The Curse of Dimensionality for Local Kernel Machines

Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux
Dept. IRO, Université de Montréal
P.O. Box 6128, Downtown Branch, Montreal, H3C 3J7, Qc, Canada
{bengioy,delallea,lerouxni}@iro.umontreal.ca

Technical Report 1258
Département d'Informatique et Recherche Opérationnelle

March 2nd, 2005
Revised May 11th, 2005

Abstract

We present a series of theoretical arguments supporting the claim that a large class of modern learning algorithms based on local kernels are sensitive to the curse of dimensionality. These include local manifold learning algorithms such as Isomap and LLE, support vector classifiers with Gaussian or other local kernels, and graph-based semi-supervised learning algorithms using a local similarity function. These algorithms are shown to be local in the sense that crucial properties of the learned function at x depend mostly on the neighbors of x in the training set. This makes them sensitive to the curse of dimensionality, well studied for classical non-parametric statistical learning. There is a large class of data distributions for which non-local solutions could be expressed compactly and potentially be learned with few examples, but which will require a large number of local bases and therefore a large number of training examples when using a local learning algorithm.

1 Introduction

A very large fraction of the recent work in statistical machine learning has been focused on so-called kernel machines, which are non-parametric learning algorithms in which the learned function is expressed in terms of a linear combination of kernel functions applied on the training examples:

$$f(x) = b + \sum_{i=1}^{n} \alpha_i K(x, x_i) \tag{1}$$

where optionally a bias term b is added, $D = \{x_1, \ldots, x_n\}$ are training examples (without the labels, in the case of supervised learning), the α_i's are scalars chosen by the learning algorithm using labels $\{y_1, \ldots, y_n\}$, and $K(\cdot, \cdot)$ is the kernel function, a symmetric function (generally expected to be positive definite). We refer to the $n \times n$ matrix M with entries $M_{ij} = K(x_i, x_j)$ as the Gram matrix. A typical kernel function is the Gaussian kernel,

$$K(u, v) = e^{-\frac{1}{\sigma^2}||u-v||^2}, \tag{2}$$

with hyper-parameter σ (the width) controlling how local the kernel is. The Gaussian kernel is part of a larger family of kernels which we call local kernels and discuss in section 5.

Examples of kernel-based nonlinear manifold learning algorithms include Locally Linear Embedding (LLE) (Roweis and Saul, 2000), Isomap (Tenenbaum, de Silva and Langford, 2000), kernel Principal Component Analysis (PCA) (Schölkopf, Smola and Müller, 1998), Laplacian Eigenmaps (Belkin and Niyogi, 2003), and Manifold Charting (Brand, 2003). This framework also includes spectral clustering algorithms (see (Weiss, 1999) for more references). For these algorithms, f represents the embedding function, which maps an example x to one of the coordinates of its low-dimensional representation embedding (and a separate set of α_i's is used for each dimension of the embedding). As shown in (Bengio et al., 2004), the α_i correspond to entries in an eigenvector of the Gram matrix, divided by the corresponding eigenvalue (following the Nyström formula (Nyström, 1928; Baker, 1977; Williams and Seeger, 2000)). Note that these algorithms employ a special kind of kernel which is data-dependent (Bengio et al., 2004), but the arguments presented in sections 5, 6 and 8 also apply.

Among the statistical classifiers most widely studied in the recent years is the Support Vector Machine (SVM) (Boser, Guyon and Vapnik, 1992; Cortes and Vapnik, 1995; Schölkopf, Burges and Smola, 1999), in which f above is the discriminant function of a binary classifier, i.e. the decision function is given by the sign of f. Since this is a supervised learning algorithm, the training data includes a set of +1 or -1 class labels $\{y_1, \ldots, y_n\}$ for training examples. Results in section 4 apply to SVMs and other kernel machines with Gaussian kernels, and a more general conjecture is developed in section 8 based on the *local-derivative* notion introduced in section 6.

Another class of kernel algorithms that is discussed here are the non-parametric graph-based semi-supervised algorithms of the type described in recently proposed papers (Zhu, Ghahramani and Lafferty, 2003; Zhou et al., 2004; Belkin, Matveeva and Niyogi, 2004; Delalleau, Bengio and Le Roux, 2005). They can be intuitively understood as performing some kind of smoothing or label propagation on the empirical graph defined by the examples (nodes) and a similarity function (e.g. a kernel) between pairs of examples. The results in section 7 are specifically tailored to such algorithms, and show that the number of required labeled examples grows linearly with a measure of the amount of local variation of the predictor required to reach a given error level.

The basic ideas behind the arguments presented in this paper are simple. One class of arguments relies on showing that some important property of $f(x)$ is mostly determined by the neighbors of x in the training set. If one imagines tiling the space with such neighborhoods, the required number of neighborhoods (hence of training examples) could grow exponentially with the dimensionality of the data (or of the manifold on which they live). One issue in this respect is the size of these neighborhoods, and we get inspiration from the classical bias-variance trade-off argument for classical non-parametric models: if we make the regions smaller, bias is reduced (more complex functions can be represented) but variance is increased (not enough data are used to determine the value of $f(x)$ around x, so $f(x)$ becomes less stable).

Another class of arguments considers "apparently complicated" target functions, in the sense that they vary a lot across space, although there might exist simple and compact representations for them that could not be discovered using a purely local representation (eq. 1). For these arguments one attempts to find lower bounds on the number of examples required in order to learn the target function. These arguments show that whatever the method used to estimate the α_i's, one must have a large number of them in order to approximate the function at a given set of points (larger than the size of the training set), i.e. in order to get meaningful generalization.

In general, what one should keep in mind is that what matters is not the dimension of the data or of the manifold near which they are found, but rather the "apparent complexity" of the function that we are trying to learn (or one that would yield an acceptable error level). By "apparent complexity" we mean a measure of the number of "variations" of that function. One way to formalize this notion, used in Proposition 7.1, is the number of regions with constant sign of the predictor.

2 The Curse of Dimensionality for Classical Non-Parametric Models

The **curse of dimensionality** has been coined by Bellman (Bellman, 1961) in the context of control problems but it has been used rightfully to describe the poor generalization performance of local non-parametric estimators as the dimensionality increases.

2.1 The Bias-Variance Dilemma

This problem has often been studied by considering the classical bias-variance analyses of statistical estimators. To keep the analysis simple, one usually only computes the conditional bias and conditional variance, in the case of supervised learning, i.e. given the x_i's of the training set, and integrating only over the y_i's. **Bias** of the estimator is the expected difference between the estimator and the target function, while **variance** of the estimator is the expected squared difference between the estimator and its expected value (in both cases the expectations are over the training set or over the y_i's only). The expected mean squared error of the estimator can be shown to be equal to the sum of variance and squared bias (plus the irreducible component of variation of Y given X, i.e. noise).

Two classical non-parametric estimators are worth discussing here. The first one is the k-nearest-neighbor estimator. It can be cast as a kernel machine (eq. 1) when K is allowed to be data-dependent: $\alpha_i = y_i$ and $K(x, x_i)$ is $1/k$ if x_i is one of the k nearest neigbors of x in D, 0 otherwise. The second one is the Nadaraya-Watson estimator (also known as the Parzen windows estimator), in which again $\alpha_i = y_i$ and $K(x, x_i)$ is a normalized kernel (i.e. also data-dependent), e.g.

$$K(x, x_i) = \frac{\mathcal{K}(x, x_i)}{\sum_{i=1}^{n} \mathcal{K}(x, x_i)}.$$

Since both estimators have the same form, much of the analysis can be shared. Let $t(x)$ be the target function that we are trying to learn, with $E[Y|X = x] = t(x)$. The conditional bias is simply

$$\text{conditional bias}(x) = E[Y|X = x] - \sum_{i=1}^{n} E[Y|X = x_i]K(x, x_i). \tag{3}$$

Clearly, the more local the kernel (i.e. $K(x, x_i)$ is nearly 0 except for x_i very close to x), the smaller the bias (note that we consider here kernels K such that $\sum_i K(x, x_i) = 1$). Assuming that $Var[y_i|x_i] = v$ does not depend on x_i the conditional variance of the estimator is

$$\text{conditional variance}(x) = v \sum_{i=1}^{n} K(x, x_i)^2.$$

For example, with the k-nearest neighbor estimator,

$$\text{conditional variance}(x) = \frac{v}{k}.$$

Clearly, the way to reduce variance is to decrease the kernel's locality (e.g. increase σ in a Gaussian kernel or k in a k-nearest neighbor estimator), which increases the effective number of examples used to obtain a prediction at x. But this also increases bias, by making the prediction smoother (possibly too smooth). Since total error involves the sum of squared bias and variance, one should choose the kernel hyper-parameter (e.g. k or σ) to strike the best balance (hence the *bias-variance dilemma* (Geman, Bienenstock and Doursat, 1992)).

2.2 Dimensionality and Rate of Convergence

A nice property of classical non-parametric estimators is that one can prove their convergence to the target function as $n \to \infty$, i.e. these are consistent estimators. Considering the above simplified exposition on bias and variance, one obtains consistency by appropriately varying the hyper-parameter that controls the locality of the estimator as n increases. Basically, the kernel should be allowed to become more and more local, so that bias goes to zero, but the "effective number of examples" involved in the estimator at x,

$$\frac{1}{\sum_{i=1}^{n} K(x, x_i)^2}$$

(equal to k for the k-nearest neighbor estimator) should increase as n increases, so that variance is also driven to 0. For example one obtains this condition with $\lim_{n\to\infty} k = \infty$ and $\lim_{n\to\infty} \frac{k}{n} = 0$ for the k-nearest neighbor. Clearly the first condition is sufficient for variance to go to 0 and the second is sufficient for the bias to go to 0 (since $\frac{k}{n}$ is proportional to the volume of space around x which contains the k nearest neighbors). Similarly, for the Nadarya-Watson estimator with bandwidth σ, consistency is obtained if $\lim_{n\to\infty} \sigma = 0$ and $\lim_{n\to\infty} n\sigma = \infty$ (in addition to regularity conditions on the kernel). See (Härdle et al., 2004) for a recent and easily accessible exposition (web version available).

The bias is due to smoothing the target function over the volume covered by the effective neighbors (consider eq. 3). As the intrinsic dimensionality of the data increases (the number of dimensions that they actually span locally), bias increases. Since that volume increases exponentially with dimension, the effect of the bias quickly becomes very severe. To see this, consider the classical example of the $[0, 1]^d$ hypercube in $\mathbb{R}^d$ with uniformly distributed data in the hypercube. To hold a fraction p of the data in a sub-cube of it, that sub-cube must have sides of length $p^{1/d}$. As $d \to \infty$, $p^{1/d} \to 1$, i.e. we are averaging over distances that cover almost the whole span of the data, just to keep variance constant (by keeping the effective number of neighbors constant).

When the input examples are not considered fixed the calculations of bias and variance are more complex, but similar conclusions are reached. For example, for a wide class of such kernel estimators, the unconditional variance and squared bias can be shown to be written as follows (Härdle et al., 2004):

$$\text{expected error} = \frac{C_1}{n\sigma^d} + C_2\sigma^4,$$

with C_1 and C_2 not depending on n nor d. Hence an optimal bandwidth is chosen proportional to $n^{\frac{-1}{4+d}}$, and the resulting generalization error (not counting the noise) converges in $n^{-4/(4+d)}$, which becomes very slow for large d. Consider for example the increase in number of examples required to get the same level of error, in 1 dimension versus d dimensions. If n_1 is the number of examples required to get a level of error e, to get the same level of error in d dimensions requires on the order of $n_1^{(4+d)/5}$ examples, i.e. the **required number of examples is exponential in** d. However, if the data distribution is concentrated on a lower dimensional manifold, it is the **manifold dimension** that matters. Indeed, for data on a smooth lower-dimensional manifold, the only dimension that say a k-nearest neighbor classifier sees is the dimension of the manifold, since it only uses the Euclidean distances between the near neighbors, and if they lie on such a manifold then the local Euclidean distances approach the local geodesic distances on the manifold (Tenenbaum, de Silva and Langford, 2000).

3 Summary of the results

In the following sections, we will see that:

- *In the case of a Gaussian kernel machine classifier, if there exists a line in $\mathbb{R}^d$ that intersects m times with the decision surface S (and is not included in S), one needs at least $\lceil \frac{m}{2} \rceil$ Gaussians (of same width) to learn S.*
- *At least 2^{d-1} examples are required to represent the d-bit parity (the function from $\{0,1\}^d$ to $\{-1,1\}$ which is 1 iff the sum of the bits is even), when using Gaussians with fixed σ centered on data points.*
- *Many graph-based semi-supervised learning algorithms cannot learn a number of regions with constant label higher than the number of labeled examples.*
- *When the test example x is far from all the x_i, a predictor given by eq. 1 with a local kernel either converges to a constant or a nearest neighbor classifier.* Neither of these are good in high dimension.
- *When using so-called local-derivative kernels, $\frac{\partial f(x)}{\partial x}$ is constrained to be approximately a linear combination of the vectors $(x - x_i)$ with x_i a near neighbor of x.* In high dimension (when the number of effective neighbors is significantly smaller than the dimension), this is a very strong constraint (either on the shape of the manifold or of the decision surface).
- *When there are examples with $||x - x_i||$ near σ (which is likely to happen for "good" values of σ, i.e. neither too small nor too big), with x on the decision surface, changes in x small w.r.t. σ yield only small changes in the normal vector of the decision surface.* The above statement is one about bias: within a ball of radius σ, the decision surface is constrained to be smooth (small changes in x yield small changes in the shape of the surface).
- *In the case of a Gaussian kernel classifier, when σ increases the decision surface becomes subject to specific smoothness constraints.* More generally, we present an argument supporting the conjecture that any learning algorithm with a local property, such as the *local-derivative property* ($\frac{\partial f}{\partial x}$ depends mostly on examples in a ball around x) and local smoothness of the learned function (e.g., within that ball) will be subject to the curse of dimensionality.

These statements highlight some important limitations of kernel methods, that one must keep in mind when applying such learning algorithms.

4 Minimum Number of Bases Required for Complex Functions

4.1 Limitations of Learning with Gaussians

In (Schmitt, 2002) tight bounds are established for the number of zeros of univariate radial basis function networks under certain parameter conditions. In particular, it is shown that for a fixed Gaussian width σ, a function

$$f : \mathbb{R} \to \mathbb{R}$$

of the form

$$f(x) = b + \sum_{i=1}^{k} \alpha_i \exp\left(-\frac{\|x - x_i\|^2}{\sigma^2}\right) \tag{4}$$

cannot have more than $2k$ zeros (if there is at least one non-zero α_i). Consider now the more general case of a multivariate decision function $f : \mathbb{R}^d \to \mathbb{R}$ written as in eq. 4. For any $u \in \mathbb{R}^d$ and any $w \in \mathbb{R}^d$ such that $\|w\| = 1$, the function $g : \mathbb{R} \to \mathbb{R}$ defined by

$$g(\alpha) = f(u + \alpha w)$$

can be written in the form

$$g(\alpha) = b + \sum_{i=1}^{k} \beta_i \exp\left(-\frac{|\alpha - \alpha_i|^2}{\sigma^2}\right)$$

where $u + \alpha_i w$ is the projection of x_i on the line $D_{u,w} = \{u + \alpha w, \alpha \in \mathbb{R}\}$. Thanks to the above result from (Schmitt, 2002) we can then conclude that g has at most $2k$ zeros, i.e. that $D_{u,w}$ crosses the decision surface at most $2k$ times (as long as there is a non-zero coefficient in the resulting Gaussian expansion, otherwise g may be constant and equal to 0).

Corollary 4.1. *Let S be a decision surface in $\mathbb{R}^d$ to be learned with an affine combination of Gaussians with unique width as in eq. 4. If there exists a line in $\mathbb{R}^d$ that intersects m times with S (and is not included in S), then one needs at least $\lceil \frac{m}{2} \rceil$ Gaussians to learn S.*

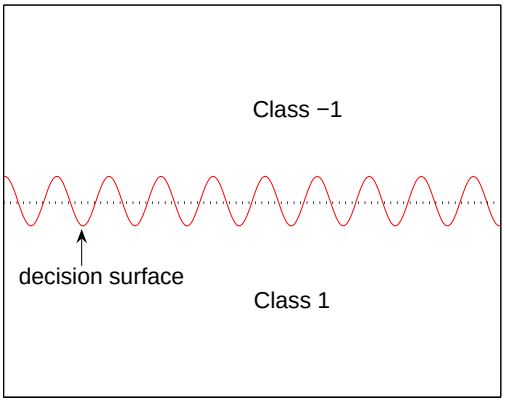

Figure 1: The dotted line crosses the decision surface 19 times: one thus needs at least 10 Gaussians to learn it with an affine combination of Gaussians with same width.

Example 4.2. Consider the decision surface shown in figure 1, which is a sinusoidal function. One may take advantage of the global regularity to learn it with few parameters (thus requiring few examples), but with an affine combination of Gaussians, corollary 4.1 implies one would need at least $\lceil \frac{m}{2} \rceil = 10$ Gaussians. For more complex tasks in higher dimension, the "complexity" of the decision surface could quickly make learning impractical when using such a local kernel method.

Remark 4.3. Of course, one only seeks to approximate the decision surface S, and does not necessarily need to learn it perfectly: corollary 4.1 says nothing about the existence of an easier-to-learn decision surface approximating S. For instance, in the example of figure 1, the dotted line could turn out to be a good enough estimated decision surface if most samples were far from the true decision surface, and this line can be obtained with only two Gaussians.

4.2 Learning the d-Bits Parity Function

The d-bits parity function is the function

$$\text{parity}: (b_1, \ldots, b_d) \in \{0,1\}^d \mapsto \begin{cases} 1 \text{ if } \sum_{i=1}^d b_i \text{ is even} \\ -1 \text{ otherwise} \end{cases}$$

We will show that learning this apparently simple function with Gaussians centered on points in $\{0,1\}^d$ is difficult, in the sense that it requires a number of Gaussians exponential in d (for a fixed Gaussian width). We will use the following notations:

$$\begin{aligned} X_d &= \{0,1\}^d = \{x_1, x_2, \ldots, x_{2^d}\} \\ H_d^0 &= \{(b_1, \ldots, b_d) \in X_d \mid b_d = 0\} \\ H_d^1 &= \{(b_1, \ldots, b_d) \in X_d \mid b_d = 1\} \\ K_\sigma(x,y) &= \exp\left(-\frac{\|x-y\|^2}{\sigma^2}\right) \end{aligned} \tag{5}$$

We say that a decision function $f : \mathbb{R}^d \to \mathbb{R}$ solves the parity problem if $\text{sign}(f(x_i)) = \text{parity}(x_i)$ for all i in $\{1, \ldots, 2^d\}$.

Lemma 4.4. *Let $f(x) = \sum_{i=1}^{2^d} \alpha_i K_\sigma(x_i, x)$ be a linear combination of Gaussians with same width σ centered on points $x_i \in X_d$. If f solves the parity problem, then $\alpha_i \text{parity}(x_i) > 0$ for all i.*

Proof. We prove this lemma by induction on d. If $d = 1$ there are only 2 points. Obviously one Gaussian is not enough to classify correctly x_1 and x_2, so both α_1 and α_2 are non-zero, and $\alpha_1 \alpha_2 < 0$ (otherwise f is of constant sign). Without loss of generality, assume $\text{parity}(x_1) = 1$ and $\text{parity}(x_2) = -1$. Then $f(x_1) > 0 > f(x_2)$, which implies $\alpha_1(1 - K_\sigma(x_1, x_2)) > \alpha_2(1 - K_\sigma(x_1, x_2))$ and $\alpha_1 > \alpha_2$ since $K_\sigma(x_1, x_2)) < 1$. Thus $\alpha_1 > 0$ and $\alpha_2 < 0$, i.e. $\alpha_i \text{parity}(x_i) > 0$ for $i \in \{1, 2\}$.

Suppose now lemma 4.4 is true for $d = d' - 1$, and consider the case $d = d'$. We denote by x_i^0 the points in H_d^0 and by α_i^0 their coefficient in the expansion of f (see eq. 5 for the definition of H_d^0). For $x_i^0 \in H_d^0$, we denote by $x_i^1 \in H_d^1$ its projection on H_d^1 (obtained by setting its last bit to 1), whose coefficient in f is α_i^1. For any $x \in H_d^0$ and $x_j^1 \in H_d^1$ we have:

$$\begin{aligned} K_\sigma(x_j^1, x) &= \exp\left(-\frac{\|x_j^1 - x\|^2}{\sigma^2}\right) \\ &= \exp\left(-\frac{1}{\sigma^2}\right) \exp\left(-\frac{\|x_j^0 - x\|^2}{\sigma^2}\right) \\ &= \gamma K_\sigma(x_j^0, x) \end{aligned}$$

where $\gamma = \exp\left(-\frac{1}{\sigma^2}\right) \in (0,1)$. Thus $f(x)$ for $x \in H_d^0$ can be written

$$\begin{aligned} f(x) &= \sum_{x_i^0 \in H_d^0} \alpha_i^0 K_\sigma(x_i^0, x) + \sum_{x_j^1 \in H_d^1} \alpha_j^1 \gamma K_\sigma(x_j^0, x) \\ &= \sum_{x_i^0 \in H_d^0} \left(\alpha_i^0 + \gamma \alpha_i^1\right) K_\sigma(x_i^0, x) \end{aligned}$$

Since H_d^0 is isomorphic to X_{d-1}, the restriction of f to H_d^0 implicitely defines a function over X_{d-1} that solves the parity problem (because the last bit in H_d^0 is 0, the parity is not modified). Using our induction hypothesis, we have that for all $x_i^0 \in H_d^0$:

$$\left(\alpha_i^0 + \gamma\alpha_i^1\right) \text{parity}(x_i^0) > 0. \tag{6}$$

A similar reasoning can be made if we switch the roles of H_d^0 and H_d^1. One has to be careful that the parity is modified between H_d^1 and its mapping to X_{d-1} (because the last bit in H_d^1 is 1). Thus we obtain that the restriction of $(-f)$ to H_d^1 defines a function over X_{d-1} that solves the parity problem, and the induction hypothesis tells us that for all $x_j^1 \in H_d^1$:

$$\left(-\left(\alpha_j^1 + \gamma\alpha_j^0\right)\right)\left(-\text{parity}(x_j^1)\right) > 0. \tag{7}$$

and the two negative signs cancel out. Now consider any $x_i^0 \in H_d^0$ and its projection $x_i^1 \in H_d^1$. Without loss of generality, assume $\text{parity}(x_i^0) = 1$ (and thus $\text{parity}(x_i^1) = -1$). Using eq. 6 and 7 we obtain:

$$\begin{aligned} \alpha_i^0 + \gamma\alpha_i^1 &> 0 \\ \alpha_i^1 + \gamma\alpha_i^0 &< 0 \end{aligned}$$

It is obvious that for these two equations to be simultaneously verified, we need α_i^0 and α_i^1 to be non-zero and of opposite sign. Moreover, $\alpha_i^0 + \gamma\alpha_i^1 > 0 > \alpha_i^1 + \gamma\alpha_i^0 \Rightarrow \alpha_i^0 > \alpha_i^1$, which implies $\alpha_i^0 > 0$ and $\alpha_i^1 < 0$, i.e. $\alpha_i^0 \text{parity}(x_i^0) > 0$ and $\alpha_i^1 \text{parity}(x_i^1) > 0$. Since this is true for all x_i^0 in H_d^0, we have proved lemma 4.4. □

Theorem 4.5. *Let $f(x) = b + \sum_{i=1}^{2^d} \alpha_i K_\sigma(x_i, x)$ be an affine combination of Gaussians with same width σ centered on points $x_i \in X_d$. If f solves the parity problem, then there are at least 2^{d-1} non-zero coefficients α_i.*

Proof. We begin with two preliminary results. First, given any $x_i \in X_d$, the number of points in X_d that differ from x_i by exactly k bits is $\binom{d}{k}$. Thus,

$$\sum_{x_j \in X_d} K_\sigma(x_i, x_j) = \sum_{k=0}^{d} \binom{d}{k} \exp\left(-\frac{k^2}{\sigma^2}\right) = c_\sigma. \tag{8}$$

Second, it is possible to find a linear combination (i.e. without bias) of Gaussians g such that $g(x_i) = f(x_i)$ for all $x_i \in X_d$. Indeed, let

$$g(x) = f(x) - b + \sum_{x_j \in X_d} \beta_j K_\sigma(x_j, x). \tag{9}$$

g verifies $g(x_i) = f(x_i)$ iff $\sum_{x_j \in X_d} \beta_j K_\sigma(x_j, x_i) = b$, i.e. the vector β satisfies the linear system $M_\sigma \beta = b\mathbf{1}$, where M_σ is the kernel matrix whose element (i, j) is $K_\sigma(x_i, x_j)$ and $\mathbf{1}$ is a vector of ones. It is well known that M_σ is invertible as long as the x_i are all different, which is the case here (Micchelli, 1986). Thus $\beta = bM_\sigma^{-1}\mathbf{1}$ is the only solution to the system.

We now proceed to the proof of the theorem. By contradiction, suppose $f(x) = b + \sum_{i=1}^{2^d} \alpha_i K_\sigma(x_i, x)$ solves the parity problem with less than 2^{d-1} non-zero coefficients α_i. Then there exist two points x_s and x_t in X_d such that $\alpha_s = \alpha_t = 0$ and $\text{parity}(x_s) = 1 = -\text{parity}(x_t)$. Consider the function g defined as in eq. 9 with $\beta = bM_\sigma^{-1}\mathbf{1}$. Since $g(x_i) = f(x_i)$ for all $x_i \in X_d$, g solves the parity problem with a linear combination of Gaussians centered points in X_d. Thus, applying lemma 4.4, we have in particular that $\beta_s \text{parity}(x_s) > 0$ and $\beta_t \text{parity}(x_t) > 0$ (because $\alpha_s = \alpha_t = 0$), so that $\beta_s \beta_t < 0$. But, because of eq. 8, $M_\sigma \mathbf{1} = c_\sigma \mathbf{1}$, which means $\mathbf{1}$ is an eigenvector of M_σ with eigenvalue $c_\sigma > 0$. Consequently, $\mathbf{1}$ is also an eigenvector of M_σ^{-1} with eigenvalue $c_\sigma^{-1} > 0$, and $\beta = bM_\sigma^{-1}\mathbf{1} = bc_\sigma^{-1}\mathbf{1}$, which is in contradiction with $\beta_s \beta_t < 0$: f must have at least 2^{d-1} non-zero coefficients. □

Remark 4.6. The bound in theorem 4.5 is tight, since it is possible to solve the parity problem with exactly 2^{d-1} Gaussians and a bias, for instance by using a negative bias and putting a positive weight on each example satisfying $\text{parity}(x) = 1$.

Remark 4.7. When trained to learn the parity function, a SVM may learn a function that looks like the opposite of the parity on test points (while still performing optimally on training points). For instance, a SVM trained with 4000 unique points from X_{15} achieves a 90% error rate on 20000 different test samples (with $\sigma = 0.5$). This is because a bit less than 90% of these test samples are at distance 1 from their nearest neighbor in the training set. With this value of σ, the SVM output is similar to nearest neighbor classification, which is wrong when the nearest neighbor is at distance 1. However, this observation cannot in general be used to build a model for parity with Gaussians (by taking the opposite of the SVM output): indeed, this approach would work here because the training samples are dense enough in X_{15}, but this will not be the case with fewer training data or in higher dimensions. Note that in all our experiments performed with parity datasets, a SVM had a 50% or more error rate on new points, which illustrates its inability to generalize for this problem.

Remark 4.8. If the centers of the Gaussians are not restricted anymore to be points in X_d, it is possible to solve the parity problem with only $d+1$ Gaussians and no bias. Indeed, consider f defined by

$$f(x) = \sum_{i=0}^{d} (-1)^i K_\sigma(y_i, x)$$

with

$$y_i = \frac{i}{d}\mathbf{1}.$$

For σ small enough, $\text{sign}(f(y_i)) = (-1)^i$. In addition, for any $x \in X_d$, $f(x) = \gamma f(\hat{x})$, where $\hat{x}$ is the projection of x on the diagonal (the line spanned by $\mathbf{1}$) and $\gamma = \exp\left(-\frac{\|x-\hat{x}\|^2}{\sigma^2}\right) \in (0,1)$. Let $x = (b_1, \ldots, b_d)$: its projection $\hat{x}$ is given by

$$\hat{x} = (x \cdot \mathbf{1})\frac{\mathbf{1}}{\|\mathbf{1}\|^2} = \frac{1}{d}\left(\sum_i b_i\right)\mathbf{1} = y_{(\sum_i b_i)}$$

and therefore $\text{sign}(f(x)) = \text{sign}(f(\hat{x})) = \text{sign}\left(f\left(y_{(\sum_i b_i)}\right)\right) = (-1)^{(\sum_i b_i)} = \text{parity}(x)$: f solves the parity problem with a linear combination of $d+1$ Gaussians.

5 When a Test Example is Far from Training Examples

The argument presented in this section is mathematically trivial but nonetheless very powerful, especially in high-dimensional spaces. We consider here functions f as in eq. 1 where K is a **local kernel**, i.e. is such that for x a test point and x_i a training point

$$\lim_{||x-x_i|| \to \infty} K(x, x_i) \to c_i \tag{10}$$

where c_i is a constant that does not depend on x. For instance, this is true for the Gaussian kernel (eq. 2) with $c_i = 0$. This is also true for the data-dependent kernel obtained by the centering step in kernel PCA (Schölkopf, Smola and Müller, 1998) when the original kernel is the Gaussian kernel, with

$$c_i = -\frac{1}{n}\sum_k K_\sigma(x_i, x_k) + \frac{1}{n^2}\sum_{k,l} K_\sigma(x_k, x_l).$$

Clearly, from inspection of equations 1 and 10, when x goes farther from the training set, i.e. when

$$d(D, x) = \min_i ||x - x_i||$$

goes to ∞, then $f(x) \to b + \sum_i \alpha_i c_i$. In the case of regression the prediction converges to a constant. In the case of manifold learning with $f(x)$ representing an embedding function, it means all points far from the training set are mapped to the same low-dimensional coordinates. Note that this convergence to a constant does not hold for non-local kernels such as those of Isomap or LLE (Bengio et al., 2004).

In the case of classification (e.g. SVMs), there are two distinct cases. If $b + \sum_i \alpha_i c_i \neq 0$, $\text{sign}(f(x))$ converges to a constant classifier. Otherwise, the result is less obvious and will depend on the kernel used. For instance, with a Gaussian kernel, it is easy to see that the classification will only depend on the nearest neighbor as long as x is not too close to a nearest neighbor classifier decision surface (which is more and more likely as x goes farther from the training set). In all cases we clearly get either a poor high-bias prediction (a constant), or a highly local one (the nearest neighbor rule, which is also likely to have a high bias when x is far from its nearest neighbor) that suffers from the curse of dimensionality. Note that when x is a high-dimensional vector, the nearest neighbor is not much closer than the other examples (the ratio of the distance between the nearest and the farthest converges to 1 (Beyer et al., 1999)), hence it is not very informative. A random test point is therefore not unlikely to be relatively far from its nearest neighbor in the training set, compared to σ, when σ is chosen to be smaller than the scale of the training data, so this situation is not a rare one for high-dimensional data.

6 Locality of the Estimator and its Tangent

In this section we consider how the derivative of $f(x)$ w.r.t x (i.e. its "shape") is influenced by the positions of training examples x_i. We say a kernel is **local-derivative** if its derivative can be written

$$\frac{\partial K(x, x_i)}{\partial x} = \sum_j \beta_{ij}(x - x_j)K'(x, x_j) \tag{11}$$

where K' is either a local kernel verifying eq. 10 with $c_i = 0$ or any kernel that is 0 when x_j is not a k nearest neighbor of x in the training set. From equations 1 and 11, we will see that $\partial f / \partial x$ is contained (possibly approximately) in the span of the vectors $(x - x_j)$ with x_j a neighbor of x (the notion of neighborhood being defined by k or by how fast K' converges to 0 when $\|x - x_j\|$ increases).

Examples of local-derivative kernels include the Gaussian kernel as well as the kernels for Isomap or LLE. In the case of the Gaussian kernel, we have

$$\frac{\partial K_\sigma(x, x_i)}{\partial x} = -\frac{2\alpha_i}{\sigma^2}(x - x_i)K_\sigma(x, x_i). \tag{12}$$

As shown in (Bengio and Monperrus, 2005), in the case of Isomap we obtain a linear combination of vectors $(x - x_j)$ with x_j one of the k nearest neighbors of x, where k is an hyper-parameter for the Isomap kernel. Thus it verifies eq. 11 with $K'(x, x_j) = 0$ when x_j is not a k nearest neighbor of x. The same property holds for the LLE kernel, where $K_{LLE}(x, x_i)$ is the weight of x_i in the reconstruction of x by its k nearest neighbors (Bengio et al., 2004). Indeed, this weight is obtained by the following equation (Saul and Roweis, 2002):

$$K_{LLE}(x, x_i) = \frac{\sum_{j=1}^k G_{ij}^{-1}}{\sum_{l,m=1}^k G_{lm}^{-1}} \tag{13}$$

with G^{-1} the inverse of the local Gram matrix G

$$G_{lm} = (x - x_l) \cdot (x - x_m)$$

for all pairs (x_l, x_m) of k nearest neighbors of x in the training set. Because $G^{-1} = |G|^{-1}C^T$ with C the cofactor matrix of G, eq. 13 can be rewritten as

$$K_{LLE}(x, x_i) = \frac{\sum_j s_j \prod_{l,m}(G_{lm})^{t_{jlm}}}{\sum_j u_j \prod_{l,m}(G_{lm})^{v_{jlm}}}$$

and consequently, thanks to the usual derivation rules, its derivative is a linear combination of derivatives of terms of the form $(G_{lm})^t$. But

$$\begin{aligned} \frac{\partial (G_{lm})^t}{\partial x} &= \frac{\partial ((x - x_l) \cdot (x - x_m))^t}{\partial x} \\ &= t(G_{lm})^{t-1}(x - x_l + x - x_m) \end{aligned}$$

which implies that the derivative of $K_{LLE}(x, x_i)$ w.r.t x is in the span of the vectors $(x - x_j)$ with x_j a k nearest neighbor of x, as for Isomap: eq. 11 is verified.

From equations 1 and 11, we obtain that

$$\frac{\partial f(x)}{\partial x} = \sum_i \gamma_i K'(x, x_i)(x - x_i). \tag{14}$$

This equation helps us to understand how changes in x yield changes in $f(x)$ through the different training examples. In particular, because of the properties of K', **only the terms involving near neighbors of x have a significant influence on the result.** These neighbors are either defined by the hyper-parameter k or by how fast K' converges to 0 (i.e. σ for a Gaussian kernel, see section 6.2 for an example). Note that in the second case, we will also need the γ_i to be bounded in order to ensure such a locality property.

To better understand this, we will consider specifically the geometry of manifold learning and the geometry of decision surfaces.

6.1 Geometry of Tangent Planes

For manifold learning, with $f_k(x)$ the k-th embedding coordinate of x, these derivatives together (for $k = 1, 2, \ldots$) span the tangent plane of the manifold at x. Indeed, $\frac{\partial f_k(x)}{\partial x}$ is the direction of variation of x which corresponds to the largest increase in the k-th embedding coordinate. Inspection of eq. 14 shows that $\frac{\partial f_k(x)}{\partial x}$ is a linear combination of the vectors $(x - x_i)$, with x_i a near neighbor of x in the training set, since the other training examples contribute very little to the sum.

Consequently, the **tangent plane of the manifold is approximately in the span of the vectors $(x - x_i)$ with x_i a near neighbor of** x. If the number of data dimensions is large compared to the number of near neighbors, then this is a very strong constraint, irrespective of the learning method. Consider for example a distribution whose samples have an intrinsic dimension of about 100, but where the algorithm uses 5 significant neighbors. Constraining the tangent plane to be in the span of these 5 near neighbors is a very strong constraint in the local $\mathbb{R}^{100}$ subspace, which will result in a **high-variance estimator** for the manifold.

6.2 Geometry of Decision Surfaces

A similar argument can be made for the decision surfaces of kernel classifiers. Consider a point x on the decision surface, i.e. $f(x) = 0$. The normal of the decision surface is the vector $\frac{\partial f(x)}{\partial x}$ evaluated at x. From eq. 14 we see that **the decision surface normal vector is a linear combination of the vectors** $(x - x_i)$**, with** x_i **a near neighbor of** x. Again, if the intrinsic dimension is large compared to the number of near neighbors, then this is a very strong constraint, irrespective of the learning method. Like in the unsupervised case, this is likely to yield a **high-variance estimator** since only a few points determine a crucial property of the decision surface, i.e. its shape.

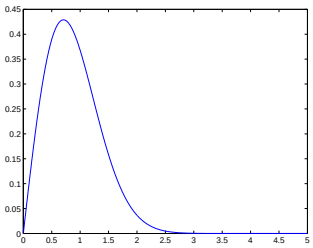

Figure 2: Plot of ze^{-z^2}.

Note that in the above statements, depending on the kernel K', we may need to require the γ_i to be bounded. Take for instance the classical 1-norm soft margin SVM. The decision surface normal vector can be obtained from eq. 1 and 12:

$$\frac{\partial f(x)}{\partial x} = -\sum_i \frac{2\alpha_i}{\sigma} K_\sigma(x, x_i) \frac{\|x_i - x\|}{\sigma} \frac{(x_i - x)}{\|x_i - x\|} \tag{15}$$

i.e. as a linear combination of vectors of norm 1 whose weights depend notably on ze^{-z^2} with $z = \|x_i - x\|/\sigma$. This function is plotted in figure 2, showing that if there are training examples x_i such that $\|x_i - x\|/\sigma$ is less than 2, they will be the main contributors in eq. 15, as long as the coefficients α_i are bounded (which is the case for the 1-norm SVM). Consider a change in x small with respect to σ: this change does not affect much the direction nor the weight of these main contributors, which means the normal vector is almost unchanged. The bottom line is that **the decision surface is constrained to be smooth**, in the sense that the normal vector is almost constant in a ball whose radius is small with respect to σ. This very qualitative statement helps us understand why one usually needs a small σ to learn complex decision functions while smooth functions can be learned with a bigger σ.

In this specific example of the 1-norm soft margin SVM, it is important to note that we do not want training examples to be all close to x or all far from x (w.r.t. σ), i.e. σ should not take extreme values. Indeed, it is shown in (Keerthi and Lin, 2003) that when $\sigma \to 0$ or $\sigma \to \infty$ and the regularization parameter C is fixed, the SVM will assign the entire data space to the majority class (and when $\sigma \to \infty$ and $C \propto \sigma^2$ it converges to a linear SVM).

7 The Curse of Dimensionality for Local Non-Parametric Semi-Supervised Learning

In this section we focus on algorithms of the type described in recent papers (Zhu, Ghahramani and Lafferty, 2003; Zhou et al., 2004; Belkin, Matveeva and Niyogi, 2004; Delalleau, Bengio and Le Roux, 2005), which are graph-based non-parametric semi-supervised learning algorithms. Because the analysis is centered on the decision surface

and eq. 1, it also applies to transductive SVMs and Gaussian processes when the kernel is either Gaussian or normalized Gaussian.

The graph-based algorithms we consider here can be seen as minimizing the following cost-function, as shown in (Delalleau, Bengio and Le Roux, 2005):

$$C(\hat{Y}) = \|\hat{Y}_l - Y_l\|^2 + \mu \hat{Y}^\top L \hat{Y} + \mu\epsilon \|\hat{Y}\|^2 \tag{16}$$

with $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_n)$ the estimated labels on both labeled and unlabeled data, and L the (un-normalized) graph Laplacian derived from a similarity function W between points such that $W_{ij} = W(x_i, x_j)$ corresponds to the weights of the edges in the graph. Here, $\hat{Y}_l = (\hat{y}_1, \ldots, \hat{y}_l)$ is the vector of estimated labels on the l labeled examples, whose known labels are given by $Y_l = (y_1, \ldots, y_l)$, and one may constrain $\hat{Y}_l = Y_l$ as in (Zhu, Ghahramani and Lafferty, 2003) by letting $\mu \to 0$.

Minimization of the cost criterion of eq. 16 can also be seen as a *label propagation* algorithm, i.e. labels are "spread" around labeled examples, with "around" being given by the structure of the graph. An intuitive view of label propagation suggests that a region of the manifold near a labeled (e.g. positive) example will be entirely labeled positively, as the example spreads its influence by propagation on the graph representing the underlying manifold. Thus, the number of regions with constant label should be on the same order as (or less than) the number of labeled examples. This is easy to see in the case of a sparse weight matrix W. We define a region with constant label as a connected subset of the graph where all nodes x_i have the same estimated label (sign of $\hat{y}_i$), and such that no other node can be added while keeping these properties. The following proposition then holds (note that it is also true, but trivial, when W defines a fully connected graph).

Proposition 7.1. *After running a label propagation algorithm minimizing the cost of eq. 16, the number of regions with constant estimated label is less than (or equal to) the number of labeled examples.*

Proof. By contradiction, if this proposition is false, then there exists a region with constant estimated label that does not contain any labeled example. Without loss of generality, consider the case of a positive constant label, with $x_{l+1}, \ldots, x_{l+q}$ the q samples in this region. The part of the cost of eq. 16 depending on their labels is

$$\begin{aligned} C(\hat{y}_{l+1}, \ldots, \hat{y}_{l+q}) = & \frac{\mu}{2} \sum_{i,j=l+1}^{l+q} W_{ij} (\hat{y}_i - \hat{y}_j)^2 \\ & + \mu \sum_{i=l+1}^{l+q} \left(\sum_{j \notin \{l+1,\ldots,l+q\}} W_{ij} (\hat{y}_i - \hat{y}_j)^2 \right) \\ & + \mu\epsilon \sum_{i=l+1}^{l+q} \hat{y}_i^2 . \end{aligned}$$

The second term is stricly positive, and because the region we consider is maximal (by definition) all samples x_j outside of the region such that $W_{ij} > 0$ verify $\hat{y}_j < 0$ (for x_i a sample in the region). Since all $\hat{y}_i$ are stricly positive for $i \in \{l+1, \ldots, l+q\}$, this means this second term can be stricly decreased by setting all $\hat{y}_i$ to 0 for $i \in \{l+1, \ldots, l+q\}$. This also sets the first and third terms to zero (i.e. their minimum), showing that the set of labels $\hat{y}_i$ are not optimal, which is in contradiction with their definition as the labels that minimize C. □

This means that if the class distributions are such that there are many distinct regions with constant labels (either separated by low-density regions or regions with samples from the other class), we will need at least the same number of labeled samples as there are such regions (assuming we are using a sparse local kernel such as the

k-nearest neighbor kernel, or a thresholded Gaussian kernel). But this number could *grow exponentially with the dimension of the manifold(s) on which the data lie*, for instance in the case of a labeling function varying highly along each dimension, *even if the label variations are "simple" in a non-local sense*, e.g. if they alternate in a regular fashion.

When the affinity matrix W is not sparse (e.g. Gaussian kernel), obtaining such a result is less obvious. However, there often exists a sparse approximation of W (for instance, in the case of a Gaussian kernel, one can set to 0 entries below a given threshold or that do not correspond to a k-nearest neighbor relationship). Thus we conjecture the same kind of result holds for dense weight matrices, if the weighting function is local in the sense that it is close to zero when applied to a pair of examples far from each other.

8 General Curse of Dimensionality Argument

Can we obtain more general results that would apply to a broader class of learning algorithms, such as transductive SVMs and Gaussian processes with Gaussian kernel? For this we must first establish the notion that if we choose the ball $\mathcal{N}(x)$ too large the predicted function is constrained to be too smooth. The following proposition supports this statement.

Proposition 8.1. *For the Gaussian kernel classifier, as σ increases and becomes large compared with the diameter of the data, in the smallest sphere containing the data the decision surface becomes linear if $\sum_i \alpha_i = 0$ (e.g. for SVMs), or else the normal vector of the decision surface becomes a linear combination of two sphere surface normal vectors, with each sphere centered on a weighted average of the examples of the corresponding class.*

Proof. The decision surface is the set of x such that

$$f(x) = \sum_i \alpha_i e^{-\frac{1}{2}||x-x_i||^2/\sigma^2} = 0.$$

The normal vector of the decision surface of the Gaussian classifier is therefore

$$\frac{\partial f(x)}{\partial x} = -\sum_i \alpha_i \frac{(x-x_i)}{\sigma^2} e^{-\frac{1}{2}||x-x_i||^2/\sigma^2}.$$

In the smallest sphere containing the data, $||x - x_i|| \leq \Delta$, Δ being the diameter of that sphere. As σ becomes large with respect to Δ, it becomes large with respect to all the distances $||x - x_i||$, and the factors $e^{-\frac{1}{2}||x-x_i||^2/\sigma^2}$ approach 1. For example for $\sigma = 4\Delta$, these factors are in the interval $[0.969, 1]$. Let us write $\beta_i(x) = \frac{\alpha_i}{\sigma^2} e^{-\frac{1}{2}||x-x_i||^2/\sigma^2}$, which approaches $\frac{\alpha_i}{\sigma^2}$. The normal vector can thus be decomposed in three terms

$$\frac{\partial f(x)}{\partial x} = -x \sum_i \beta_i(x) + \sum_{i:\alpha_i>0} \beta_i(x) x_i - \sum_{i:\alpha_i<0} (-\beta_i(x)) x_i. \tag{17}$$

Let us introduce two weighted averages, corresponding to each of the two classes. Define for $k = 1$ and $k = -1$

$$\begin{aligned} S_k(x) &= k \sum_{i:sign(\alpha_i)=k} \beta_i(x) \\ m_k(x) &= k \sum_{i:sign(\alpha_i)=k} \beta_i(x) x_i \\ \mu_k(x) &= \frac{m_k(x)}{S_k(x)} \end{aligned}$$

These are centers of mass for each of the two classes, with weights $|\beta_i(x)|$. The normal vector is thereby written as

$$\frac{\partial f(x)}{\partial x} = -xS_1(x) + xS_{-1}(x) + m_1(x) - m_{-1}(x)$$

If $\sum_i \alpha_i = 0$ (as in many SVMs), then $S_1(x) - S_{-1}(x)$ approaches 0, so that the first two terms above vanish and the dominant remainder is the difference vector $m_1 - m_{-1}$, which defines a linear decision surface whose normal vector is aligned with the difference of two weighted means (one per class).

Without the condition $\sum_i \alpha_i = 0$ we obtain a very specific quadratic decision surface. More precisely, the normal vector can be rewritten

$$\frac{\partial f(x)}{\partial x} = S_1(x)(\mu_1(x) - x) - S_{-1}(x)(\mu_{-1}(x) - x).$$

We can now interpret the above as follows. As σ increases, $\mu_k(x)$ approaches a fixed center and $S_k(x)$ a fixed scalar, and the decision surface normal vector is a linear combination of two vectors: the normal vector of the surface of the sphere centered at μ_1 and the normal vector of the surface of the sphere centered at μ_{-1}. □

We are now ready to put all these arguments together to argue that the curse of dimensionality plagues a large class of semi-supervised and supervised learning algorithms.

Conjecture 8.2. *Consider a smoothness property S of discriminant functions f. Let P be a data-generating process from which training sets are sampled from a bounded region Ω. Suppose that P is such that, with N balls covering Ω with property S in each ball, one cannot build a classifier with generalization error level less than ϵ. For kernel machines with*

1. *the local-derivative property definining balls $\mathcal{N}(x)$ around a given test point x,*
2. *a smooth decision surface within any ball $\mathcal{N}(x)$ in the sense that smoothness property S is satisfied within $\mathcal{N}(x)$ (e.g. consider S shown for the Gaussian kernel in Proposition 8.1),*

the number of training examples required for achieving error level ϵ grows linearly with N. N may grow exponentially with the dimension of a manifold near which the density concentrates, on the decision surface.

Supporting argument. The notion of locality provided by the *local-derivative* property allows us to define a ball $\mathcal{N}(x)$ around each test point x, containing neighbors that have a dominating influence on $\frac{\partial f(x)}{\partial x}$. The smoothness property S defined in Proposition 8.1 constrains the decision surface to be either linear (case of SVMs) or a particular quadratic form (the decision surface normal vector is a linear combination of two vectors defined by the center of mass of examples of each class). If P is as defined above, one must have at least N balls covering Ω, and let k be the smallest number such that one needs at least k examples in each ball to reach error level ϵ. The number of examples thus required is kN. To see that N can be exponential in some dimension, consider the maximum radius r of all these balls and the radius R of Ω. If the region of the decision surface in which the density is non-negligible has intrinsic dimension d, then N could be as large as the number of radius r balls that can tile a d-dimensional manifold of radius R, which is at least $\left(\frac{R}{r}\right)^d$. This completes the argument. □

9 Conclusion

The central claim of this paper is that there are fundamental problems with non-parametric local approaches to learning, due to the curse of dimensionality. Even though learning algorithms such as SVMs behave better in this respect than classical algorithms such as k-nearest-neighbor classifiers (because they can do something between the

k-nearest-neighbor and linear classifier), they also suffer from the curse of dimensionality, at least when the kernels are local.

We have presented a series of theoretical results, some obvious and others less, that illustrate this phenomenon. They highlight some intrinsic limitations of those local learning algorithms, that can make them fail when applied on high-dimensional problems where one has to look beyond what happens locally in order to overcome the curse of dimensionality. However, there is still work to do in order to obtain more general theoretical results that would show precisely how the generalisation error degrades when the (effective) dimension of the data grows, as can be done for classical non-parametric algorithms.

Acknowledgments

The authors would like to thank the following funding organizations for support: NSERC, MITACS, and the Canada Research Chairs. The authors are also grateful for the feedback and stimulating exchanges that helped to shape this paper, with Yann Le Cun and Léon Bottou.

References

Baker, C. (1977). *The numerical treatment of integral equations*. Clarendon Press, Oxford. 2

Belkin, M., Matveeva, I., and Niyogi, P. (2004). Regularization and semi-supervised learning on large graphs. In Shawe-Taylor, J. and Singer, Y., editors, *COLT'2004*. Springer. 2, 12

Belkin, M. and Niyogi, P. (2003). Using manifold structure for partially labeled classification. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press. 2

Bellman, R. (1961). *Adaptive Control Processes: A Guided Tour*. Princeton University Press, New Jersey. 3

Bengio, Y., Delalleau, O., Le Roux, N., Paiement, J.-F., Vincent, P., and Ouimet, M. (2004). Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16(10):2197–2219. 2, 10

Bengio, Y. and Monperrus, M. (2005). Non-local manifold tangent learning. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*. MIT Press. 10

Beyer, K. S., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful? In *Proceeding of the 7th International Conference on Database Theory*, pages 217–235. Springer-Verlag. 10

Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh. 2

Brand, M. (2003). Charting a manifold. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press. 2

Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20:273–297. 2

Delalleau, O., Bengio, Y., and Le Roux, N. (2005). Efficient non-parametric function induction in semi-supervised learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*. 2, 12, 13

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58. 3

Härdle, W., Müller, M., Sperlich, S., and Werwatz, A. (2004). *Nonparametric and Semiparametric Models*. Springer, http://www.xplore-stat.de/ebooks/ebooks.html. 4

Keerthi, S. S. and Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667–1689. 12

Micchelli, C. A. (1986). Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22. 8

Nyström, E. (1928). Über die praktische auflösung von linearen integralgleichungen mit anwendungen auf randwertaufgaben der potentialtheorie. *Commentationes Physico-Mathematicae*, 4(15):1–52. 2

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326. 2

Saul, L. and Roweis, S. (2002). Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155. 10

Schmitt, M. (2002). Descartes' rule of signs for radial basis function neural networks. *Neural Computation*, 14(12):2997–3011. 5, 6

Schölkopf, B., Burges, C. J. C., and Smola, A. J. (1999). *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA. 2

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319. 2, 9

Tenenbaum, J., de Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323. 2, 4

Weiss, Y. (1999). Segmentation using eigenvectors: a unifying view. In *Proceedings IEEE International Conference on Computer Vision*, pages 975–982. 2

Williams, C. and Seeger, M. (2000). The effect of the input density distribution on kernel-based classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann. 2

Zhou, D., Bousquet, O., Navin Lal, T., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA. MIT Press. 2, 12

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML'2003*. 2, 12, 13