Weighted Nearest Neighbor Learning and First-order Error

Maya R. Gupta and William H. Mortensen
gupta@ee.washington.edu, william5@u.washington.edu

Dept of EE, University of Washington
Seattle WA, 98195-2500

Abstract

Weighted nearest-neighbor learning is analyzed in terms of squared error, with a focus on classification problems where the squared error of probability estimates is considered. Two classes of algorithms for calculating weights are studied with respect to their ability to minimize the first-order term of the squared error: local linear regression and a new class termed regularized linear interpolation. A number of variants of each class are considered and compared analytically, by simulations, and with experiments on benchmark datasets. The experiments establish that weighting methods which aim to minimize first-order error can perform significantly better than standard k-NN, particularly in high-dimensions. Regularization functions, the fitted surfaces, cross-validated neighborhood size, and the effect of high-dimensionality are also analyzed.

1 Introduction

Nearest-neighbor learning is one of the oldest and most intuitive approaches to classification. Nearest neighbor classifiers can have optimal asymptotic properties [1, 2, 3, 4], and in practice, the simple k-NN classifier often achieves competitive error rates. For example, in a recent paper comparing the standard k-NN to a polynomial-kernel support vector machine (SVM) on 35 benchmark datasets, k-NN tied or beat the SVM on 17 of the 35 datasets [5, Table 10]. Nearest neighbor learning is sometimes called lazy learning because most of the calculations are done at test time, which makes it useful for online applications where the training data changes or grows.

In recent years, researchers have studied how to learn a distance function for nearest-neigbor learning (e.g. [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]), and how to choose the neighbors (e.g. [16, 17, 18]). There have also been significant advances in designing fast algorithms that find approximately-nearest neighbors, allowing a trade-off between speed and accuracy (e.g. [19, 20, 21, 22]).

Recently, using regularized linear interpolation weights was proposed for nearest-neighbor learning [23], and was seen to produce significantly better performance than standard kernel-weighted k-NN for some high-dimensional problems. Similarly state-of-the-art classification results have been shown with kernelized (similarity-based) [24]. These results motivated this present analysis of how to optimally weight nearest-neighbors to minimize zero-order error, first-order error and estimation variance. Our focus is on classification, which we analyze as a regression problem where the class posterior probabilities must be estimated.

After establishing the nearest neighbor problem and notation, we analyze the first-order error for weighted nearest neighbors in Section 2. We show that there are two approaches that can be used to minimize the first-order error: local linear regression and regularized linear interpolation. First we consider local linear regression in Section 3. We review how local linear regression can be formulated as a weighted nearest-neighbor classifier, and then we analyze the zero-order error and first-order error for different variants, some new. Then we consider the new class of regularized linear interpolation classifiers in Section 4. Some analytic comparisons are made between these two classes of nearest-neighbor methods in Section 5. Some simulations and benchmark datasets are presented in Section 6, and are used to further analyze the different methods in Section 7. The paper closes with conclusions and open questions. For the reader's reference, Table 1 summarizes the frequently-used notation.

This paper builds on previous analysis of local learning, predominantly [25] and [3]. The main contributions of this paper are: the multi-dimensional first-order analysis which shows how to optimally weight nearest-neighbors, the analysis of local linear regression with respect to zero-order and first-order error for different neighborhood sizes k

Table 1: Key Notation

I	identity matrix	$I_{(\cdot)}$	indicator function
$X_j \in \mathbb{R}^d$	j^{th} random training sample ($d \times 1$)	n	number of training samples
$Y_j \in \{1, 2, \ldots, G\}$	random class label for X_j	$\mathcal{J}$	indices for k neighbors of X
G	number of class labels	k	number of neighbors $\lvert\mathcal{J}\rvert$
$X \in \mathbb{R}^d$	random test sample ($d \times 1$)	β	regression model coefficients
$X[i]$	ith component of vector X	β_g	gth component of β
$\hat{Y} \in \{1, 2, \ldots, G\}$	predicted class label for X	w, v	$k \times 1$ weight vectors
$\mathbb{X} \in \mathbb{R}^{d+1 \times k}$	matrix with columns $[X_j\ 1]^T$ for $j \in \mathcal{J}$	w_j	weight on (X_j, Y_j) for $j \in \mathcal{J}$
$\mathbb{X}_0 \in \mathbb{R}^{d \times k}$	matrix with columns X_j^T for $j \in \mathcal{J}$	K	pre-weight kernel
$f(X) \in [0, 1]^G$	class probability vector given X	A	$A[j, j] = K(X - X_k)$, else 0
$f_g(X) \in [0, 1]$	gth element of $f(X)$; $\sum_{g=1}^G f_g(X) = 1$	R	regularization function
$\hat{f}(X) \in \mathbb{R}^G$	estimate of $f(X)$	D	continuous convex function

and number of features d, and the analytic and experimental comparisons of standard and new variants of local linear regression and regularized linear interpolation classifiers. A key conclusion is that while standard k-NN is a popular baseline and can perform well in practice, weighted k-NN with weights formed by regularized linear regression or regularized linear interpolation can significantly improve performance, particularly in very high-dimensional cases where standard k-NN can simply fail.

1.1 The Nearest Neighbor Problem and Notation

Denote a random test feature vector $X \in \mathbb{R}^d$ and random training samples pairs $\{(X_j, Y_j)\}$ for $j = 1 \ldots, n$, where $X_j \in \mathbb{R}^d$, and the associated class label $Y_j \in \{1, 2, \ldots, G\}$. For our analysis, we assume that all feature vectors are in general position. For some results, we will further assume that X and each X_j are drawn independently and identically from some distribution P_X. Let a set of neighbors for X be defined by a set of k indices $\mathcal{J}$, such that training sample X_j is in the neighborhood of test point X if $j \in \mathcal{J}$. Weights are calculated for each of the neighbors, and we use w_j to denote the weight on the neighbor X_j, where w_j may be a function of X and $\{X_j\}$ for $j \in \mathcal{J}$. The weights may or may not be positive, and may or may not sum to one.

For the multi-class classifier with G classes, the class estimate $\hat{Y}$ corresponding to X is

$$\hat{Y} = \underset{g \in \{1,2,\ldots,G\}}{\operatorname{argmin}} \sum_{h=1}^{G} C(g, h) \left(\sum_{j \in \mathcal{J}} w_j \hat{f}_h(X_j) \right), \tag{1}$$

where $C(g, h)$ is the cost of classifying X as class g if its true class is class h, and $\hat{f}_h(X_j)$ is the estimated probability that X_j is of class h, usually taken to be $\hat{f}_h(X_j) = 1$ if $Y_j = h$ and $\hat{f}_h(X_j) = 0$ otherwise. The term $\sum_j w_j \hat{f}_h(X_j)$ can be interpreted as a maximum likelihood estimate of the probability that X is from class h given constant class probabilities over the neighborhood [26].

Many weighting functions have been proposed that are based on the choice of a fixed decreasing function K such that $w_j = \gamma K(X - X_j)$, where γ is a normalization constant so that $\sum_j w_j = 1$ [27, 25]. The rationale for such distance-decay of the weights is that training samples further from the test point may be less relevant. However, unless the training samples are in symmetric locations about the test sample, such Nadaraya-Watson estimates will have a first-order bias [25].

2 First-order Error

In this section we present an analysis of the squared error for weighted nearest neighbor class probability estimates, and establish how to minimize first-order error.

Let $f(X)$ be a $G \times 1$ vector of class probabilities for X, with $f_g(X)$ the probability that X is from the gth class. For this theoretical analysis, we assume the true class probabilities $\{f(X_j)\}$ are known. In practice what is given is

Y_j, which is only one sample drawn from $f(X_j)$, and can be used to form an estimate of $f(X_j)$. We discuss the impact of knowing Y_j rather than $f(X_j)$ at the end of the section.

For weighted-nearest neighbor classification, the estimate $\hat{f}(X)$ of $f(X)$ is restricted as per (1) to be a weighted combination of the neighbor class probability vectors $\{f(X_j)\}$ such that $\hat{f}(X) = \sum_j w_j f(X_j)$.

The weights that produce the optimal nearest neighbor probability estimate with respect to squared error would solve

$$w^* = \underset{w \in \mathbb{R}^k}{\operatorname{argmin}} \sum_{g=1}^{G} \left(f_g(X) - \sum_{j \in \mathcal{J}} w_j f_g(X_j) \right)^2 \tag{2}$$

Squared error is a standard error function for regression, and an appropriate error for classification because the posterior probability estimates that minimize the expected mean-squared error also minimize the expected misclassification cost [28].

Let $f : \mathbb{R}^d \rightarrow \mathbb{R}^G$ have gth component f_g, and suppose f_g has continuous derivatives. (Note that by the Stone-Weierstrass theorem, if f_g is only continuous, it can be uniformly approximated by a polynomial for any closed interval and this analysis can be done for the approximating polynomial). Form a Taylor series expansion of each $f_g(X_j)$,

$$f_g(X_j) = f_g(X) + \sum_{r=1}^{\infty} \frac{1}{r!} \left((X_j - X) \cdot \nabla \right)^r f_g(X). \tag{3}$$

Substitute (3) into (2), and define the zero-order error $e_g = f_g(X) - f_g(X) \sum_{j \in \mathcal{J}} w_j$. Note that $e_g = 0$ if and only if the weights are normalized such that $\sum_{j \in \mathcal{J}} w_j = 1$. The optimization problem (2) can be expressed:

$$w^* = \underset{w \in \mathbb{R}^k}{\operatorname{argmin}} \sum_{g=1}^{G} \left(e_0 - \sum_{j \in \mathcal{J}} \left(w_j (X_j - X)^T \nabla f_g(X) + w_j \sum_{r=2}^{\infty} \frac{1}{r!} \left((X_j - X) \cdot \nabla \right)^r f_g(X) \right) \right)^2 . \tag{4}$$

Assume $e_g = 0$, expand the square in (4), and consider only the first term, which depends only on the first-order gradients of the true class probabilities. For classification problems, this first-order term may constitute a significant portion of the total error. Then, the first-order error of (4) is minimized by the weights

$$\begin{aligned} w^* &= \underset{w \in \mathbb{R}^k}{\operatorname{argmin}} \sum_{g=1}^{G} \left(\left(\sum_{j \in \mathcal{J}} w_j X_j - X \right)^T \nabla f_g(X) \right)^2 \\ &\equiv \underset{w \in \mathbb{R}^k}{\operatorname{argmin}} \left(\sum_{j \in \mathcal{J}} w_j X_j - X \right)^T F \left(\sum_{j \in \mathcal{J}} w_j X_j - X \right), \end{aligned} \tag{5}$$

where the $d \times d$ matrix $F = \sum_g \nabla f_g(X) \left(\nabla f_g(X) \right)^T$. Thus for multi-dimensional problems, if the weights solve $\sum_j w_j X_j = X$ they yield zero first-order error.

If no weights solve $\sum_j w_j X_j = X$, then the best weights (with regard to first-order error) depend on F, unless the dimension $d = 1$ such that F is a scalar and can be ignored when minimizing (5). For multi-dimensional problems where $d > 1$, F can be interpreted as changing the distance measure based on the test sample. This analysis provides theoretical weight to learning a local distance measure for each test point, as already attempted for standard k-NN (see for example, [11, 12, 13, 14]). In general, the matrix F will not be diagonal, since it is the sum of outer products of gradient vectors, and thus, (5) is generally not equivalent to first scaling each feature, and then ignoring F.

In practice the known quantities are $\{Y_j\}$, not $\{f(X_j)\}$, which changes the weighted nearest neighbor estimate. However, if the Bayes' error is zero conditioned on each $\{X_j\}$, then the above analysis is exact, with $f(X_j)$ equal to the $G \times 1$ vector with $f_g(X_j) = 1$ if $Y_j = g$ and zero otherwise. For the case that the Bayes' error is not zero, let $\hat{f}_g(X_j) = 1$ if $Y_j = g$, and zero otherwise, and let $\hat{f}(X_j)$ be the $G \times 1$ vector with gth component $\hat{f}_g(X_j)$. Then $\hat{f}(X_j) = f(X_j) + N_j$, where N_j is a random $G \times 1$ vector with $E[N_j] = \vec{0}$. Because the N_j are independent for $j \in \mathcal{J}$, the N_j add randomness that will predominantly manifest as high-order error terms especially for large k. Thus, using the nearest neighbor classifier with the $\{Y_j\}$ and not the true $\{f(X_j)\}$ may have only a negligible impact on the above zero-order and first-order error analysis.

In some cases, first-order optimal weights such that they solve $\sum_j w_j X_j = X$ can be computed in closed-form using local linear regression. We review local linear regression in the next section, and discuss how to modify it to be a more effective classifier.

3 Local Linear Regression Weights for Classification

Local linear regression is a standard regression technique in which a hyperplane is fit to training samples that are local to a test sample [29, 25]. First, we review how local linear regression forms a closed-form weighted nearest-neighbor classifier [3]. Then we analyze the zero-order and first-order error, note that an easily-fixed zero-order error problem arises when $k < d$, and consider a standard and new approach to regularizing the classifier.

To use local linear regression for the G-class classification problem, fit G hyperplanes to the neighbors of the test point X, where the gth hyperplane is fit to $\{(X_j, \hat{f}_g(X_j))\}$. Calculate the G discriminants $\{\hat{f}_g(X)\}$, such that $\hat{f}_g(X) = \hat{\beta}_g^T X + \hat{\beta}_{g,0}$ where $\hat{\beta}_g, \hat{\beta}_{g,0}$ are the estimated $d \times 1$ gradient vector and scalar offset for the gth class in the local neighborhood of the test point X. Then the estimated class label is calculated:

$$\hat{Y} = \underset{g \in \{1,2,\ldots,G\}}{\operatorname{argmin}} \sum_{h=1}^{G} C(g,h) \left(X^T \hat{\beta}_g + \hat{\beta}_{g,0} \right). \tag{6}$$

To compute $\hat{\beta}_g$ and $\hat{\beta}_{g,0}$ let $\hat{f}_g$ be a $k \times 1$ vector with ordered elements $\hat{f}_g(X_j)$ for $j \in \mathcal{J}$, and let $\mathbb{X}$ be a $(d+1) \times k$ matrix with ordered columns $\begin{bmatrix} X_j \\ 1 \end{bmatrix}$ for $j \in \mathcal{J}$. Ideally, the model coefficients $\hat{\beta}_g, \hat{\beta}_{g,0}$ would solve

$$\mathbb{X}^T \begin{bmatrix} \hat{\beta}_g \\ \hat{\beta}_{g,0} \end{bmatrix} = \hat{f}_g. \tag{7}$$

The system of equations (7) has k constraints and $d+1$ unknowns, and when $k > d+1$ there is no solution (because the points were assumed to be in general position). A common fix is to only require that the squared error between the terms in (7) be minimized:

$$(\hat{\beta}_g, \hat{\beta}_{g,0}) = \underset{(\beta \in \mathbb{R}^d, \beta_0 \in \mathbb{R})}{\operatorname{argmin}} \sum_{j \in \mathcal{J}} \left(\hat{f}_g(X_j) - \beta^T X_j - \beta_0 \right)^2. \tag{8}$$

Problem (8) is solved by the Moore-Penrose pseudoinverse *(pinv)* [30]:

$$\begin{bmatrix} \hat{\beta}_g \\ \hat{\beta}_{g,0} \end{bmatrix} = \begin{cases} \mathbb{X} \left(\mathbb{X}^T \mathbb{X} \right)^{-1} \hat{f}_g & \text{if rank}(\mathbb{X}) = k \\ \left(\mathbb{X} \mathbb{X}^T \right)^{-1} \mathbb{X} \hat{f}_g & \text{if rank}(\mathbb{X}) = d+1 \end{cases} \tag{9}$$

where (7) is also solved if $\text{rank}(\mathbb{X}) = k$.

Given the model coefficients (9), the classification discriminant can be expressed

$$\hat{f}_g(X) = \begin{bmatrix} X \\ 1 \end{bmatrix}^T \begin{bmatrix} \hat{\beta}_g \\ \hat{\beta}_{g,0} \end{bmatrix} = w^T \hat{f}_g = \sum_{j \in \mathcal{J}} w_j \hat{f}_g(X_j),$$

where

$$w = \begin{cases} \left(\mathbb{X}^T \mathbb{X} \right)^{-1} \mathbb{X}^T \begin{bmatrix} X \\ 1 \end{bmatrix} & \text{if rank}(\mathbb{X}) = k \\ \\ \mathbb{X}^T \left(\mathbb{X} \mathbb{X}^T \right)^{-1} \begin{bmatrix} X \\ 1 \end{bmatrix} & \text{if rank}(\mathbb{X}) = d+1. \end{cases} \tag{10}$$

Thus (6) is equivalent to the standard weighted nearest-neighbor classifier (1) if the model coefficients β solve (8). This re-interpretation as a nearest-neighbor classifier is helpful in practice because one need only calculate the weights w given in (10), which conveniently do not depend on the class labels or the number of classes G. We refer to this classifier as the *pinv classifier*.

An interesting symmetry emerges: the model coefficients given in (9) are the pinv solution to the model-fitting problem (8) which does not include the test point, whereas the weight vector given in (10) is the pinv solution that minimizes the squared error in expressing the test point as a linear sum of the training samples:

$$w = \underset{v}{\operatorname{argmin}} \left\| \mathbb{X}v - \begin{bmatrix} X \\ 1 \end{bmatrix} \right\|_2^2, \tag{11}$$

which does not depend on the class probabilities $\{f(X_j)\}$. Minimizing the squared norm of the weights corresponds to minimizing estimation variance with respect to noise in this context (see Proposition 1 in Section 4.2.2).

If $\text{rank}(\mathbb{X}) = d+1$, then $k \geq d+1$, and the hyperplane fit in (7) may not hold, but (11) is solved exactly such that the $d+1$ system of equations holds:

$$\mathbb{X}w = \begin{bmatrix} X \\ 1 \end{bmatrix}. \tag{12}$$

On the other hand, if $\text{rank}(\mathbb{X}) = k$, then $k \leq d+1$, and (9) exactly solves the linear regression problem (7), but (12) may not hold. Thus, if $k < d+1$, the pinv weights are not guaranteed to yield zero first-order error, and generally will not minimize the first-order error (5) due to the term F in (5).

In addition, if $k < d+1$, the pinv weights are not guaranteed to sum to one, so there may also be zero-order error. To fix this, we form a slightly different weighted nearest-neighbor classifier by first subtracting out the mean class posterior as follows. Assume $\text{rank}(\mathbb{X}) = k$, and define the weights $v^* = \left(\mathbb{X}_0^T \mathbb{X}_0\right)^{-1} \mathbb{X}_0^T X$ where $\mathbb{X}_0$ has ordered columns X_j for $j \in \mathcal{J}$. The weights v^* are the pinv weights for fitting hyperplanes to the pairs $\left\{(X_j, f_g(X_j) - \bar{f}_g)\right\}$ for $g \in \{1, 2, \ldots, G\}$, where $\bar{f}_g = \frac{1}{k}\sum_{i \in \mathcal{J}} f_g(X_i)$. Then

$$\hat{f}_g(X) = (\hat{f}_g - \bar{f}_g)^T v^* + \bar{f}_g = w^{*T} \hat{f}_g,$$

for $w^* = v^* - \frac{1}{k}\sum_{i \in \mathcal{J}} v_i^* + \frac{1}{k}$. The weights w^* sum to one, and we term them the *pinv weights norm one*.

3.1 Pre-weighting the Model Errors

A common variant of local linear regression is to pre-weight the error term corresponding to each training sample in (8), where the pre-weight is a decreasing function of distance to the test sample. The pre-weight matrix A is a diagonal, $k \times k$ matrix where $A[j,j] = K(X - X_j)$ and K is a decreasing function of its argument. Pre-weighted linear regression with a choice of K that down-weights outliers is termed lowess (also known as LOWESS, LOESS, or loess). A popular choice of K is the tricube function [27],

$$K(X - X_j) = \begin{cases} \left(1 - \left(\frac{\|X - X_j\|_2}{\|X - X_{far}\|_2}\right)^3\right)^3, & j \in \mathcal{J} \\ 0 & j \notin \mathcal{J} \end{cases} \tag{13}$$

where X_{far} is X's furthest neighbor in $\mathcal{J}$.

Given a pre-weight matrix A, the model coefficients for the case that $\mathbb{X}$ has rank k are $\mathbb{X}A\left(\mathbb{X}^T\mathbb{X}A\right)^{-1}\hat{f}_g$, and the lowess weights are $w = \left(A\mathbb{X}^T\mathbb{X}\right)^{-1} A\mathbb{X}^T \begin{bmatrix} X \\ 1 \end{bmatrix}$.

3.2 Regularized Local Linear Regression

Another common variant of linear regression is regularized linear regression. In this paper, we consider the popular example of ridge regression [31, 27], which uses a squared ℓ_2 penalty on the model coefficients and results in a closed-form solution. Since the estimated coefficients will change if the input values are scaled, the inputs are normalized first. Take the k training samples and shift them so they have mean zero, and scale them so they have identity covariance, and perform the same shifting and scaling on the test point X. Then let $\mathbb{X}_r$ be the $d \times k$ matrix whose columns are the shifted, scaled versions $\widetilde{X}_j$ of the feature vectors X_j, let $\widetilde{X}$ be the shifted, scaled version of the test point X, and let $\bar{f}_g \in \mathbb{R}$ be the mean of the $\hat{f}_g(X_j)$'s. Then, the model coefficients in the shifted, scaled space solve

$$\hat{\beta}_g = \underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{j \in \mathcal{J}} \left(\hat{f}_g(X_j) - \bar{f}_g - \beta^T \widetilde{X}_j\right)^2 + \kappa \beta^T \beta, \tag{14}$$

where the scalar κ dictates the trade-off between the least-squares fit and the regularization of the regression coefficients. The ridge regression problem (14) is solved by the closed-form solution

$$\hat{\beta}_g = \left(\mathbb{X}_r \mathbb{X}_r^T + \kappa I\right)^{-1} \mathbb{X}_r (\hat{f}_g - \bar{f}_g),$$

where I is the $d \times d$ identity matrix. Then the discriminant is given by

$$\hat{f}_g(X) = \hat{\beta}_g^T \widetilde{X} + \bar{f}_g = v^T (\hat{f}_g - \bar{f}_g) + \bar{f}_g = w_R^T \hat{f}_g, \tag{15}$$

where $v = \mathbb{X}_r^T \left(\mathbb{X}_r \mathbb{X}_r^T + \kappa I\right)^{-1} \widetilde{X}$ and the ridge weights are $w_R = v - \frac{1}{k} \sum_{j \in \mathcal{J}} v_j + \frac{1}{k}$. Because the ridge weights w_R sum to one, there will be no zero-order error. Unlike the pinv weights (10), w_R will not in general yield zero first-order error as per (5) due to the regularization term.

[32] showed that a variant of the above local ridge regression achieved lower error on the benchmark handwritten character classification task than k-NN, a Parzen window classifier, and a neural network. In their implementation of ridge regression, they shift the X_j by X, and do not subtract the local mean $\bar{f}_g$, which will result in zero-order error. Also, they do not standard-normalize the X_j, which changes the effect of the regularization.

We propose regularizing the weights directly, that is, solve

$$\operatorname*{argmin}_v \|\mathbb{X}_0 v - X\|_2^2 + \kappa v^T v,$$

which has solution

$$v^* = \left(\mathbb{X}_0^T \mathbb{X}_0 + \kappa I\right)^{-1} \mathbb{X}_0^T X.$$

We term the normalized weights $w^* = v^* - \frac{1}{k} \sum_{j \in \mathcal{J}} v_j^* + \frac{1}{k}$ the *regularized pinv weights*, and compare the different regularizations in the experiments.

We expect that other robust regression methods will perform similarly to the squared ℓ_2 penalty of ridge and regularized pinv for classification problems [27]. At least one comparison has shown ridge regression to be more effective than variable subset selection, principal components regression, or partial least squares regression [33]. Although ℓ_p regularization penalties can provide sparser solutions, in this context sparser solutions offer little advantage compared to the squared ℓ_2 penalty which produces both closed-form solutions and noise averaging because of the non-sparse model coefficients and weights.

4 Linear Interpolation for Classification

In this section, we consider a second weighted nearest-neighbor approach to minimizing the first-order error that we term *regularized linear interpolation classifiers*. These classifiers aim to solve $\mathbb{X}_0 w = X$ with the constraints that the weights $w \in [0,1]^k$ and $\sum_{j \in \mathcal{J}} w_j = 1$. The difference with respect to using local linear regression is the constraint $w \in [0,1]^k$, which by restricting the solution, should produce lower estimation variance. Additionally, this constraint is key if the desired output is an estimate of the class posterior for the gth class, such as $\sum_{j \in \mathcal{J}} w_j I_{Y_j = g}$. Because the pinv weights are unrestricted to $[0,1]$, the class posterior estimates are unrestricted to $[0,1]$, and clipping the class posterior estimates to this interval may result in an estimated class posterior that does not sum to one for all the classes.

Linear interpolation is a standard approach to interpolation [34, 18, 35], and it was first proposed for classification by [36]. Their algorithm later was formalized as the *linear interpolation with maximum entropy* (LIME) classifier [23]. Here, we generalize the LIME weights to *regularized linear interpolation weights*, which solve

$$\operatorname*{argmin}_{w \in [0,1]^k} \left(D\left(\mathbb{X}_0 w - X\right) + \lambda R(w)\right) \tag{16}$$

such $\sum_j w_j = 1$, and where D is a continuous strictly-convex function, λ is a trade-off parameter, and R is some regularizing function used to reduce estimation variance and define a unique solution.

We also define *constrained regularized linear interpolation weights*, which do not require a regularization parameter λ:

$$\operatorname*{argmin}_{w \in [0,1]^k} R(w) \quad \text{such that} \quad D\left(\mathbb{X}_0 - X\right) = \min_{z \in [0,1]^k} D\left(\mathbb{X}_0 z - X\right) \tag{17}$$

such that $\sum_{j \in \mathcal{J}} w_j = 1$ and $\sum_{j \in \mathcal{J}} z_j = 1$.

The regularized linear interpolation objective (16) is similar to other regularization problems in statistical learning, such as the minimization problems for ridge regression given in (14), or the weight decay regularization used with neural networks [27]. However, (16) differs in that it does not trade-off the fit of the regression function to the data with a penalty on the smoothness of the regression function. In contrast, the regularized linear interpolation objective (16) trades-off the error between the test point X and a convex combination of the neighboring training samples $\{X_j\}$ with a penalty on the weights, without regard for the values $\{f(X_j)\}$. Of course, if $R(w)$ rewards more uniform weights, then the resulting function will be more locally-constant, and in this sense, locally smoother.

Next, we review the LIME classifier, then propose and analyze two other regularization criteria. These classifiers do not directly minimize the first-order error given in (5). To that end, we propose in Section 4.3 a new regularized linear interpolation classifier that aims to explicitly minimize (5). All of the described regularized linear interpolation classifiers are convex optimization problems and can be implemented with standard convex programming methods.

4.1 LIME weights

LIME solves (16) where the regularization function is the negative entropy of the weights $R(w) = \sum_{j \in \mathcal{J}} w_j \ln w_j$ [23]. Maximizing entropy keeps each of the weights away from zero, so that all neighbors are included in the estimation, and pushes the solution towards the uniform weight vector. The LIME weights exist and are unique because the objective function is a continuous strictly convex function of w.

Simulation results showed LIME could achieve significantly better results than using uniform or distance-decaying weights, and better results than local linear regression weights [23]. In applications, LIME classification has performed better than other classifiers including the boosted decision tree MART, regularized QDA, and Gaussian mixture model classification [23, 37, 38, 39].

Applying Corollary 4.5 of [40], it follows that if the distortion function D is taken to be some ℓ_p norm, and if $\lambda \leq \lambda_0$ where λ_0 depends on the training and test data, then solving for the LIME weights is equivalent to solving the constrained problem given in (17) with $R(w) = \sum_{j \in \mathcal{J}} w_j \ln w_j$. [23] showed that interpolation with these constrained LIME (cLIME) weights over a regular grid is equivalent to the standard linear interpolation technique of interpolating a regular grid dimension-by-dimension, such as the bilinear interpolation commonly used in image processing, and the trilinear interpolation used to interpolate look-up tables for color management [34].

4.2 Different regularization strategies

We propose two different regularization functions $R(w)$ for (16): relative entropy and variance.

4.2.1 Linear interpolation with minimum relative entropy (LIMRE)

One expects that neighbors further away from the test point may be less relevant to classifying the test point, and thus we propose regularizing the weights towards a weight vector that decays with distance from the test point. To this end, given a normalized, strictly positive weight function v let the linear interpolation with minimum relative entropy (LIMRE) weights solve (16) with regularization $R(w) = \sum_{j \in \mathcal{J}} w_j \ln(w_j / v_j)$. The LIMRE weights exist and are unique because the objective function is a continuous strictly convex function of w.

4.2.2 Linear interpolation with minimum variance (LIMV)

Let the linear interpolation with minimum variance (LIMV) weights solve (16) with $R(w) = w^T w$. Note that minimizing $w^T w$ is equivalent to minimizing the variance of the weights in this case, because the sum of the w_j is constrained. The LIMV weights exist and are unique because the objective function is a continuous strictly convex function of v. This is a quadratic optimization problem which can be solved with standard quadratic programming methods. However, the LIMV weights are the optimal regularized linear interpolation classifier in terms of minimizing variance given noisy measurements:

Proposition 1 [LIMV Minimizes Estimation Variance]: Consider the case that in the neighborhood $\mathcal{J}$ of the test point x, $f(x_j) = g(x_j) + N_j$, where $g(x_j)$ is a deterministic function of x_j, and each N_j is independently and identically drawn from a distribution with zero mean and finite variance σ^2. Consider also the second case that $f(x_j) = a^T(x_j + \vec{N}_j)$, where $a \in \mathbb{R}^d$ is some constant such that $a^T a$ is bounded and $\vec{N}_j$ is a d-dimensional vector where each

component for each j is drawn iid from a distribution with zero mean and finite variance σ^2. Express the weighted nearest-neighbor estimate as a functional of the regularization function R such that $\hat{f}(x, R) = \sum_{j \in \mathcal{J}} w_j^R f(x_j)$, where the w^R satisfy (16) with regularization function $R(w)$. Then for both cases, of all the weight vectors w^R that achieve the same distortion D, the estimation variance $\mathrm{var}_N(\hat{f}(x, R))$ is minimized if $R(w) = w^T w$.

The proofs of this and all the paper's propositions are in the appendix.

4.3 Gradient LIME

We propose a variant of LIME that we term *gradient LIME* which attempts to explicitly minimize (16). Let gradient LIME weights solve (16) with distortion function $D(\tilde{x}) = \left(\|\tilde{x}\|_{\hat{F}}\right)^2 = \tilde{x}^T \hat{F} \tilde{x}$, where $\hat{F}$ is an estimate of F, and the regularization is maximum entropy, $R(w) = \sum_j w_j \ln w_j$.

For the experiments presented in Section 6 we estimate $\hat{F} = \sum_g \left(\nabla \hat{f}_g\right) \left(\nabla \hat{f}_g\right)^T$, where each $\nabla \hat{f}_g$ is estimated individually as the d regression coefficients of the ridge-regression hyperplane fitted to the $\{\hat{f}_g(X_j)\}$ for the neighborhood samples $j \in \mathcal{J}$. It is common wisdom that a little regularization can have a large stabilizing effect, and to avoid more free parameters and the associated risk of overfitting, we use a default $\kappa = 10^{-9}$ for the ridge regression regularization parameter.

4.4 Exponential Weights for LIME, LIMRE, and Gradient LIME

As a consequence of maximizing the entropy subject to the soft moment constraint $\sum_j w_j X_j = X$, the LIME weights are an exponential function of the training samples such that [40]

$$w_j^{LIME} = \frac{e^{-\phi^T X_j}}{\sum_{i \in \mathcal{J}} e^{-\phi^T X_i}}, \tag{18}$$

where $\phi \in \mathbb{R}^d$ is a function of the test and training samples. The exponential form of the weights shows clearly that the LIME weights are asymmetric about the test point.

By direct application of Lemma 3.1 from [40], the LIMRE weights also have an asymmetric exponential form such that

$$w_j^{LIMRE} = \frac{v_j e^{-\phi^T X_j}}{\sum_{i \in \mathcal{J}} v_i e^{-\phi^T X_i}}.$$

Further, by direct application of Corollary 4.4 of [40], if the distortion function D in (16) is an exact penalty (such as an ℓ_p norm), then the exponential decay ϕ is bounded.

Similarly, Lemma 3.1 and Corollary 4.4 of [40] can be applied to the gradient LIME weights (because $\|\tilde{X}\|_{\hat{F}}$ is a convex function) to show that the gradient LIME weights have the same exponential form as the LIME weights (18).

4.5 First-order Error for Regularized Linear Interpolation Classifiers

In this section, we establish that the regularized linear interpolation classifiers have optimal first-order error under some conditions.

Proposition 2: [First Order Optimal for $d = 1$] For $d = 1$, solutions of (17) have equal or smaller first-order error than any other positive and normalized weights over the same neighborhood.

Proposition 3: [Zero First-Order Error for Enclosing Neighborhood] Consider weighted nearest-neighbor classification over an *enclosing neighborhood* [41, 16, 42] which is defined to be any $\mathcal{J}$ such that there exists some normalized positive weight vector a such that $\sum_{j \in \mathcal{J}} a_j X_j = X$. Then solutions of (17) have zero first-order error as per (5).

Proposition 4: [Smaller First-Order Error Than Uniform] If the matrix F in (5) is known such that the distortion function in (16) is $D = \|\cdot\|_F^2$, then the LIME weights and LIMV weights yield equal or smaller first-order error as per

(5) than uniform weights, and the LIMRE weights yield equal or smaller first-order error given in (5) than using the weight vector v used in the LIMRE regularization function. If $D = \|\cdot\|^2$, then the above statements are guaranteed to hold only for $d = 1$.

5 Further Analysis

We present some further analysis comparing the two classes of weighted nearest-neighbor classifiers, the pinv classifiers and the regularized linear interpolation classifiers.

5.1 Regression Surfaces

Consider the estimated class probability surface formed by $\hat{f}_g(x, w) = \sum_j w_j(x) f_g(x_j)$ for the different weighting methods over a fixed set of neighbors $\{(x_j, f(x_j))\}$, for $j \in \mathcal{J}$. For uniform weights the fitted surface is a constant. For all of the discussed local linear regression variants, the fitted surface is always a hyperplane, as the fitted surface can be expressed as $x^T\beta$ for some β that does not depend on x.

For the linear interpolation weights, we are not able to describe the fitted surface in general. For test points that lie outside the convex hull of the training samples, the fitted surface will be constant along hyper-rays extending from the convex hull of the training samples because of the constraint $w \in [0, 1]^k$: suppose x lies outside the convex hull, and let w^* be the linear interpolation weights that solve (16), or (17), for some x that lies outside the convex hull of the neighbors. Define $\hat{x} \doteq \sum_j w_j^* x_j$, and note that $\hat{x}$ lies in the closure of the convex hull by the constraints on w^*. Then w^* is also the minimizing solution for $\hat{x}$, and thus $f(x, w^*) = f(\hat{x}, w^*) = \sum_j w_j^* f(x_j)$.

Some insight into the regularized linear interpolation classifiers can be gained by considering the regression surface when the neighbors lie on the vertices of a unit hypercube. For this case, we can show that the functional form is linear in a product basis expansion of the original features of x.

Proposition 5: [cLIME Fitted Surface] Let Υ be the set $\{x[1], x[2], \ldots, x[d]\}$. Let $P(\Upsilon)$ be the power set of Υ, and assume a lexicographical ordering of the 2^d subsets comprising $P(\Upsilon)$. Define $z_i(x)$ to be the arithmetic product of the elements in the ith subset of $P(\Upsilon)$, where the product of the empty set is defined to be 1. Then the cLIME estimate has the form

$$f(x) = \sum_{i=1}^{2^d} a_i z_i(x),$$

where $a_i = \sum_{j=1}^{k} b_{ij} f(x_j)$ for some scalars $\{b_{ij}\}$ which do not depend on x.

Figures 1 and 2 show two example fitted surfaces for cLIME for test points inside the convex hull. The fitted surfaces are smooth, curvaceous, and are correct when the test sample is equal to any of the training samples. As per Proposition 5, one can show that these fitted surfaces have the form $f(x) = f(x_1) + (f(x_2) - f(x_1))x[1] + (f(x_3) - f(x_1))x[2] + (f(x_4) + f(x_1) - f(x_2) - f(x_3))x[1]x[2]$.

Whereas the pinv classifier weights can be re-formulated as a linear combination of the neighbor feature vectors as given by (6), Proposition 5 shows that given vertices of a unit hypercube, the cLIME classifier can be re-formulated as a linear combination of polynomial terms of the neighbor feature vectors. Thus, the cLIME classifier has a similar functional form as a support vector machine using a finite polynomial kernel.

For the LIME weights, the surface will have a similar form, but will be closer to a constant surface for larger λ.

5.2 Weighting Nearest Neighbors is a k-dimensional Problem

Consider the case $k < d$, which is likely if the problem is relatively high-dimensional. Once the k neighbors are determined, the pinv weights norm one or the regularized linear interpolation weights have only $k - 1$ degrees of freedom. This is in contrast to many classification approaches where the number of free parameters that must be fit grows with d. Thus, these nearest neighbor approaches adaptively reduce the dimensionality of the weighting problem to the subspace spanned by the neighbors.

5.3 Consistency

A trimmed variant of local linear regression weights forms a consistent classifier [3, Corollary 4]. LIME is consistent for bounded feature spaces under the standard assumptions that $k \to \infty$, $n \to \infty$, and $k/n \to 0$ [40]. The proof of LIME's consistency relies on the exponential form of the weights, and on using an exact penalty function for D, such as an ℓ_p norm (but not a squared ℓ_p norm) [40]. Classifying with the LIMRE weights and an exact penalty function D will be consistent if the weight vector v chosen for the LIMRE regularization function $R(w) = \sum_j w_j \ln(w_j/v_j)$ would itself produce a consistent classifier; the proof follows trivially from the proof in Section 5 of [40] for LIME consistency, and uses the properties that the LIMRE weights are exponential. Although we hypothesize that the LIMV classifier is consistent, this remains an open question. We hypothesize consistency for weights that solve the general regularized linear interpolation objective (16) for some large class of distortion functions D and regularization functions R.

The first-order error given in (5) asymptotically goes to zero for constrained regularized linear interpolation classifiers under standard conditions:

Proposition 6: [First-order Error Vanishes] Suppose X and $X_1, X_2, \ldots, X_n$ are drawn iid from an absolutely continuous distribution with Lebesgue density p, and that the weighted nearest-neighbor classifier acts on the $k(n)$ nearest neighbors of X. Then any weights that solve (17) will asymptotically have zero first-order error as per (5) with probability one in the limit $n \to \infty$, $k(n) = a_n \ln n$, where a_n is a sequence such that $a_n \to \infty$ and $\frac{k(n)}{n} \to 0$.

6 Experiments

We compare the different weighting methods summarized in Table 2 on simulations and benchmark datasets.

For every pair of training and test data, leave-one-out cross-validation was performed on the training data to select the best combination of k and λ or κ. If multiple combinations of k and λ gave the minimal error rate, we took the smallest k that gave the minimum error rate with any λ or κ, then λ or κ was taken to be the smallest λ or $kappa$ that gave the minimum error rate with the chosen k. For the simulations, the choice of k was $k \in \{3, 4, 5, 10, 15, \ldots, 95\}$. For the benchmark dataset experiments, the choice of k was $k \in \{1, 2, , \ldots, 20, 25, \ldots, 100, 110, \ldots, 200, 220, \ldots 400, 440, \ldots, \min(n, 800)\}$. The choice of κ for all experiments for the ridge and regularized pinv classifier was $\kappa \in \{10^{-9}, 10^{-8}, \ldots, 10^4\}$. The choices of λ for LIMV was $\lambda \in \{2^{\frac{4}{2}}, 2^{\frac{5}{2}}, \ldots, 2^{\frac{25}{2}}\}$. The choices of λ for LIME, LIMRE, and Gradient LIME was $\lambda \in \{10^{\frac{-10}{3}}, 10^{\frac{-6}{3}}, \ldots, 10^{\frac{10}{3}}\}$. To speed the cross-validation of the simulations, the cross-validation choices of λ were restricted to a subset of the choices where the smaller objective values occurred.

The regularized linear interpolation weights were computed using MOSEK's standard convex optimizer.

6.1 Simulations

The weighted classifiers were compared on two standard simulations [43] where the true class posterior for each class is a spherical Gaussian, and the classes are differentiated either by their mean and covariance (*Different Means*), or only by their covariances (*Same Means*).

For each run of each simulation, 100 random training samples and 2,000 random test samples were drawn iid from the density $f(x) = 0.5f_1(x) + 0.5f_2(x)$, where $f_1(x)$ and $f_2(x)$ are the class-conditional densities. In the Same Means simulation, the class conditional densities are Gaussian with the same means: $f_1 = \mathcal{N}(0, \Sigma)$ and $f_2 = \mathcal{N}(0, 4\Sigma)$, where Σ is the $d \times d$ identity matrix. In the Different Means simulation, the class conditional densities are Gaussian with different means: $f_1 = \mathcal{N}(0, \Sigma)$ and $f_2 = \mathcal{N}(\mathbf{1}, 4\Sigma)$, where $\mathbf{1}$ is a vector of ones. For each dimension d, each of the two simulations was run five times. Mean results for the Same Means simulation are shown in Figures 2, 3, and 4; mean and standard deviation errors for the Different Means simulation are given in Table 3.

6.2 Benchmark Dataset Results

Comparisons were run on the five benchmark data sets from the UCI Machine Learning Repository that, as of May 2006, satisfied the following requirements: had separate training and test sets, all features were Euclidean numbers, the data was not simulated, and there were no missing features. Selected results are given in Table 4, the cross-validated parameter choices are listed in Table 5, and the benchmark datasets are summarized in Table 6. Before classification,

Table 2: Nearest-Neighbor Weights Summarized for $\mathbb{X}$ with Rank k

kNN	$w_i = 1/k$
Tricube	$w_j = v_j / \sum_j v_j$, where $v_j = K(X - X_j)$ given by (13)
LIME	solves (16): $R(w) = \sum_j w_j \ln w_j$, $D = \|\cdot\|_2^2$, $w \in [0,1]^k$, $\sum_j w_j = 1$
LIMV	solves (16): $R(w) = \sum_j w_j^2$, $D = \|\cdot\|_2^2$, $w \in [0,1]^k$, $\sum_j w_j = 1$
cLIME	solves (17): $R(w) = \sum_j w_j \ln w_j$, $D = \|\cdot\|_2^2$, $w \in [0,1]^k$, $\sum_j w_j = 1$
LIMRE	solves (16): $R(w) = \sum_j w_j \ln \frac{w_j}{v_j}$, $D = \|\cdot\|_2^2$, $w \in [0,1]^k$, $\sum_j w_j = 1$ where $v_j \sim K(X - X_j)$ with $K(X - X_j)$ given by (13)
Gradient LIME	solves (16): $R(w) = \sum_j w_j \ln w_j$, $D = \|\cdot\|_{\hat{F}}^2$, $w \in [0,1]^k$, $\sum_j w_j = 1$
Gradient cLIME	solves (17): $R(w) = \sum_j w_j \ln w_j$, $D = \|\cdot\|_{\hat{F}}^2$, $w \in [0,1]^k$, $\sum_j w_j = 1$
Pinv	$w = \left(\mathbb{X}^T\mathbb{X}\right)^{-1}\mathbb{X}^T \begin{bmatrix} X \\ 1 \end{bmatrix}$ if rank$(\mathbb{X}) = k$
Pinv Norm One	$w_j = v_j - \frac{1}{k}\sum_i v_i + \frac{1}{k}$ where $v = \left(\mathbb{X}_0^T\mathbb{X}_0\right)^{-1}\mathbb{X}_0^T X$
Reg. Pinv	$w_j = v_j - \frac{1}{k}\sum_i v_i + \frac{1}{k}$ where $v = \left(\mathbb{X}_0^T\mathbb{X}_0 + \kappa I\right)^{-1}\mathbb{X}_0^T X$
Ridge	$w_j = v_j - \frac{1}{k}\sum_i v_i + \frac{1}{k}$ where $v = \tilde{\mathbb{X}}^T\left(\tilde{\mathbb{X}}\tilde{\mathbb{X}}^T + \kappa I\right)^{-1}\tilde{X}$.
Lowess	$w = \left(A\mathbb{X}^T\mathbb{X}\right)^{-1}A\mathbb{X}^T \begin{bmatrix} X \\ 1 \end{bmatrix}$ where $A[j,j] = K(X - X_j)$ as per (13).
Lowess Norm One	$w_j = v_j - \frac{1}{k}\sum_i v_i + \frac{1}{k}$ where $v = \left(A\mathbb{X}_0^T\mathbb{X}_0\right)^{-1}A\mathbb{X}_0^T X$ and $A[j,j] = K(X - X_j)$ given by (13).

training and test data were standard-normalized using the sample mean and diagonal sample covariance of the training data for each benchmark data set.

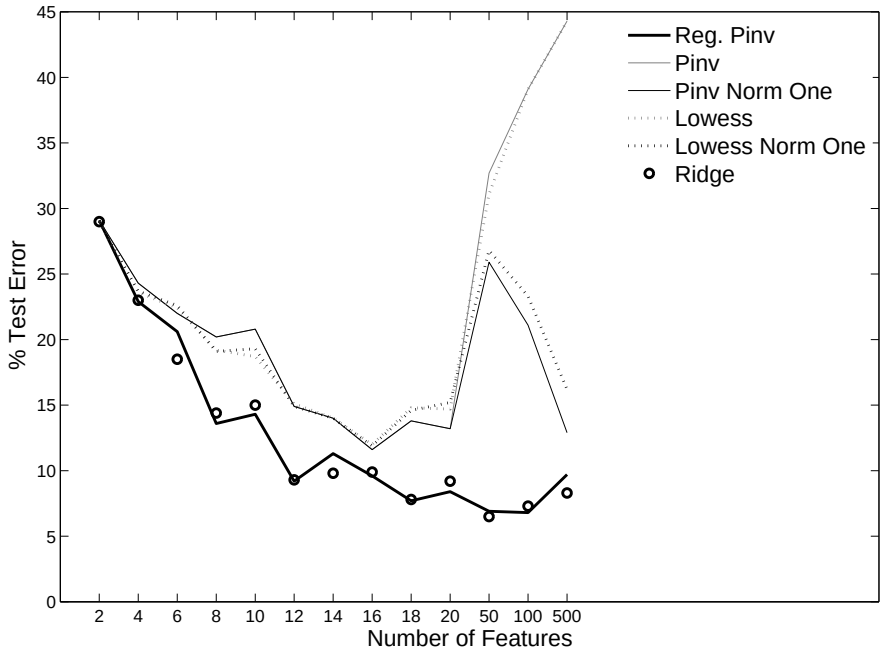

Figure 1: Average error for Same Means simulations plotted over dimension for local linear regression classifiers.

7 Discussion of Results

In this section we discuss the experimental results.

7.1 Zero-order Error

As discussed in Section 2, if the weights do not sum to one and $k < d + 1$, there will generally be zero-order error. Of all the classifiers in the experiments, only the pinv and lowess yield weights that may not sum to one, which can be compared to pinv norm one and lowess norm one, which are different only in that the offset is first subtracted off so that there is no zero-order error. Fig. 2 shows that for the Same Means simulation and $d < 20$ there is little difference between pinv (solid line) and pinv norm one (gray solid line), or between lowess (dotted line) and lowess norm one (gray dotted line). However, for higher d, the gray lines shoot away from the black lines. For $d = 500$ and $n = 100$, it must be that $k < d + 1$, and thus this case clearly shows what can happen if the zero-order error is not removed. Similar results are shown in Table 3 for the Different Means simulation.

7.2 Performance Over Dimension

Table 3 and Figs. 2, 3, and 4 show performance over dimension for the two simulations. For classifiers, there are three different effects as the dimensions increase. One: the class separability increases, making it easier to differentiate classes. Two: the density of the $n = 100$ training samples decreases, making it harder to learn. Three: the relationship of the $n = 100$ training samples actually becomes very predictable as $d \to \infty$. As shown by [44], the n_1 points from class one will occur at the vertices of an equilateral polyhedron, the n_2 points from class two will occur at the vertices of another equilateral polyhedron, and points from the two classes will also be equally far away. We hypothesize that

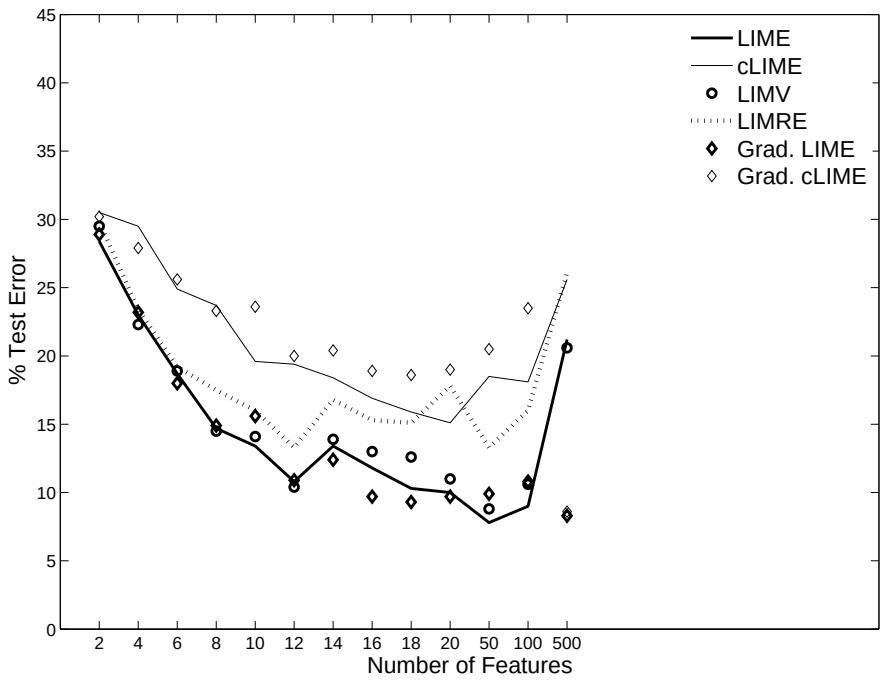

Figure 2: Average error for Same Means simulations plotted over dimension for regularized linear interpolation classifiers.

for $d = 500$ and $n = 100$ these effects are already felt, and that this high-dimensional structure can lower estimation variance for some classifiers by lowering the risk of overfitting some cross-validated parameters.

For high dimensions in the Same Means simulation, the points from class 1 are generally closest to other class 1 points, but the points from class 2 are on average closer to class 1 points than to class 2 points. Thus the 50% asymptotic error rate of k-NN and tricube in this case is because these classifiers classify all points as class 1 points. In the Different Means simulation errors occur for both classes.

In general, one expects to achieve the lowest learning error by using a statistical learning technique that is best fit to the true class conditional distributions. For the two simulations, the "correct" classifier would be quadratic discriminant analysis (QDA), which models each class conditional distribution as a Gaussian with different means and different covariances. In fact, QDA using the standard maximum likelihood parameter estimation does not perform well for high-dimensional runs of the simulation because the high ratio of parameters to estimate to data samples causes ill-posed estimates. For example, over ten different draws of 100 training samples in $d = 500$ dimensions for the Same Means simulation, the QDA error was 47%, whereas the regularized linear interpolation and ridge weighted nearest-neighbor classifiers achieved zero error. (However, using a Bayesian QDA can also achieve approximately zero error for this simulation [45].)

7.3 Cross-validation and Overfitting of the Regularization Parameters

It is difficult to choose a set of parameters for cross-validation, with too dense a set of choices leading to overfitting, and too sparse a set not able to exploit the full flexibility of the model. The simulations show that cross-validated regularization of either the weights or the linear model coefficients can perform significantly better if the training and test data are truly iid, with the largest differences for the Same Means simulation results shown in Figs. 2 and 3. However, of the unregularized/regularized pairs (LIME/cLIME, gradient LIME/gradient cLIME, pinv norm one/regularized pinv, and pinv norm one/ridge), only ridge performed consistently better than its unregularized counterpart on the benchmark datasets, which are not truly iid and thus pose a greater risk for overfitting. We hypothesize that the local standard

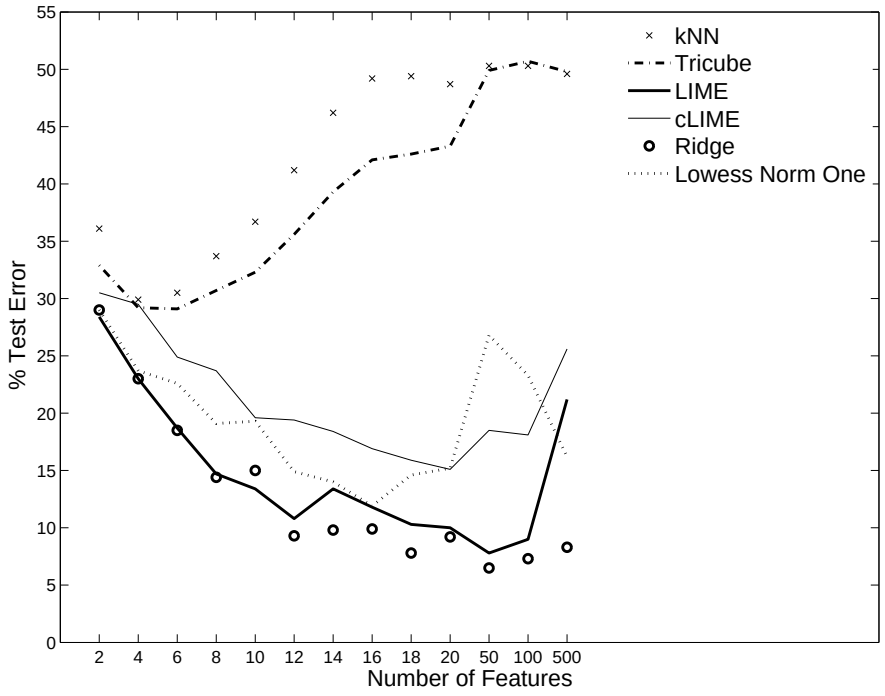

Figure 3: Average error for Same Means simulations plotted over dimension for selected nearest-neighbor classifiers.

Table 3: Different Means Simulation: average error, with the standard deviation of the errors in parentheses for five runs of the simulation for each dimension.

	Dimension									
	5		20		50		100		500	
kNN	20.4	*(2.8)*	33.4	*(4.7)*	50.6	*(0.9)*	50.4	*(1.6)*	50.2	*(0.5)*
Tricube	19.8	*(2.1)*	26.0	*(3.7)*	45.8	*(2.3)*	49.4	*(1.9)*	50.1	*(0.6)*
LIMV	13.6	*(1.2)*	4.2	*(0.7)*	1.3	*(0.7)*	**0.5**	*(0.3)*	**0**	*(0)*
LIME	14.4	*(2.8)*	4.0	*(0.8)*	**1.1**	*(0.5)*	**0.5**	*(0.3)*	**0**	*(0)*
cLIME	18.2	*(1.4)*	5.6	*(1.9)*	1.5	*(0.8)*	0.6	*(0.3)*	**0**	*(0)*
LIMRE	15.4	*(2.3)*	5.1	*(0.7)*	1.5	*(0.7)*	**0.5**	*(0.3)*	**0**	*(0)*
Gradient LIME	**13.4**	*(0.7)*	4.1	*(1.8)*	5.9	*(1.1)*	1.3	*(0.5)*	**0**	*(0)*
Gradient cLIME	21.0	*(2.4)*	6.1	*(0.5)*	2.3	*(0.9)*	0.8	*(0.5)*	**0**	*(0)*
Pinv	18.4	*(1.0)*	5.2	*(3.6)*	5.6	*(2.3)*	18.7	*(3.7)*	22.0	*(1.5)*
Pinv Norm One	18.4	*(1.0)*	5.2	*(3.6)*	5.6	*(2.3)*	2.6	*(0.2)*	**0**	*(0)*
Lowess	17.4	*(2.1)*	5.4	*(2.5)*	9.7	*(7.7)*	18.7	*(3.7)*	22.0	*(1.5)*
Lowess Norm One	16.6	*(1.4)*	5.4	*(2.5)*	6.9	*(2.2)*	3.2	*(0.4)*	0.2	*(0.3)*
Regularized Pinv	15.6	*(1.1)*	5.3	*(3.5)*	2.0	*(1.7)*	1.1	*(1.1)*	**0**	*(0)*
Ridge	15.2	*(3.0)*	**3.5**	*(0.6)*	2.5	*(2.6)*	1.3	*(1.3)*	0.1	*(0.1)*

normalization of the training feature vectors helps homogenize the different neighbor-test point geometries, and that the increased homogeneity reduces the risk of overfitting relative to the other regularization methods.

While a different set of cross-validation choices for the regularization parameters may have performed better, it

Table 4: Test error % on UCI benchmark datasets

	Image Seg.	Isolet	Opt. Digits	Pen Digits	Vowel
kNN	12.6	9.1	3.4	2.5	49.4
Tricube	15.7	6.7	3.3	2.4	49.4
LIME	9.9	**4.7**	2.3	2.0	49.4
cLIME	9.9	**4.7**	2.5	1.9	49.4
Pinv Norm One	9.4	6.9	2.4	1.7	45.9
Lowess Norm One	11.7	7.4	1.9	**1.2**	46.1
Regularized Pinv	10.4	6.7	2.1	1.5	45.9
Ridge	**8.7**	5.7	**1.7**	1.5	**40.5**

Table 5: Cross-validated Parameters for UCI benchmark datasets

	Image Seg.		Isolet		Opt. Digits		Pen Digits		Vowel	
	k	λ	k	λ	k	λ	k	λ	k	λ
kNN	6		25		3		3		1	
Tricube	55		110		10		19		1	
LIME	45	10^{-5}	75	10	220	10^{-1}	35	$10^{-\frac{4}{3}}$	1	
cLIME	45		45		140		35		1	
Pinv Norm One	3		35		240		70		6	
Lowess Norm One	180		35		640		640		7	
Regularized Pinv	45	10^{-4}	95	10^{2}	30	10^{1}	60	10^{-1}	6	10^{-9}
Ridge	11	10^{0}	300	10^{2}	120	10^{1}	70	10^{0}	11	10^{-1}

will never be possible a priori to choose an optimal balance between overfitting and model flexibility.

In general, the cross-validation made the most positive difference on the Same Means simulation. We hypothesize that this is because in the Same Means simulation all points have the same relationship to the decision boundary, whereas in the Different Means simulation the classes are more separable, but some points are far from the decision boundary and some points are close, thus one expects a greater number of points may be needed to train the classifier, and there is likely more variability in the optimal regularization parameter with respect to different test points.

Table 6: Information About the Benchmark Datasets

	Number of Classes	Number of Features	Number of Total Samples (Training/Test)
Image Seg.	7	19	210/2,100
Isolet	26	617	6,238/1,559
Opt. Digits	10	64	3,823/1,797
Pen Digits	10	16	7,494/3,498
Vowel	11	10	528/462

7.4 Distance-decay weights

The experiments included three distance-decay comparisons: kNN to tricube, pinv to lowess, and LIME to LIMRE, with all comparisons using the tricube kernel given in (13) to implement the distance-decay.

The tricube kernel performs slightly better than kNN throughout. There is no statistically significant difference between lowess and pinv on the simulations, but lowess achieved relatively low error on the Opt. Digits and Pen Digit

datasets. These results agree with the common intuition that distance-decay can be of some help, and is unlikely to hurt the results.

However, implementing distance-decay by regularization had different results: LIMRE performed consistently worse than LIME on the simulations, and since there was no difference in cross-validation parameters, we concluded that it is not a reasonable alternative to LIME.

7.5 Different Regularizations

For the regularized linear interpolation classifiers, the entropy regularization of LIME and the variance regularization of LIMV performed similarly. Theoretically, gradient LIME should perform better than LIME in terms of the first-order error. In practice, Gradient LIME performance did not improve over LIME. The reason could be that our ridge-based estimate of $\hat{F}$ was not accurate enough for gradient LIME to be an improvement. However, we hypothesize that the primary reason is that the first-order error provided by LIME may already be small enough to ensure correct classification, such that further improvement of the first-order error that could be afforded by Gradient LIME would not improve error rates. This hypothesis is supported, for example, by comparing the simulation results for LIME and pinv (see Fig. 2 and Fig. 3). For small d, pinv is guaranteed to have zero first-order error, and yet does not perform as well as LIME. Thus there is little reason to believe that further reducing the first-order error for LIME (by using a gradient LIME with exact F) would be helpful for classification.

For the local linear classifiers, we compared regularizing the hyperplane slope coefficients $\beta^T\beta$ to regularizing the weights $w^T w$. Given the same regularization choices, the two regularizations performed similarly better than pinv norm one on the simulations, where the training and test data were truly iid. As discussed in Sec. 7.3, only ridge performed consistently better than its unregularized counterpart on the benchmark datasets, which we hypothesize is largely due to the pre-processing of the neighborhood feature vectors which would have increased homogeneity of the regularization's impact on different test sample draws.

7.6 Neighborhoods and Locality

Nearest neighbor methods are often called "local classifiers," but in fact, the cross-validated neighborhood sizes k were often extremely non-local. For the simulations, the cross-validated neighborhood size was usually $k = 1$ for the kNN and tricube classifier for $d > 5$. For the other classifiers, k increased as d increased. With $d = 500$, both the locally linear classifiers and the regularized linear interpolation classifiers choose a k in the range $60 - 95$ out of $n = 100$ possible training samples. This suggests that the effect of using the neighborhood was to exclude the farthest points, rather than to include only "near" points.

Usually nearest neighbor methods are run with choices of k from one to twenty, while n may be in the thousands. For the benchmark datasets, we allowed choices of k up to 800 neighbors. As shown in Table 5, lowess chose $k = 640$ by cross-validation on Pen Digits and Opt. Digits, and produced relatively low errors. For Pen Digits, $k = 640$ out of $n = 7494$ training samples in $d = 16$ feature dimensions is roughly 10% of the data, which corresponds roughly to the number of training samples in each of the ten classes. For Opt Digits, all of the locally linear and regularized linear interpolation classifiers except for regularized pinv chose $k > 100$. Lowess's choice of $k = 640$ neighbors for Opt Digits is almost 20% of the training samples. Similarly, many of the methods choose roughly 20% of the training samples for Image Segmentation, with lowess choosing $6/7$ of the training samples.

Cross-validating the neighborhood parameter k is likely to suffer from some overfitting, and raises the larger question of how to define the neighborhood. Although using k nearest neighbors is a classic approach, there is little reason to believe that each test sample is best served by the same neighborhood size, or that the neighborhood should be parameterized by the number of neighbors. For linear interpolation, [18] suggested using the set of training samples whose Voronoi cells are adjacent to the test point's Voronoi cell given a Voronoi tessellation of the training points plus the test point. Sibson termed these the *natural neighbors*. [16] has shown significant reductions in error for color management using Sibson's natural neighbors for local linear regression over a three-dimensional feature space. [16] also proposed an adaptive neighborhood which uses the smallest k neighbors that minimize the distance between the convex hull of the training samples and the test point. These *enclosing neighborhooods* can enable zero first-order error as described in Proposition 3, but not if the test point is outside the convex hull of the n training points, as is often true for practical classification problems. Optimal neighborhood definitions for nearest neighbors classifiers is an open question.

The ability of lowess to work well given a large percentage of the training data suggests that using all the training samples but with a fixed, aggressive distance-decay weighting could also be effective, and would remove the need to cross-validate k.

7.7 Local Linear Regression Compared to Regularized Linear Interpolation

The simulation and experimental results show that both the local linear regression and the regularized linear interpolation classification methods can work well, and differences in mean results may be attributable to differences in cross-validation and in the data pre-processing for ridge. Using the Wilcoxon paired test for statistical significance with a significance level of .05, the simulation results for $d < 50$ were not statistically significantly different for any of the comparable pairs LIME v. Ridge, LIME v. regularized pinv, or cLIME v. pinv norm one. For $d = 500$ for the Same Means simulation, ridge and regularized pinv were statistically significantly better than LIME, but not statistically significantly different than Gradient LIME. For the Different Means simulation and $d = 50$ and $d = 100$, cLIME and gradient cLIME were statistically significantly better than their counterparts pinv norm one and lowess norm one (this is a relevant comparison as none of these algorithms has a cross-validated regularization parameter). On the benchmark datasets, ridge beat LIME on the four lower-dimensional datasets. For Isolet, which has about ten times more features than the next highest-dimensional dataset (Opt. Digits), LIME and cLIME performed the best.

Another difference suggested by the simulations data and the theory is that the regularized linear interpolation classifiers appear to have less variance than the corresponding local linear regression classifiers. For example, comparing the standard deviations of the errors in Table 3 for cLIME and Gradient cLIME to that of pinv norm one and lowess norm one, there is generally less error variance. Lower estimation variance is supported by the theory because the regularized linear interpolation classifiers have the additional constraint that $w \in [0, 1]^k$, and constraints will lower the variance.

Based on the overall similar performances between these two classes of classifiers, the choice to use either a pinv classifier or the LIME/cLIME classifiers or should be driven by concerns other than performance, such as computational efficiency and whether one wishes to interpret the outputs as class probability estimates.

7.8 Computation

The local linear classifiers all have closed-form solutions, while the regularized linear interpolation classifiers require convex optimization. However for large values of k and/or d, it may be faster to run a convex optimization than to invert the large matrices in the closed-form solution. The convex optimization can also trade-off accuracy of the weights for speed, for example by adjusting the number of iterations in the numerical solver. Lastly, ridge requires a $d \times d$ matrix inversion, which for relatively high-dimensional feature spaces may be infeasible or more computationally intensive than the $k \times k$ matrix inversion required for regularized pinv.

8 Conclusions and Open Questions

The main contributions of this paper are the analysis of the first-order error for weighted nearest-neighbor classifiers in general, zero-order and first-order analysis for local linear regression classifiers, and first-order analysis for a new class of regularized linear interpolation classifiers. In addition to the analysis, we established through simulations and experiments that weighted nearest-neighbors whose weights are designed to minimize first-order error can significantly outperform simple distance-decay weights such as the tricube kernel. In fact, for the Different Means simulation, tricube and uniform weights are little better than coin-flipping for $d = 50$ dimensions and higher, but the methods which take into account both zero-order and first-order error can capture the class separability, with all of the regularized linear interpolation classifiers achieving zero errors by $d = 500$ dimensions. Similarly on the benchmark datasets, the first-order error classifiers produced up to 50% lower error than the simpler uniform and tricube weights.

Within the class of local linear regression classifiers, we showed it was important to take into account the response variable offset, so that the final weights would sum to one. We compared two different regularization techniques and pre-weighting the errors, and showed that with standard data pre-processing for ridge and our set of cross-validation parameters, consistently low classification errors could be achieved.

For the class of regularized linear interpolation classifiers, we compared three regularization strategies, and showed that minimizing relative entropy was not as effective as maximizing entropy or minimizing variance, which performed

similarly. Based on our first-order error analysis, we proposed the gradient LIME weights, but given that the local gradient had to be estimated, we were not able to show an improvement over LIME. Also, we compared unregularized versions of LIME and gradient LIME to regularized versions, and with the given cross-validation parameters, showed that on benchmark datasets where the data was not iid, the regularization did not add value. Because the regularization does show a statistically significant improvement on the iid simulations, we believe that a sparser cross-validated regularization could still prove effective.

This work highlights a number of issues and open questions, such as the difficulty of cross-validation, the size and definition of neighborhoods, whether local learning is best understood as learning from near samples or as rejecting far samples, and whether, for high-dimensional problems, a closed-form solution will be computationally more effective than an iterative solution.

Acknowledgments

The authors thank Santosh Srivastava and Rob Tibshirani for helpful discussions.

Appendix 1: Proofs

Proof of Proposition 1

Consider the first case of the proposition. Let N be a vector with jth component equal to N_j, and let w^* solve (16). Given any of the three cases of $f(X_j)$ described in the proposition,

$$\mathrm{var}(\hat{f}(x,R)) = \mathrm{var}({w^*}^T N) \overset{(a)}{=} E[({w^*}^T N)^2] \overset{(b)}{=} {w^*}^T E[NN^T] w^* \overset{(c)}{=} \sigma^2 {w^*}^T w^*,$$

where (a) and (b) follow because w^* is deterministic in this scenario, w^* is independent of $f(x_j)$, and $E[N] = \vec{0}$, and (c) follows because $E[NN^T] = \sigma^2 I$ Thus, the w^* that minimizes $\mathrm{var}(\hat{f}(x,R))$ is the w^* that minimizes $w^T w$. Given that the proposition restricts the consideration to weights w that yield fixed distortion D_0, the weights that solve (16) with $R(w) = w^T w$ minimize the estimation variance. The proof of the second case follows similarly, where $N_j = a^T \vec{N}_j$, such that $E[N_j] = 0$ and $E[N_j N_j^T] = \sigma^2 a^T a$.

Proof of Proposition 2

For scalar features, minimizing the first-order error given by (5) does not depend on F, and solving (17) minimizes (5).

Proof of Proposition 3

By assumption there exists some set of positive normalized weights a such that $\sum_{j \in E} a_j X_j = X$ and $\sum_{j \in E} a_j = 1$ so that zero first-order error is feasible, and because of the constraint in (17), any constrained regularized linear interpolation weights must achieve this minimum.

Proof of Proposition 4

Let $H(w) = -\sum_{j \in \mathcal{J}} w_j \ln w_j$, let u be the uniform weight vector. By construction, the LIME weights w^* minimize objective (16) with $R(w) = H(w)$, thus

$$\| \sum_j w_j^* X_j - X \|_F^2 - \lambda H(w^*) \leq \| \sum_j u_j X_j - X \|_F^2 - \lambda H(u).$$

Uniform weights over a discrete set of events uniquely maximize entropy, thus it must be that $H(u) - H(w^*) \geq 0$, so that the the first-order errors satisfy

$$\| \sum_j w_j^* X_j - X \|_F^2 \leq \| \sum_j u_j X_j - X \|_F^2.$$

with equality if and only if the LIME weights are uniform. If $d = 1$, then the above proof works with $D = \|\tilde{X}\|^2$. The proof for the LIMV weights follows similarly, using the fact that u uniquely minimizes variance out of all positive normalized weight vectors. The proof for the LIMRE weights follows similarly, using the fact that v uniquely minimizes the relative entropy with respect to itself out of all positive normalized weight vectors.

Proof of Proposition 5

Assuming the training samples lie at the vertices of a unit hypercube, the cLIME weights have a product form [23, Theorem 2],

$$w_j^*(x) = \prod_{m=1}^{d} |1 - x_i[m] - x[m]|.$$

The weights do not depend on shifts; shift the training and test samples so that x_1 lies at the origin, and then $w_1^* = \prod_{m=1}^{d}(1 - x[m])$, since it was assumed that $x \in [0,1]^d$. Expanding the product and re-writing in terms of $g_i(x)$, one arrives at $w_1(x) = \sum_i^{2^d} b_{i1} g_i(x)$, where $\{b_{i1}\}$ are scalar coefficients that do not depend on x. Similarly, the other weights can be expanded and expressed as $w_j(x) = \sum_i^{2^d} b_{ij} g_i(x)$. Then

$$f(x) = \sum_{j=1}^{2^d} w_j f(x_j) = \sum_{j=1}^{2^d}\sum_{i=1}^{2^d} b_{ij} g_i(x) f(x_j) = \sum_{i=1}^{2^d} \left(\sum_{j=1}^{2^d} b_{ij} f(x_j) \right) g_i(x).$$

Let $a_i = \sum_{j=1}^{2^d} b_{ij} f(x_j)$ to complete the proof.

Proof of Proposition 6

Under the proposition's assumptions, [23] Theorem 3 establishes that with probability one the test point X lies in the convex hull of its k nearest neighbors. Applying Proposition 3 completes the proof.

References

[1] E. Fix and J. L. Hodges, "Discriminatory analysis, nonparametric discrimination: Consistency properties," *Technical Report 4*, 1951, uSAF School of Aviation Medicine, TX.

[2] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Information Theory*, vol. 13, pp. 21–27, 1967.

[3] C. J. Stone, "Consistent nonparametric regression," *The Annals of Statistics*, vol. 5, no. 4, pp. 595–645, 1977.

[4] L. Devroye, L. Gyorfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. New York: Springer-Verlag Inc., 1996.

[5] W. Lam, C. Keung, and D. Liu, "Discovering useful concept prototypes for classification based on filtering and abstraction," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1075–1090, August 2002.

[6] K. Fukunaga and L. Hostetler, "Optimization of k-nearest neighbor density estimates," *IEEE Trans. on Information Theory*, vol. 19, pp. 320–326, 1973.

[7] R. Short and K. Fukunaga, "The optimal distance measure for nearest neighbor classification," *IEEE Trans. on Information Theory*, vol. 27, no. 5, pp. 622–627, 1981.

[8] K. Fukunaga and T. Flick, "An optimal global nearest neighbor metric," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 314–318, 1984.

[9] J. Myles and D. Hand, "The multi-class metric problem in nearest neighbour discrimination rules," *Pattern Recognition*, vol. 23, pp. 1291–1297, 1990.

[10] J. H. Friedman, "Flexible metric nearest neighbor classification," *Technical Report, Stanford University, CA*, 1994.

[11] T. Hastie and R. Tibshirani, "Discriminative adaptive nearest neighbour classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, pp. 607–615, 1996.

[12] C. Domeniconi, J. Peng, and D. Gunopulos, "Locally adaptive metric nearest neighbor classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1281–1285, 2002.

[13] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *NIPS*, pp. 480–483, 2006.

[14] R. Paredes and E. Vidal, "Learning weighted metrics to minimize nearest-neighbor classification error," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1100–1110, 2006.

[15] E. K. Garcia, S. Feldman, M. R. Gupta, and S. Srivastava, "Completely lazy learning," *IEEE Trans. on Knowledge and Data Engineering*, 2009.

[16] E. M. Chin, E. K. Garcia, and M. R. Gupta, "Color management of printers by regression over enclosing neighborhoods," *Proc. of the IEEE Intl. Conf. on Image Processing*, 2007.

[17] R. Nock, M. Sebban, and D. Bernard, "A simple locally adaptive nearest neighbor rule with application to pollution forecasting," *Intl. Journal of Pattern Recognition and Artificial Intelligence*, vol. 17, no. 8, pp. 1–14, 2003.

[18] R. Sibson, *Interpreting multivariate data*. John Wiley, 1981, ch. : A brief description of natural neighbour interpolation, pp. 21–36.

[19] K. Clarkson, *Nearest-neighbor methods in learning and vision*. Cambridge, MA: MIT Press, 2005, ch. : Nearest-neighbor searching and metric space dimensions, pp. 15–60.

[20] T. Liu, A. W. Moore, and A. Gray, "New algorithms for efficient high-dimensional nonparametric classification," *Journal of Machine Learning Research*, vol. 7, pp. 1135–1158, 2006.

[21] J. McNames, "A fast nearest-neighbor algorithm based on a principal axis search tree," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 9, pp. 964–976, 2001.

[22] 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 Intelligence*, vol. 8, no. 6, pp. 761–766, 1986.

[23] M. R. Gupta, R. M. Gray, and R. A. Olshen, "Nonparametric supervised learning with linear interpolation and maximum entropy," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, pp. 766–781, 2006.

[24] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti, "Similarity-based Classification: Concepts and Algorithms," *Journal of Machine Learning Research*, vol. 10, pp. 747–776, 2009.

[25] T. Hastie and C. Loader, "Local regression: automatic kernel carpentry," *Statistical Science*, vol. 8, no. 2, pp. 120–143, 1993.

[26] M. R. Gupta, L. Cazzanti, and S. Srivastava, "Minimum expected risk estimates for nonparametric neighborhood classifiers," *Proc. of the IEEE Workshop on Statistical Signal Processing*, 2005.

[27] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. New York: Springer-Verlag, 2001.

[28] M. R. Gupta, S. Srivastava, and L. Cazzanti, "Minimum expected risk estimation for near-neighbor classification," *University of Washington Electrical Engineering Technical Report 2006-0006*, 2006.

[29] C. Loader, *Local Regression and Likelihood*. New York: Springer, 1999.

[30] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2006.

[31] A. E. Hoerl and R. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, pp. 55–67, 1970.

[32] L. Bottou and V. Vapnik, "Local learning algorithms," *Neural Computation*, vol. 4, pp. 888–900, 1992.

[33] I. Frank and J. H. Friedman, "A statistical view of some chemometrics regression tools," *Technometrics*, vol. 35, no. 2, pp. 109–148, 1993.

[34] G. Sharma, *Digital Color Handbook*. CRC Press, 2003.

[35] J. Chang, J. Allebach, and C. Bouman, "Sequential linear interpolation of multidimensional functions," *IEEE Trans. on Image Processing*, vol. 6, no. 9, pp. 1231 – 1245, 1997.

[36] M. R. Gupta, M. P. Friedlander, and R. M. Gray, "Maximum entropy classification applied to speech," *Proc. of Asilomar Systems and Signals Conference*, pp. 1480–1483, 2000.

[37] D. B. O'Brien, M. R. Gupta, and R. M. Gray, "Analysis and classification of internal pipeline images," *Proc. of the IEEE Intl. Conf. on Image Proc.*, pp. 577–580, 2003.

[38] L. Cazzanti, M. R. Gupta, L. Malmstrom, and D. Baker, "Quality assessment of low free-energy protein structure predictions," *Proc. of the IEEE Workshop on Machine Learning for Signal Processing*, pp. 375–380, 2005.

[39] B. Smith, L. Atlas, and M. R. Gupta, "Beamforming alternatives for multi-channel transient acoustic event classification," *Proc. of the IEEE Intl. Conf. on Acoustics, Speech and Signal Processing*, 2007.

[40] M. P. Friedlander and M. R. Gupta, "On minimizing distortion and relative entropy," *IEEE Trans. on Information Theory*, vol. 52, no. 1, pp. 238–245, 2006.

[41] M. R. Gupta, "Custom color enhancements by statistical learning," *IEEE Intl. Conf. on Image Proc.*, pp. 968–971, 2005.

[42] M. R. Gupta, E. K. Garcia, and E. M. Chin, "Adaptive local linear regression with application to printer color management," *To appear, IEEE Trans. on Image Processing*, 2007.

[43] J. H. Friedman, "Regularized discriminant analysis," *Journal American Stat. Assoc.*, vol. 84, pp. 165–175, 1989.

[44] P. Hall, J. S. Marron, and A. Neeman, "Geometric representation of high dimension, low sample size data," *Journal Royal Statistical Society B*, vol. 67, pp. 427–444, 2005.

[45] S. Srivastava, M. R. Gupta, and B. A. Frigyik, "Bayesian quadratic discriminant analysis," *Journal of Machine Learning Research*, vol. 8, pp. 1287–1314, 2007.