art neighborhood selection techniques [115], [52]. Neighborhood selection can make a significant difference for nonparametric methods.

The LIME algorithm maximizes the entropy of the weights given some emphasis on the linear interpolation equations. As discussed in the chapter on principles of inference, maximizing the entropy is equivalent to minimizing the entropy to a uniform distribution. A uniform distribution might not be the best prior, or model for the weights. A kernel that decays with distance may be a better model. Then one could minimize the relative entropy to such a kernel. However, in high dimensions with sparse data, almost all points are roughly equally far away, and a kernel may not result in much better performance.

Adding feature dimensions that are nonlinear transformations of the original feature dimensions has been useful for machine learning, both for discriminant analysis (flexible discriminant analysis [53]) and support vector machines [54]. It would be interesting to explore the impact of such extra feature dimensions for LIME.

In Chapter 3, an analytical exponential form for the LIME weights and consistency results were given for the l_1 distortion. However, the actual implementation used throughout the thesis was an iterative optimization for l_2 distortion. LIME with l_1 distortion was not implemented. Empirical l_2 results as the size of the training set grew agreed with the consistency result obtained with l_1 distortion. It is expected that l_2 and l_1 results would not differ greatly, but this disconnect between consistency theory and practice could be resolved.

For 0-1 cost environments, the estimated observation is the sum of the training observations weighted with the LIME weights. An idea, inspired by the generalized additive model work of Tibshirani and Hastie [51], would be to estimate the observation as the weighted sum of functions of the training data, or as a function of the

weighted sum of functions of the training data.

LIME unravels some local bias, as shown in Chapter 2. Friedman proposed adjusting a density estimation by an additive factor t [35], which could correct for consistent global bias. Friedman's global t-compensation could be added to LIME estimates.

The equations of linear interpolation are useful in resolving bias caused by unsymmetrically distributed training points. As discussed in Section 1.3.2, traditional linear interpolation is not flexible enough for general supervised learning problems. LIME generalizes linear interpolation using the principle of maximum entropy, based on the intuition that all training points are equally likely to be contributors. A different principle of inference might lead to better results in certain cases. For example, generalizing linear interpolation with the principle of minimum expected risk could be explored, as outlined in Section 6.3.

Appendix A

Weierstrass Theorem

A continuous function f defined on a compact set S has a minimum point in S; that is, there is an $x^* \in S$ such that for all $x \in S$, $f(x) \geq f(x^*)$.

Gradient, Hessian

Suppose f is a real-valued function defined over an open set of d-dimensional Euclidean space. Suppose f has continuous first partial derivatives. Then the *gradient* of f is defined to be the vector

$$\nabla f(x) = [\partial f(x)/\partial x_1 \; \partial f(x)/\partial x_2 \; \ldots \; \partial f(x_k)/\partial x_k].$$

Suppose also that f has continuous second partial derivatives. Then the *Hessian* of f is denoted $\nabla^2 f(x)$ and is defined to be the symmetric $d \times d$ matrix

$$\nabla^2 f(x) = \left[\frac{\partial^2 f(x)}{\partial x_i \partial x_j}\right]$$

for $i, j = 1, \ldots, d$.

The Jacobian Matrix

The Jacobian matrix of a vector function $f(x) = [f_1(x); f_2(x); \ldots f_m(x)]^T$, where $x \in \mathcal{R}^n$ is defined as the $m \times n$ matrix whose (i, j)th element is the derivative of f_i with respect to x_j.

Second order necessary minimization conditions

Suppose that x^* is a local minimum of f subject to the set of t constraints $g_1(x^*) = 0, g_2(x^*) = 0, \ldots, g_t(x^*) = 0$, and that the gradient vectors $\nabla g_i(x^*)$, $i = 1, \ldots, t$ are linearly independent. Then there is a $\lambda \in \mathcal{R}^t$ such that

$$\nabla f(x^*) + \lambda^T \nabla g(x^*) = 0$$

Further, it holds that

$$d^T \left(\nabla^2 f(x^*) + \lambda^T \nabla^2 g(x^*) \right) d \geq 0$$

for all d such that $\nabla g(x^*) d = 0$.

Second order sufficiency minimization conditions

Suppose that there is a point x^* satisfying the set of t constraints $g_1(x^*) = 0, g_2(x^*) = 0, \ldots, g_t(x^*) = 0$, and a $\lambda \in \mathcal{R}^t$ such that

$$\nabla f(x^*) + \lambda \nabla g(x^*) = 0.$$

Suppose also that

$$d^T \left(\nabla^2 f(x^*) + \lambda^T \nabla^2 g(x^*) \right) d > 0$$

for all d such that $\nabla g(x^*) d = 0$ and $d \neq 0$. Then x^* is a strict local minimum of f subject to the set of constraints $g(x) = 0$.

Maximum entropy and maximum likelihood

The maximum entropy solution is the maximum likelihood solution under the assumption that no event is known to be more probable than any other event. The basic argument is sketched here as expounded by Wu [127].

Imagine that Nature was forming a distribution over k events by independently distributing B discrete chunks of probability to the events uniformly randomly. Let B_j represent the number of tiny bits of probability that fall on the jth event.

Let $W(\{B_1/B, B_2/B, \ldots B_k/B\})$ represent the number of ways to form the distribution $\{B_1/B, B_2/B, \ldots B_k/B\}$. Then combinatorics dictates,

$$W(\{B_1/B, B_2/B, \ldots B_k/B\}) = \frac{B!}{B_1!B_2!\ldots B_k!}$$

The pmf with maximum likelihood is the pmf that could have occurred in the greatest number of ways. Equivalently, maximize the log of the number of ways:

$$\log W(\{B_1/B, B_2/B, \ldots B_k/B\}) = \log\left(\frac{B!}{\prod_{j=1}^{k} B_j!}\right)$$

Let B become very large (approximating the continuous case) and use Stirling's formula for the logarithm of a factorial:

$$\log W(\{B_1/B, B_2/B, \ldots B_k/B\}) \approx -\sum_{j=1}^{k} B_j \log B_j + B \log B \tag{A.1}$$

Then the distribution that can be formed in the most number of ways is the distribution that maximizes (A.1). Since B is independent of which distribution is chosen, the maximum likelihood distribution is equivalent to the distribution that maximizes entropy.

When a constraint on the set of possible distributions is added, only those distributions that satisfy the constraint are considered. From that restricted set it is asked, which distribution could have occurred in the most ways? Similar logic leads to the distribution with maximum entropy on the restricted set.

Bibliography

[1] www.chromix.com. 2002. ICC profiling service.

[2] www.color.org/profile.html. 2002. International Color Consortium.

[3] www.matlab.com. 2002. Matlab version 6.1 by Mathworks.

[4] www.openmind.org. 2002.

[5] www.stanford.edu/dept/msande/faculty/saunders. 2002.

[6] D. L. Atherton. Magnetic inspection is key to ensuring safe pipelines. *Oil and Gas Journal*, 87(32):52–61, 1989.

[7] S.G. Bakamidis. An exact fast nearest neighbor identification technique. *Proc. of the IEEE ICASSP*, pages 658–661, 1993.

[8] A. R. Barron and T. Cover. Minimum complexity density estimation. *IEEE Trans. on Information Theory*, 37:1034–1054, 1991.

[9] Jean-Francois Bercher, Guy LeBesnerais, and Guy Demoment. The maximum entropy on the mean method, noise, and sensitivity. *Maximum Entropy and Bayesian Methods*, pages 223–232, 1996.

[10] Toby Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Prentice Hall, Englewood Cliffs, New Jersey, 1971.

[11] William Bickford. *A first course in the finite element method*. Irwin, Inc., United States, 1990.

[12] C. L. Blake and C. J. Merz. www.ics.uci.edu/*sim*mlearn/mlrepository.html. 2002. UCI Repository of machine learning databases.

[13] Stephen Boyd and Lieven Vandenberghe. *Introduction to Convex Optimization wth Engineering Applications*. Stanford University, Palo Alto, 1995.

[14] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. *Classification and Regression Trees*. Chapman and Hall, United States of America, 1984.

[15] J. P. Burg. Maximum entropy spectral analysis. *37th Annual International Meeting of the Society of Exploratory Geophysics*, 1967.

[16] J. P. Burg. *Maximum entropy spectral analysis*. Stanford University PhD Dissertation, Stanford, CA, 1975.

[17] A.I. Burshtein. *Introduction to thermodynamics and kinetic theory of matter*. J. Wiley, United States of America, 1995.

[18] L. Lorne Campbell. Minimum cross-entropy estimation with inaccurate side information. *IEEE Trans. Information Theory*, 45:2650–2652, November 1999.

[19] R. Carnap. What is probability? *Scientific American*, 188:128–138, 1953.

[20] James Chang, Jan Allebach, and Charles Bouman. Sequential linear interpolation of multidimensional functions. *IEEE Trans. on Image Processing*, 6(9):1231 – 1245, 1997.

[21] P. A. Chou and R. M. Gray. On decision trees for pattern recognition. *IEEE International Symposiom on Information Theory Summaries*, page 69, 1986.

[22] P. A. Chou, T. Lookabaugh, and R. M. Gray. Entropy-constrained vector quantization. *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, pages 31–42, January 1989.

[23] A. R Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2000.

[24] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, 13:21–27, 1967.

[25] Thomas Cover and Joy Thomas. *Elements of Information Theory*. John Wiley and Sons, United States of America, 1991.

[26] Richard Cox. *Algebra of Probable Inference*. John Hopkins Press, Baltimore, 1961.

[27] I. Csiszár, F. Gamboa, and E. Gassiat. Mem pixel correlated solution for generalized moment and interpolation problems. *IEEE Trans. on Information Theory*, 45(7):2253–2270, November 1999.

[28] Imre Csiszár. Why least squares and maximum entropy? an axiomatic approah to inference for linear inverse problems. *The Annals of Statistics*, 19(4):2032–2066, 1991.

[29] E. F. D'Azevedo. Are bilinear quadrilaterals better than linear triangles. *SIAM Journal of Scientific Computing*, 22(1):198–217, 2000.

[30] Luc Devroye, Laszlo Gyorfi, and Gabor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag Inc., New York, 1996.

[31] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, 2nd Edition*. John Wiley and Sons, New York, 2001.

[32] Gary Erickson and C. Ray Smith. *Maximum Entropy and Bayesian Methds in Science and Engineering*. Kluwer Academic Publishers, U.S.A., 1988.

[33] Mark Fairchild. *Color Appearance Models*. Addison Wesley Inc., Reading, Massachusetts, 1998.

[34] E. Fix and J. L. Hodges. Discriminatory analysis, nonparametric discrimination: Consistency properties. *Technical Report 4*, 1951. USAF School of Aviation Medicine, TX.

[35] Jerome H. Friedman. On bias, variance, 0/1 loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77, 1997.

[36] K. Fukunaga and D. Hummels. Bias of nearest neighbor error estimates. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9:103–112, January 1987.

[37] Robert Gallager. *Information theory and reliable communication*. John Wiley and Sons, New York, 1968.

[38] F. Gamboa and E. Gassiat. Bayesian methods and maximum entropy for ill-posed inverse problems. *The Annals of Statistics*, 25:328–350, 1997.

[39] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[40] James Goin. Classification bias of the k-nearest neighbor algorithm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6:379–381, 1984.

[41] Robert M. Gray. *Entropy and Information Theory*. Springer-Verlag, New York, 1990.

[42] Robert M. Gray and Richard Olshen. Vector quantization and density esimation. *Proc. of the Compression and Complexity of Sequences Conference*, pages 172–193, 1997.

[43] Maya Gupta, Michael Friedlander, and Robert M. Gray. Maximum entropy classification applied to speech. *Proc. of Asilomar Systems and Signals Conference*, 2000.

[44] Maya Gupta and Robert M. Gray. Color conversions using maximum entropy estimation. *Proc. of the IEEE International Conference on Image Processing*, 2001.

[45] H. Gzyl and Y. Velasquez. Maxentropic interpolation by cubic splines with possibly noisy data. *Bayesian Inference and Maximum Entropy Methods in Science and Engineering: 20th International Workshop*, pages 216–228, 2001.

[46] L. Hadjiiski, B. Sahiner, C. Heang-Ping, N. Petrick, and M. Helvie. Classification of malignant and benign masses based on hybrid art2lda approach. *IEEE Trans. on Medical Imaging*, 18(12):1178–1187, 1999.

[47] R. M. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *IEEE Trans. on Systems, Man and Cybernetics*, 3(6):610–621, 1973.

[48] Peter Hart. The condensed nearest-neighbor rule. *IEEE Trans. on Information Theory*, 14:515–516, 1968.

[49] John Hartigan. *Clustering algorithms*. Wiley and Sons, New York, 1975.

[50] T. Hastie and C. Loader. Local regression: automatic kernel carpentry. *Statistical Science*, 8(2):120–143, 1993.

[51] T. Hastie and R. Tibshirani. *Generalized additive models*. St. Edmundsbury Press Limited, Great Britain, 1990.

[52] T. Hastie and R. Tibshirani. Discriminative adaptive nearest neighbour classification. *IEEE Trans. on Pattern Analysis and Machine Learning*, 18(6):607–615, 1996.

[53] T. Hastie, R. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89:1255–1269, 1994.

[54] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001.

[55] Tin Kam Ho and Henry S. Baird. Large-scale simulation studies in pattern recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(10):1067–1079, 1997.

[56] R. C. Holte. Very simple classification rules prform well on most commonly used data sets. *Machine Learning*, 11:63–90, 1993.

[57] David Hume. *An enquiry concerning human understanding*. Oxford University Press, Oxford, 1999.

[58] T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. *Advances in Neural Information Processing Systems 12*, 1999.

[59] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(1):4–37, January 2000.

[60] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.

[61] E. T. Jaynes. On the rationale of maximum entropy methods. *Proc. of the IEEE*, 70(9):939–952, 1982.

[62] E.T. Jaynes. *Probability theory: the logic of science*. Available at ftp://bayes.Wustl.edu/Jaynes.book, 1996. This is Jaynes' unfinished book on probability.

[63] Harold Jeffreys. *Theory of Probability*. Oxford University Press, New York, 1961.

[64] Dhiraj Kacker, Ufuk Agar, Jan Allebach, and Bradley Lucier. Wavelet decomposition based representation of nonlinear color transformations and comparison with sequential linear interpolation. *Proc. of the IEEE International Conference on Image Processing*, 1:186–190, 1998.

[65] Henry Kang. Color scanner calibration. *Journal of Imaging Science and Technology*, 36:162 – 170, 1992.

[66] Henry Kang. *Color Technology for Electronic Imaging Devices*. SPIE Press, United States of America, 1997.

[67] J. N. Kapur. *Measures of Information and Their Application*. Wiley Eastern Limited, New Delhi, 1994.

[68] N. Karayiannis. Meca: Maximum entropy clustering algorithm. *Proc. of the IEEE 3rd International Fuzzy Systems Conference*, pages 630–635, 1994.

[69] J. Kasson, W. Plouffe, and S. Nin. A tetrahedral interpolation technique for color space conversion. *Proc. of the SPIE Conference on Color Imaging*, 1909:127, 1993.

[70] James Kasson, Sigfredo Nin, Wil Plouffe, and James Hafner. Performing color space conversions with three-dimensional linear interpolation. *Journal of Electronic Imaging*, 4(3):226–250, 1995.

[71] Y. Kawata, N. Niki, H. Ohmatsu, M. Kusumoto, R. Kakinuma, K. Mori, H. Nishiyama, K. Eguchi, M. Kaneko, and N. Moriyama. Computer-aided differential diagnosis of pulmonary nodules based on a hybrid classification approach. *Proc. of the SPIE*, 4322:1796–1806, 2001.

[72] Steven M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, New Jersey, 1993.

[73] J. M. Keynes. *A treatise on probability*. Macmillan and Company, London, 1957.

[74] B.S. Kim and S.B. Park. A fast k nearest neighbor finding algorithm based on the ordered partition. *IEEE Trans. on Pattern Analysis and Machine Learning*, 8(6):761–766, 1986.

[75] W. Kneale. *Probability and Induction*. Clarendon Press, Oxford, 1949.

[76] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, United States of America, 1989.

[77] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: benchmarking studies. *IEEE International Conference on Neural Networks*, 1:61–68, 1988.

[78] S.R. Kulkarni, G. Lugosi, and S. S. Venkatesh. Learning pattern classification-a survey. *IEEE Trans. on Information Theory*, 44(6):2178–2206, October 1998.

[79] S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

[80] A. Kumar and I. Olmeda. A study of composite or hybrid classifiers for knowledge discovery. *INFORMS Journal on Computing*, 11(3):267–277, 1999.

[81] Pierre Simon Laplace. *Essai Philosophique sur les Probabilités, fifth edition*. Huzard-Courcier, Paris, 1825.

[82] S. Lubiarz and P. Lockwood. Evaluation of fast algorithms for finding the nearest neighbor. *Proc. of the IEEE ICASSP*, pages 1491–1494, 1997.

[83] J.R. Lucas. *The Concept of Probability*. Clarendon Press, Oxford, England, 1970.

[84] David Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, United States of America, 1984.

[85] G. Lugosi and K. Zeger. Concept learning using complexity regularization. *IEEE Trans. on Information Theory*, 42:48–54, 1996.

[86] Geoffrey McLachlan. *The EM algorithm and extensions*. John Wiley, New York, 1996.

[87] James McNames. A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE Trans. on Pattern Analysis and Machine Learning*, 23(9):964–976, 2001.

[88] John Stuart Mill. *A system of Logic, ratiocinative and inductive; being a connected view of the principles of evidence, and the methods of scientific investigation.* Harper and Brothers, London, 1900.

[89] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, United States of America, 1997.

[90] Halina Mortimer. *The Logic of Induction*. John Wiley and Sons, New York, 1988.

[91] Amir Najmi. *Data compression, model selection and statistical inference*. Stanford University PhD Dissertation, Stanford, CA, 1999.

[92] Stephen Nash and Ariela Sofer. *Linear and Nonlinear Programming*. McGraw-Hill Company, United States of America, 1996.

[93] J. Navaza. The use of non-local constraints in maximum-entropy electron density reconstruction. *Acta Crystallographica*, pages 212–223, 1986.

[94] Deirdre O'Brien, Maya Gupta, Robert M. Gray, and Jon Kristian Hagene. Automatic classification of images from internal optical inspection of gas pipelines. *International Chemical and Petroleum Industry Inspection Technology VIII Conference*, 2003.

[95] I. Olmeda and E. Fernandez. Hybrid classifiers for financial multicriteria decision-making: the case of bankruptcy prediction. *Computational Economics*, 10(4):317–335, 1997.

[96] W. J. Padgett. *Laws of large numbers for normed linear spaces and certain Frechet spaces*. Springer-Verlag, New York, 1973.

[97] E. Parzen. On the estimation of a probability density function and the mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[98] Judea Pearl. An application of rate-distortion theory to pattern recognition and classification. *Pattern Recognition*, 8:11–22, 1976.

[99] C. S. Peirce. *The philosophy of Peirce: selected writings*. Jarrold and Sons Limited, Great Britain, 1956.

[100] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–210, 2001.

[101] W. Pruitt. Summability of independent random variables. *Journal of Math and Mechanics*, 15:769–776, 1966.

[102] V. Rasche, R. Proksa, R. Sinkus, P. Börnert, and H. Eggers. Resampling of data between arbitrary grids using convolution interpolation. *IEEE Trans. on Medical Imaging*, 18(5):385–391, May 1999.

[103] S.J. Raudys and A.K. Jain. Small sample size effects in statistical pattern recognition: recommendations for practitioners. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(3):252–264, March 1991.

[104] S.J. Raudys and V. Pikelis. On dimensionality, sample size, classification error, and complexity of classification algorithm in pattern recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2(3):242–252, 1980.

[105] Nicholas Rescher. *Induction*. University of Pittsburgh Press, Pittsburgh, 1980.

[106] Francesco Ricci and Paolo Avesani. Data compression and local metrics for nearest neighbor classification. *IEEE Trans. on Pattern Analysis and Machine Learning*, 21(4):380–384, 1999.

[107] John Rice. Bandwidth choice for nonparametric regression. *The Annals of Statistics*, 12:1215 – 1230, 1984.

[108] John Rice. Boundary modification for kernel regression. *Communications in Statistics, Theory and Methods*, 13:893–900, 1984.

[109] Kenneth Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. of the IEEE*, 86(11):2210–2239, November 1998.

[110] Riccardo Rovatti, Michele Borgatti, and Roberto Guerrieri. A geometric approach to maximum-speed n-dimensional continuous linear interpolation in rectangular grids. *IEEE Trans. on Computers*, 47(8):894–898, August 1998.

[111] Robert Schalkoff. *Pattern recognition: statistical, structural, and neural approaches*. John Wiley, New York, 1992.

[112] Louis Scharf. *Statistical Signal Processing*. Addison-Wesley, United States of America, 1991.

[113] J.E. Shore and R.W. Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Trans. on Information Theory*, 33:26–37, 1 1980.

[114] J.E. Shore and R.W. Johnson. Properties of cross-entropy minimization. *IEEE Trans. on Information Theory*, 27:472–482, 7 1981.

[115] R. Short and K. Fukunaga. The optimal distance measure for nearest neighbor classification. *IEEE Trans. on Information Theory*, 27(5):622–627, 1981.

[116] D.B. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. *Proc. 11th Intl. Machine Learning Conference*, pages 293–301, 1994.

[117] J. W. Smith, J.E. Everhart, W.C. Dickson, and W.C. Knowler. Using the adap learning algorithm to forecast the onset of diabetes mellitus. *IEEE Proc. of the Symposium on Computer Applications and Medical Care*, pages 261–265, 1988.

[118] Charles Stone. Consistent nonparametric regression. *The Annals of Statistics*, 5(4):595–645, 1977.

[119] Richard Swinburne. *The justification of induction*. Oxford University Press, Oxford, 1974.

[120] Takehisa Tanaka, Katsuji Aoki, Mutsuko Nichogi, and Katsuhiro Kanamori. Color management systems with multilayer perceptrons. *Proc. of the SPIE Conference on Color Imaging*, 3963:110 – 118, 2000.

[121] Robert Taylor. *Stochastic convergence of weighted sums of random elements in linear spaces*, volume 672. Springer-Verlag Lecture Notes in Mathematics, Berlin, 1978.

[122] F.W. Truscott and F. L. Emory. *A Philophical Essay on Probabilities by Pierre Simon, Marquis de Laplace*. John Wiley & Sons, London, 1902. translated from the 6th French edition.

[123] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1991.

[124] G. H. von Wright. *A treatise on induction and probability*. Littlefield, Adams, and Co., New York, 1960.

[125] P.D. Wasserman. *Advanced methods in neural computing*. Von Nostrand Reinhold, New York, 1993.

[126] Sanford Weisberg. *Applied Linar Regression*. John Wiley and Sons, Minnesota, 1985.

[127] Nailong Wu. *The Maximum Entropy Method*. Springer-Verlag, Berlin, 1997.