# Non-Local Manifold Parzen Windows

Yoshua Bengio and Hugo Larochelle
Dept. IRO, Université de Montréal
P.O. Box 6128, Downtown Branch, Montreal, H3C 3J7, Qc, Canada
{bengioy,larocheh}@iro.umontreal.ca
**Technical Report 1264**, February 2005, Revised June 2005
Département d'Informatique et Recherche Opérationnelle

July 5, 2005

**Abstract**

In order to escape from the curse of dimensionality, we claim that one can learn non-local functions, in the sense that the value and shape of the learned function at $x$ must be inferred using examples that may be far from $x$. With this objective, we present a non-local non-parametric density estimator. It builds upon previously proposed Gaussian mixture models with regularized covariance matrices to take into account the local shape of the manifold. It also builds upon recent work on non-local estimators of the tangent plane of a manifold, which are able to generalize in places with little training data, unlike traditional, local, non-parametric models.

## 1 Introduction

A central objective of statistical machine learning is to discover structure in the joint distribution between random variables, so as to be able to make predictions about new combinations of values of these variables. A central issue in obtaining generalization is how information from the training examples can be used to make predictions about new examples and, without strong prior assumptions (i.e. in non-parametric models), this may be fundamentally difficult, as illustrated by the curse of dimensionality.

(Bengio, Delalleau and Le Roux, 2005) and (Bengio and Monperrus, 2005) present several arguments illustrating some fundamental limitations of modern kernel methods due to the curse of dimensionality, when the kernel is local (like the Gaussian kernel). These arguments are all based on the locality of the estimators, i.e., that very important information about the predicted function at $x$ is derived mostly from the near neighbors of $x$ in the training set. This analysis has been applied to supervised learning algorithms such as SVMs as well as to unsupervised manifold learning algorithms such as LLE (Roweis and Saul, 2000), Isomap (Tenenbaum, de Silva and Langford, 2000), and kernel Principal Component Analysis (kPCA) (Schölkopf, Smola and Müller, 1998). The analysis in (Bengio, Delalleau and Le Roux, 2005) highlights intrinsic limitations of such 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.

This strongly suggests to investigate **non-local learning methods**, which can in principle generalize at $x$ using information gathered at training points $x_i$ that are far from $x$. We present here such a non-local learning algorithm, in the realm of density estimation.

The proposed non-local non-parametric density estimator builds upon the Manifold Parzen density estimator (Vincent and Bengio, 2003) that associates a regularized Gaussian with each training point, and upon recent work on non-local estimators of the tangent plane of a manifold (Bengio and Monperrus, 2005). The *local* covariance matrix characterizing the density in the immediate neighborhood of a data point is learned as a **function** of that data point, with *global* parameters. This allows to potentially generalize in places with little or no training data, unlike traditional, local, non-parametric models. The implicit assumption in such a model is that there is some kind of regularity in the shape of the density, such that learning about its shape in one region could be informative of the shape in another region that is not adjacent. Note that the smoothness assumption typically underlying non-parametric models relies on a simple form of such transfer, but only for neighboring regions, which is not very helpful when the intrinsic dimension of the data (the dimension of the manifold on which or near which it lives) is high or when the underlying density function has many variations (Bengio, Delalleau and Le Roux, 2005). The proposed model is also related to the Neighborhood Component Analysis algorithm (Goldberger et al., 2005), which learns a global covariance matrix for use in the Mahalanobis distance within a non-parametric classifier. Here we generalize this global matrix to one that is a function of the datum $x$.

## 2 Manifold Parzen Windows

In the Parzen Windows estimator, one puts a spherical (*isotropic*) Gaussian around each training point $x_i$, with a single shared variance hyper-parameter. How can we improve upon this estimator in the light of the *manifold hypothesis* (according to which high dimensional data is likely concentrated around a lower dimensional manifold), and make it a bit less local? One approach, introduced in (Vincent and Bengio, 2003), is to use not just the presence of $x_i$ and its neighbors but also their geometry, trying to infer the principal characteristics of the local shape of the manifold (where the density concentrates), which can be summarized in the covariance matrix of the Gaussian, as illustrated in Figure 1. If the data concentrates in certain directions around $x_i$, we want that covariance matrix to be "flat" (near zero variance) in the orthogonal directions.

One way to achieve this is to parametrize each of these covariance matrices in terms of "principal directions" (which correspond to the tangent vectors of the manifold, if the data concentrates on a manifold). In this way we do not need to specify individually all the entries of the covariance matrix. The only required assumption is that the "noise directions" orthogonal to the "principal directions" all have the same variance.

$$\hat{p}(y) = \frac{1}{n}\sum_{i=1}^{n} N(y; x_i + \mu(x_i), S(x_i)) \tag{1}$$

where $N(y; x_i + \mu(x_i), S(x_i))$ is the Gaussian density of $y$, with mean vector $x_i + \mu(x_i)$ and covariance matrix $S(x_i)$ represented compactly by

$$S(x_i) = \sigma^2_{noise}(x_i) I + \sum_{j=1}^{d} s_j(x_i) v_j(x_i) v_j(x_i)' \tag{2}$$

where $s_j(x_i)$ and $\sigma^2_{noise}(x_i)$ are scalars, and $v_j(x_i)$ denotes a "principal" direction with variance $s_j(x_i)+\sigma^2_{noise}(x_i)$, while $\sigma^2_{noise}(x_i)$ is the noise variance (the variance in all the other directions). $v_j(x_i)'$ denotes the transpose of $v_j(x_i)$.

In (Vincent and Bengio, 2003), $\mu(x_i) = 0$, and $\sigma^2_{noise}(x_i) = \sigma_0^2$ is a global hyper-parameter, while $(\lambda_j(x_i), v_j) = (s_j(x_i) + \sigma^2_{noise}(x_i), v_j(x_i))$ are the leading (eigenvalue,eigenvector) pairs from the eigen-decomposition of a locally weighted covariance matrix (e.g. the empirical covariance of the vectors $x_l - x_i$, with $x_l$ a near neighbor of $x_i$). The "noise level" hyper-parameter $\sigma_0^2$ must be chosen such that the principal eigenvalues are all greater than $\sigma_0^2$. Another hyper-parameter is the number $d$ of principal components to keep. Alternatively, one can choose $\sigma^2_{noise}(x_i)$ to be the $d+1$-th eigenvalue, which guarantees that $\lambda_j(x_i) > \sigma^2_{noise}(x_i)$, and gets rid of a hyper-parameter. This very simple model was found to be consistently better than the ordinary Parzen density estimator in numerical experiments in which all hyper-parameters are chosen by cross-validation.

Like mixtures of factor analysis models (Roweis, Saul and Hinton, 2002) and other low-rank Gaussian mixtures (Brand, 2003), this model works well when the data locally span a smooth low-dimensional manifold. In (Teh and Roweis, 2003), an algorithm is proposed to form a globally coherent coordinate system from all these local patches (assuming a connected overlap graph). However, such models are still very local, since only the examples in the neighborhood of $x_i$ contribute in inferring the shape of the manifold around $x_i$.

# 3 Non-Local Manifold Tangent Learning

In (Bengio and Monperrus, 2005) a manifold learning algorithm was introduced in which the tangent plane of a $d$-dimensional manifold at $x$ is learned as a function of $x \in \mathbb{R}^D$, using globally estimated parameters. The output of the predictor function $F(x)$ is a $d \times D$ matrix whose rows are (possibly non-orthogonal) vectors that span the tangent plane. The training information about the tangent plane is obtained by considering pairs of near neighbors $x_i$ and $x_j$ in the training set. Consider the predicted tangent plane of the manifold at $x_i$, characterized by the rows of $F(x_i)$. For a good predictor we expect the vector $(x_i - x_j)$ to be close to its projection on the tangent plane, with local coordinates $w \in \mathbb{R}^d$. $w$ can be obtained analytically by solving a linear system of dimension $d$. The training criterion chosen in (Bengio and Monperrus, 2005) then minimizes the sum over such $(x_i, x_j)$ of the sinus of the projection angle, i.e.:

$$\frac{||F'(x_i)w - (x_j - x_i)||^2}{||x_j - x_i||^2}$$

The above is a heuristic criterion, which will be replaced in our new algorithm by one derived from the maximum likelihood criterion, considering that $F(x_i)$ indirectly provides the principal eigenvectors of the local covariance matrix at $x_i$. Both criteria gave similar results experimentally, but the model proposed here yields a complete density estimator. In both cases $F(x_i)$ can be interpreted as specifying the directions in which one expects to see the most variations when going from $x_i$ to one of its near neighbors in a finite sample.

# 4 Proposed Algorithm: Non-Local Manifold Parzen Windows

In equations (1) and (2) we wrote $\mu(x_i)$ and $S(x_i)$ as if they were **functions** of $x_i$ rather than simply using indices $\mu_i$ and $S_i$. This is because we introduce here a non-local version of Manifold Parzen Windows inspired from the non-local manifold tangent learning algorithm, i.e., in which we can **share information about the density across different regions of space**. In our experiments we use a neural network of $nhid$ hidden neurons, with $x_i$ in input to predict $\mu(x_i)$, $\sigma^2_{noise}(x_i)$, and the $s_j(x_i)$ and $v_j(x_i)$. The vectors computed by the neural network do not need to be orthonormal: we only need to consider the subspace that they span. Also, the vectors' norm will be used to infer $s_j(x_i)$, instead of having a separate output for them. We will note $F(x_i)$ the matrix whose rows are the vectors output of the neural network. From it we obtain the $s_j(x_i)$ and $v_j(x_i)$ by performing a singular value decomposition, i.e. $FF' = \sum_{j=1}^d s_j v_j v_j'$. Moreover, to make sure $\sigma^2_{noise}$ does not get too small, which could make the optimization unstable, we impose $\sigma^2_{noise}(x_i) = s^2_{noise}(x_i) + \sigma_0^2$, where $s_{noise}(\cdot)$ is an output of the neural network and $\sigma_0^2$ is a fixed constant.

Imagine that the data were lying near a lower dimensional manifold. Consider a training example $x_i$ near the manifold. When the likelihood of the other examples is computed, only the near neighbors of $x_i$ will have a significant gradient contribution for the Gaussian associated with $x_i$, because its posterior will be very small compared to Gaussian components closer to the example. The Gaussian centered near $x_i$ tells us how neighbors of $x_i$ are expected to differ from $x_i$. Its "principal" vectors $v_j(x_i)$ span the tangent of the manifold near $x_i$. The Gaussian center variation $\mu(x_i)$ tells us how $x_i$ is located with respect to its projection on the manifold. The noise variance $\sigma^2_{noise}(x_i)$ tells us how far from the manifold to expect neighbors, and the directional variances $s_j(x_i) + \sigma^2_{noise}(x_i)$ tell us how far to expect neighbors on the different local axes of the manifold, near $x_i$'s projection on the manifold. Figure 1 shows an illustration in 2 dimensions of the shape of the a Gaussian using this parametrization, with $s_1 = \sqrt{\lambda_i}$, $\sigma_{\text{noise}} = \sqrt{\sigma^2_{noise}(x_i)}$ and $\mu = \mu(x_i)$.

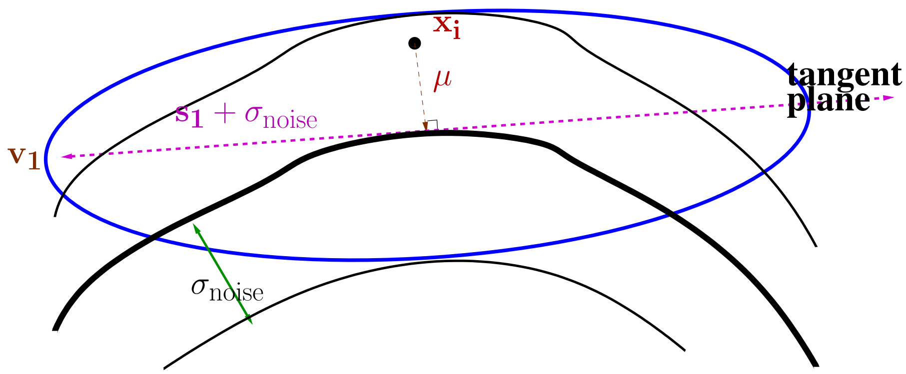



Figure 1: *Illustration of the local parametrization of local or Non-Local Manifold Parzen. The examples around training point $x_i$ are modeled by a Gaussian. $\mu(x_i)$ specifies the center of that Gaussian, which should be non-zero when $x_i$ is off the manifold. $v_k$'s are principal directions of the Gaussian and are tangent vectors of the manifold. $\sigma_{\text{noise}}$ represents the thickness of the manifold.*

The important element of this model is that the parameters of the predictive neural network can potentially represent non-local structure in the density, i.e., they allow to potentially discover shared structure among the different covariance matrices in the mixture.

We propose to estimate $\mu(x)$ and $S(x)$ by minimizing the regularized negative log-likelihood of the $k$ neighbors $x_j$ of each training point $x_i$, according to the Gaussian with mean $x_i + \mu(x_i)$ and covariance matrix $S(x_i)$. In the implementation made for the experiments reported here, training proceeds by stochastic gradient, visiting each example $x_i$ (with all of its neighbors) and making a parameter update.

## 4.1 Parametrization of $\mu(x)$ and $S(x)$

The $\mu(x)$ function is simply the output of a neural net function of $x$. The $S(x)$ function is more difficult to parametrize. As explained earlier, in order to capture information about the underlying manifold of the data, we will parametrize $S(x)$ as follows:

$$S(x) = \sigma^2_{noise}(x)I + \sum_{j=1}^{d} F_j(x)' F_j(x)$$

where $d$ is the number of principal directions of variance. Note that the vectors $F_j(x)$ do not need to be orthogonal. The neural net will then have to ouput the $d \times n$ matrix $F(x)$ and the scalar $\sigma^2_{noise}(x)$. To make sure that $\sigma^2_{noise}(x)$ is positive and does not get too small, which could make the stochastic gradient optimization unstable, we impose $\sigma^2_{noise}(x_i) = (s_{noise}(x_i))^2 + \sigma_0^2$, where $s_{noise}(\cdot)$ is an output of the neural network and $\sigma_0^2$ is a fixed constant.

When computing $S^{-1}(x)$, we simply compute the $d$ eigenvalues $\lambda_i$ and eigenvectors $v_i$ of $F(x)'F(x)$ and get:

$$S^{-1}(x) = \frac{1}{\sigma^2_{noise}(x)} I + \sum_{j=1}^{d} \left( \frac{1}{\lambda_j + \sigma^2_{noise}(x)} - \frac{1}{\sigma^2_{noise}(x)} \right) v_j v_j'$$

Figure 1 shows an illustration in 2 dimensions of the shape of the a Gaussian using this parametrization, with $s_1 = \sqrt{\lambda_i}$, $\sigma_{\text{noise}} = \sqrt{\sigma^2_{noise}(x_i)}$ and $\mu = \mu(x_i)$.

## 4.2 Gradient derivation for $\mu(x)$ and $S(x)$

We consider here the error signal for a pair consisting of a training example $x$ and a neighbor $y$ of $x$.

Consider the derivative of $NLL(y) = -\log(p(y))$ with respect to $\log(p(y|x))$:

$$\frac{\partial(-\log(p(y)))}{\partial(\log(p(y|x)))} = -\frac{p(y|x)p(x)}{p(y)} = -p(x|y)$$

which also corresponds to the 'posterior' factor in EM. The conditional log-likelihood $\log(p(y|x))$ for a neighbor $y$ of an example $x$ is written:

$$\log(p(y|x)) = -0.5 \log(|S(x)|) - (n/2)\log(2\pi) - 0.5(y - x - \mu(x))' S(x)^{-1} (y - x - \mu(x)).$$

Hence

$$\begin{aligned} \frac{\partial NLL(y)}{\partial \theta} &= 0.5 p(x|y) \left[ \frac{\partial}{\partial \theta} \log(|S(x)|) \right. \\ &\quad \left. + \frac{\partial}{\partial \theta} (y - x - \mu(x))' S(x)^{-1} (y - x - \mu(x)) \right] \end{aligned}$$

which gives significant weight only to the near neighbors $y$ of $x$. This justifies the approximation used here:

$$\begin{aligned} \frac{\partial NLL(y)}{\partial \theta} &\approx 0.5 \, \frac{\mathbb{1}_{y \in \mathcal{N}_k(x)}}{k} \left[ \frac{\partial}{\partial \theta} \log(|S(x)|) \right. \\ &\quad \left. + \frac{\partial}{\partial \theta} (y - x - \mu(x))' S(x)^{-1} (y - x - \mu(x)) \right] \end{aligned} \qquad (3)$$

where $\mathcal{N}_k(x)$ is the set of the $k$ nearest neighbors of $x$.

Now, we can compute the gradient for $\theta = \mu(x)$ and $\theta = S(x)$. It is straightforward to show that:

$$\frac{NLL(y)}{\partial\mu(x)} \approx -0.5\,\frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}S(x)^{-1}(y-x-\mu(x))$$

For $S(x)$, the derivation is less obvious. Let's start with $\sigma^2_{noise}(x)$. We have from equation 3:

$$\begin{aligned}
\frac{\partial NLL(y)}{\partial\sigma^2_{noise}} &\approx 0.5\,\frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\left[\frac{\partial}{\partial\sigma^2_{noise}}\log(|S(x)|)\right.\\
&\left.+\frac{\partial}{\partial\sigma^2_{noise}}(y-x-\mu(x))'S(x)^{-1}(y-x-\mu(x))\right]\\
&\approx 0.5\,\frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\left[\frac{|S(x)|}{|S(x)|}\mathrm{Tr}\left(S(x)^{-1}\frac{\partial}{\partial\sigma^2_{noise}}S(x)\right)\right.\\
&\left.-(y-x-\mu(x))'S(x)^{-1}\left(\frac{\partial}{\partial\sigma^2_{noise}}S(x)\right)S(x)^{-1}(y-x-\mu(x))\right] \qquad (4)\\
&\approx 0.5\,\frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\left[\mathrm{Tr}\left(S(x)^{-1}\right)-(y-x-\mu(x))'S(x)^{-1}S(x)^{-1}(y-x-\mu(x))\right]\\
&\approx 0.5\,\frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\left[\mathrm{Tr}\left(S(x)^{-1}\right)-||(y-x-\mu(x))'S(x)^{-1}||^2\right]
\end{aligned}$$

where at equation 4, we used:

$$\frac{\partial}{\partial x}|A| = |A|\mathrm{Tr}\left(A^{-1}\frac{\partial}{\partial x}A\right) \qquad (5)$$

and

$$\frac{\partial}{\partial x}A^{-1} = -A^{-1}\left(\frac{\partial}{\partial x}A\right)A^{-1} \qquad (6)$$

Also, for $F(x)$, let $f_{ji}$ be the element at position $(j,i)$ of matrix $F(x)$. From equation 3, we have:

$$\begin{aligned}
\frac{\partial NLL(y)}{\partial f_{ji}} &\approx 0.5\,\frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\left[\frac{\partial}{\partial f_{ji}}\log(|S(x)|)\right.\\
&\left.+\frac{\partial}{\partial f_{ji}}(y-x-\mu(x))'S(x)^{-1}(y-x-\mu(x))\right]\\
&\approx 0.5\,\frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\left[\frac{|S(x)|}{|S(x)|}\mathrm{Tr}\left(S(x)^{-1}\frac{\partial}{\partial f_{ji}}S(x)\right)\right.\\
&\left.-(y-x-\mu(x))'S(x)^{-1}\left(\frac{\partial}{\partial f_{ji}}S(x)\right)S(x)^{-1}(y-x-\mu(x))\right] \qquad (7)
\end{aligned}$$

where we obtain equation 7 using equations 5 and 6. Since:

$$\frac{\partial}{\partial f_{ji}} S = (e_i F_j(x) + F_j(x)' e_i')$$

where $e_i = (\mathbb{1}_{k=i})_{k=1}^n$, we then have:

$$\begin{aligned}
\frac{\partial NLL(y)}{\partial f_{ji}} &\approx 0.5\,\frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\left[\mathrm{Tr}\left(S(x)^{-1}(e_iF_j(x)+F_j(x)'e_i')\right)\right.\\
&\quad \left. -(y-x-\mu(x))'S(x)^{-1}(e_iF_j(x)+F_j(x)'e_i')S(x)^{-1}(y-x-\mu(x))\right]\\
&\approx 0.5\,\frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\left[\mathrm{Tr}\left(S(x)^{-1}e_iF_j(x)\right)+\mathrm{Tr}\left(S(x)^{-1}F_j(x)'e_i'\right)\right.\\
&\quad -(y-x-\mu(x))'S(x)^{-1}e_iF_j(x)S(x)^{-1}(y-x-\mu(x))\\
&\quad \left. -(y-x-\mu(x))'S(x)^{-1}F_j(x)'e_i'S(x)^{-1}(y-x-\mu(x))\right]\\
&\approx \frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\left[\mathrm{Tr}\left(F_j(x)S(x)^{-1}e_i\right)-F_j(x)S(x)^{-1}(y-x-\mu(x))(y-x-\mu(x))'S(x)^{-1}e_i\right]\\
&\approx \frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}F_j(x)S(x)^{-1}\left(I-(y-x-\mu(x))(y-x-\mu(x))'S(x)^{-1}\right)e_i
\end{aligned}$$

From that equation, we finally obtain:

$$\frac{\partial NLL(y)}{\partial F} \approx \frac{\mathbb{1}_{y\in\mathcal{N}_k(x)}}{k}\,FS^{-1}\left(I+(y-x-\mu(x))(y-x-\mu(x))'S^{-1}\right)$$

Note that a different number of neighbors can be selected to train $\mu(x)$ and $S(x)$.

## 4.3 Algorithms and Optimization details

To sum up, here is the pseudo-code algorithm for computing the density at a test point with Non-Local Manifold Parzen:

**Algorithm NLMP::Test($x, X, \mu(\cdot), S(\cdot), \sigma_0^2$)**
**Input**: test point $x$, training set $X$, functions $\mu(\cdot)$ and $S(\cdot)$, and regularization hyper-parameter $\sigma_0^2$.
**(1)** $s \leftarrow 0$
**(2)** For $x_i \in X$
**(3)** $\quad s \leftarrow s + N(x; x_i + \mu(x_i), S(x_i))$
**Output**: NLMP estimator $\hat{p}_{nlmp(x)} = \frac{s}{|X|}$.

where $N(x;\mu,\Sigma)$ is the density value at $x$ for a Gaussian of mean $\mu$ and covariance $\Sigma$. Of course, for problems in high dimension, it is highly recommended to work with log-densities instead, in order to avoid decimal precision issues.

Also, here is the pseudo-code algorithm for training the Non-Local Manifold Parzen model:

**Algorithm NLMP::Train($X, d, k, k_\mu, \mu(\cdot), S(\cdot), \sigma_0^2$)**

**Input**: training set $X$, chosen number of principal directions $d$, chosen number of neighbors $k$ and $k_\mu$, functions $\mu(\cdot)$ and $S(\cdot)$, and regularization hyper-parameter $\sigma_0^2$.

**(1)** For $x_i \in X$

**(2)** Collect $k$ nearest neighbors $x_j$ and $k_\mu$ nearest neighbors $x_j^\mu$ of $x_i$ .

**(3)** Perform a stochastic gradient step on parameters of $S(\cdot)$, using the negative log-likelihood error signal on the $x_j$, with a Gaussian of mean $x_i + \mu(x_i)$ and of covariance matrix $S(x_i)$. Do the same for the parameters of $\mu(\cdot)$ using the $x_j^\mu$

The gradients are:

$$\frac{\partial C(x_j^\mu, x_i)}{\partial \mu(x_i)} = -\frac{1}{k_\mu} S^{-1}(x_i)(x_j^\mu - x_i - \mu(x_i))$$

$$\frac{\partial C(x_j, x_i)}{\partial \sigma^2_{noise}(x_i)} = 0.5\frac{1}{k}\left(Tr(S^{-1}(x_i)) - ||(x_j - x_i - \mu(x_i))' S^{-1}(x_i)||^2\right)$$

$$\frac{\partial C(x_j, x_i)}{\partial F(x_i)} = \frac{1}{k} F(x_i) S^{-1}(x_i)\left(I - (x_j - x_i - \mu(x_i))(x_j - x_i - \mu(x_i))' S^{-1}(x_i)\right)$$

**(4)** Go to **(1)** until a given criterion is satisfied (e.g. average NLL of NLMP density estimation on a validation set stops decreasing)

**Result**: trained $\mu(\cdot)$ and $S(\cdot)$ functions, with corresponding $\sigma_0^2$.

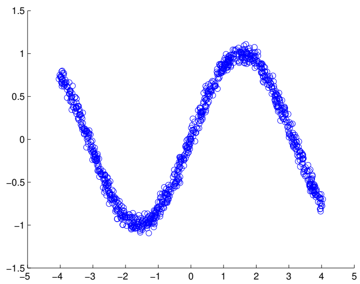
Optimization can get quite unstable if $\sigma^2_{noise}(x)$ is trained without constraints on its progression. The reason is that, when $\sigma^2_{noise}(x)$ gets too small and that the principal directions are not correctly estimated, then the term $(x_j - x_i - \mu(x_i))' S^{-1}(x_i)$ gets quite large, giving a gradient step that is too big. To escape this problem, we imposed a threshold on the error signal through $\sigma^2_{noise}(x)$. When the absolute value of the error signal is bigger than a certain proportion $\alpha$ of the $\sigma^2_{noise}(x)$ value, the error signal is fixed to that threshold, with the appropriate sign. In our experiments, we used $\alpha = 0.1$. Also, it is quite useful to set the initial value of $\sigma^2_{noise}(x)$, in order to make sure that it is not too small, so that the optimization is stable, and that it is not too big, so that the optimization does not take too long.

# 5 Experimental results

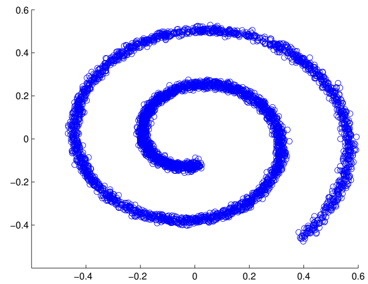
We have performed comparative experiments on both toy and real-world data, on density estimation and classification tasks. All hyper-parameters are selected by cross-validation, and the costs on a large test set is used to compare final performance of all algorithms.

## 5.1 Experiments on toy 2D data

To understand and validate the non-local algorithm we tested it on toy 2D data where it is easy to understand what is being learned. The **sinus** data set (see figure 2(a)) includes examples sampled from a sinusoidal distribution with uniformly distributed noise. In the **spiral** data set (see figure 2(b)) examples are sampled near a spiral. Respectively, 57 and 113 examples are used for training, 23 and 48 for validation (hyper-parameter selection), and 920 and 3839 for testing.




(a) Sinusoidal data set




(b) Spiral data set

Figure 2: Distribution samples of data sets

| Algorithm | sinus ANLL | spiral ANLL |
|---|---|---|
| **Non-Local Manifold Parzen** | **1.144** | **-1.346** |
| Manifold Parzen | 1.345 | -0.914 |
| Gaussian Mixture | 1.567 | -0.857 |
| Parzen Windows | 1.841 | -0.487 |

Table 1: *Average out-of-sample Negative Log-Likelihood (ANLL) on two toy problems, for Non-Local Manifold Parzen, a Gaussian mixture with full covariance, Manifold Parzen, and Parzen Windows. The non-local algorithm dominates all the others.*

The following algorithms were compared:

• Non-Local Manifold Parzen Windows. The hyper-parameters are the number of principal directions (i.e., the dimension of the manifold), the number of nearest neighbors $k$ and $k_\mu$, the minimum constant noise variance $\sigma_0^2$ and the number of hidden units of the neural network. The starting learning rate and decrease constant also have to be selected, and early stopping is used on the validation set.

• Gaussian mixture with full but regularized covariance matrices. Regularization is done by setting a minimum constant value $\sigma_0^2$ to the eigenvalues of the Gaussians. It is trained by EM and initialized using the k-means algorithm. The hyper-parameter is $\sigma_0^2$, and early stopping is used with EM on a validation set.

• Parzen Windows density estimator, with a spherical Gaussian kernel. The hyper-parameter is the spread of the Gaussian kernel.

• Manifold Parzen density estimator. The hyper-parameters are the number of principal components, the number of nearest neighbors of the k-nearest neighbors kernel and the minimum eigenvalue $\sigma_0^2$.

Note that, for these experiments, the number of principal directions (or components) was fixed to 1 for both NLMP and Manifold Parzen.

Density estimation results are shown in table 1. To help understand why Non-Local Manifold Parzen works well on these data, figure 3 illustrates the learned densities for the sinus and spiral data. Basically, it works better here because it yields an estimator that is less sensitive to the specific samples around each test point, thanks to its ability to share structure across the whole training set. In figure 4, the principal directions learned by Non-Local

Manifold Parzen and local Manifold Parzen are displayed for the spiral distribution data. We can clearly see that the former predicts principal directions that are more consistent with the true shape of the manifold, whereas the latter is very sensible to the noise of the data.

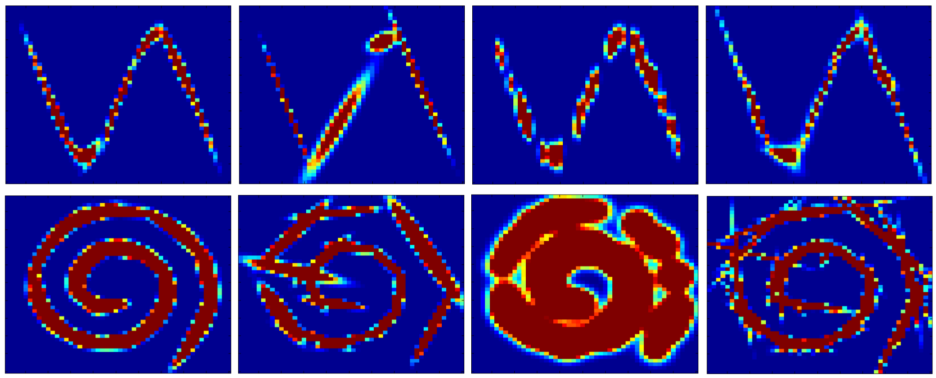
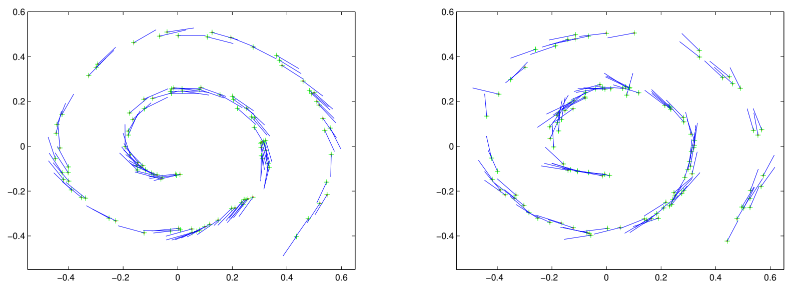


Figure 3: *Illustration of the learned densities (sinus on top, spiral on bottom) for four compared models. From left to right: Non-Local Manifold Parzen, Gaussian mixture, Parzen Windows, Manifold Parzen. Parzen Windows wastes probability mass in the spheres around each point, while leaving many holes. Gaussian mixtures tend to choose too few components to avoid overfitting. The Non-Local Manifold Parzen exploits global structure to yield the best estimator.*

## 5.2 Experiments on rotated digits

The next experiment is meant to show both qualitatively and quantitatively the power of non-local learning, by using 9 classes of rotated digit images (from 729 first examples of the USPS training set) to learn about the rotation manifold and testing on the left-out class (digit 1), not used for training. Each training digit was rotated by 0.1 and 0.2 radians and all these images were used as training data. We used NLMP for training, and for testing we formed an augmented mixture with Gaussians centered not only on the training examples, but also on the original unrotated 1 digits. We tested our estimator on the rotated versions of each of the 1 digits. We compared this to Manifold Parzen trained on the training data containing both the original and rotated images of the training class digits and the unrotated 1 digits. During the test, we forced the Gaussian components to be centered on the test point. We tested our estimator on the rotated versions of each of the original images. We compared it to Manifold Parzen where the parameters of the Gaussian components for digit 1 were computed as usual, by including the unrotated 1 digits to its training set. The objective of the experiment was to see if the model was able to infer the density correctly around the original unrotated images, i.e., to predict a high probability for the rotated versions of these images. In table 2 we see quantitatively that the non-local estimator predicts the rotated images much better.

As qualitative evidence, we used small steps in the principal direction predicted by the neural net to rotate an image of the digit 1. To make this task even more illustrative of the generalization potential of non-local learning, we followed the tangent in the direction opposite to the rotations of the training set. It can be seen in figure 5 that the rotated digit obtained is quite similar to the same digit analytically rotated. For comparison, we tried to apply the same rotation technique to that digit, but by using the principal direction, computed by Manifold Parzen, of its nearest neighbor's Gaussian component in the training set. This clearly did not work, and hence shows how crucial




(a) Non-Local Manifold Parzen (b) Manifold Parzen

Figure 4: *Illustration of the learned principal directions for Non-Local Manifold Parzen and local Manifold Parzen, for the spiral distribution data set. Note how the principal directions are more consistent with the underlying manifold for Non-Local Manifold Parzen than for Manifold Parzen*

| Algorithm | Validation ANLL | Test ANLL |
|---|---|---|
| **Non-Local Manifold Parzen** | **-73.10** | **-76.03** |
| Manifold Parzen | 65.21 | 58.33 |
| Parzen Windows | 77.87 | 65.94 |

Table 2: *Average Negative Log-Likelihood (ANLL) on the digit rotation experiment, when testing on a digit class (1's) not used during training, for Non-Local Manifold Parzen, Manifold Parzen, and Parzen Windows. The non-local algorithm is clearly superior.*

non-local learning is for this task.

Note that, to make sure that NLMP learns the tangent plane of the rotation manifold, we fixed the number of principal directions $d$ and the number of nearest neighbors $k$ to one, and we also imposed $\mu(\cdot) = 0$. The same was done for Manifold Parzen.

## 5.3 Experiments on classification by density estimation

The USPS data set was used to perform a classification experiment. The original training set (7291) was split into a training (first 6291) and validation set (last 1000), used to tune hyper-parameters. One density estimator for each of the 10 digit classes is estimated. For comparison we also show the results obtained with a Gaussian kernel Support Vector Machine (already used in (Vincent and Bengio, 2003)). The improvement brought by the Non-Local Manifold Parzen algorithm over the other methods is statistically significant with a $t$-test on the test set of 2007 examples. Note that better SVM results (about 3% error) can be obtained using prior knowledge about image invariances, e.g. with virtual support vectors (Decoste and Scholkopf, 2002). However, as far as we know the NLMP performance is the best on the original USPS data set among algorithms that do not use prior knowledge about images.

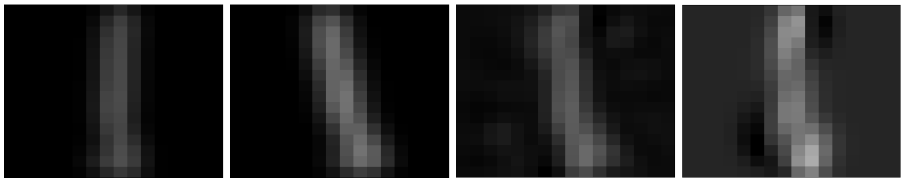


Figure 5: *From left to right: original image of a digit 1; same image analytically rotated by $-0.2$ radians; rotation predicted using Non-Local Manifold Parzen; rotation predicted using local Manifold Parzen. The rotations are obtained by following the tangent vector in small steps.*

Table 3: *Classification error obtained on USPS with SVM, Parzen Windows and Local and Non-Local Manifold Parzen Windows classifiers.*

| Algorithm | Valid. | Test | Parameters |
|---|---|---|---|
| SVM | 1.2% | 4.68% | $C = 100$, $\sigma = 8$ |
| Parzen Windows | 1.8% | 5.08% | $\sigma = 0.8$ |
| Manifold Parzen | 0.9% | 4.08% | $d = 11$, $k = 11$, $\sigma_0^2 = 0.1$ |
| **Non-local MP** | **0.6%** | **3.54%** | $d = 7$, $k = 10$, $k_\mu = 4$, $\sigma_0^2 = 0.05$, $nhid = 30$ |

# 6 Conclusion

We have proposed a non-parametric density estimator that, unlike its predecessors, is able to generalize far from the training examples by capturing global structural features of the density. It does so by learning a function with global parameters that successfully predicts the local shape of the density, i.e., the tangent plane of the manifold along which the density concentrates. Three types of experiments showed that this idea works, yields improved density estimation and classification error compared to its local predecessors.


### 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 report, with Sam Roweis and Olivier Delalleau.


# References


Bengio, Y., Delalleau, O., and Le Roux, N. (2005). The curse of dimensionality for local kernel machines. Technical Report 1258, Département d'informatique et recherche opérationnelle, Université de Montréal.

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.

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

Decoste, D. and Scholkopf, B. (2002). Training invariant support vector machines. *Machine Learning*, 46:161–190.

Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2005). Neighbourhood component analysis. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*. MIT Press.

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

Roweis, S., Saul, L., and Hinton, G. (2002). Global coordination of local linear models. In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

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

Teh, Y. W. and Roweis, S. (2003). Automatic alignment of local representations. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT Press.

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

Vincent, P. and Bengio, Y. (2003). Manifold parzen windows. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.