Gaussian fields for semi-supervised regression and correspondence learning

Jakob Verbeek, Nikos Vlassis

Gaussian fields for semi-supervised regression and correspondence learning

Jakob J. Verbeek and Nikos Vlassis

Intelligent Systems Laboratory Amsterdam, University of Amsterdam
Kruislaan 403, 1098 SJ, Amsterdam, The Netherlands

Abstract

Gaussian fields (GF) have recently received considerable attention for dimension reduction and semi-supervised classification. In this paper we show how the GF framework can be used for semi-supervised regression on high-dimensional data. We propose an active learning strategy based on entropy minimization and a maximum likelihood model selection method. Furthermore, we show how a recent generalization of the LLE algorithm for correspondence learning can be cast into the GF framework, which obviates the need to choose a representation dimensionality.

Key words: Gaussian fields, regression, active learning, model selection

1 Introduction

Learning from high dimensional data is a challenging problem that becomes increasingly important as large and high dimensional data collections need to be analyzed in different application domains. Such data collections appear in home and industrial settings, for example in document collections, image databases, and audio recordings. In this paper we consider Gaussian fields (GFs) defined over high dimensional input data for two learning problems.

The first problem is semi-supervised regression on high dimensional data that exhibits a low dimensional structure. For example, the input data can be images of a face looking in different directions and we want to map these images to pose parameters. If we consider the images as a set of pixel values forming a vector, then the input vectors have a dimensionality that equals the number of pixels in the image. However, in this high dimensional space the data are confined to an embedding of a low dimensional manifold that contains the images of the face looking in different directions. In this setting, unsupervised data (with unknown pose parameters) can be exploited to identify the low

dimensional manifold structure of the data. Due to the low dimensionality of the manifold, we expect that only few supervised examples are needed to successfully learn a function on the manifold to predict the pose parameters.

The second problem we consider is correspondence learning. In this problem two high-dimensional data sets are given that share the same underlying modes of variability. For example, the sets can be collections of images of different views of objects x and y, or the sets can be collections of images of persons x and y showing different facial expressions. The manifolds on which the two data sets lie, parameterized by the shared modes of variability (camera pose or facial expression), can be aligned if some 'correspondences' are given. A correspondence is a pair that consists of a point in the first data set and one in the second, which are known to share the same underlying low-dimensional coordinate (e.g. images of the first and second object viewed from the same direction, see Fig. 4). The goal is to predict for a data point in one set the corresponding observation in the other set. This problem is similar to regression, except that in this case the 'output' is also high dimensional. Also in this setting, the low dimensional manifolds can be recovered using unsupervised data, and can then be aligned using only relatively few correspondences due to the low dimensionality of the manifolds.

In order to successfully learn regression functions from high dimensional data either an estimation bias must be introduced by using a limited class of regression models (e.g. linear models), or alternatively, large quantities of data are required to reduce the estimation variance induced by larger model class. Recently, non-parametric techniques have been proposed [1–6] for unsupervised dimension reduction and semi-supervised classification that are based on a nearest neighbor graph on the set of high-dimensional 'inputs'. The graph on the data encodes the assumption that nearest neighbors in the input space tend to have similar 'outputs' (low-dimensional coordinates for the inputs in case of dimension reduction, and class labels in the case of semi-supervised classification). These techniques are based on a non-negative energy function that sums terms that are quadratic in the outputs of neighbors in the graph. With the energy function we can associate a Gaussian field (GF); a probability density function on the outputs proportional to the exponent of the negative energy function. The associated GF makes it possible to quantify uncertainty in the predictions which is useful for active learning. The main difference between the graph-based approaches and more traditional approaches lies in how the learning bias is introduced. Graph based methods are biased towards smooth functions over the graph; in this manner the bias is derived from —and reflects the structure of— the given inputs. Traditional techniques introduce a bias by limiting attention to some parametric model class. Notably, the choice of a specific function class is often motivated by reasons of computational efficiency rather than the suitability of this class for the given problem.

The rest of this paper is organized as follows. In the next section we review GFs defined on nearest neighbor graphs. In Section 3 we discuss related methods for unsupervised dimension reduction and semi-supervised classification. In section Section 4 we focus on the first learning problem: semi-supervised regression using GFs. We propose a maximum likelihood model selection method to set the parameters of the GF. Furthermore, we propose an entropy minimization based query selection procedure for active learning. We present experimental results that show the effectiveness of both procedures and the advantage of the GF method over a method that separates unsupervised dimension reduction and supervised learning. In Section 5 we turn to correspondence learning using GFs. We propose a new approach for the correspondence learning based on GFs, and experimentally compare it to an existing approach [7] based on the Locally Linear Embedding (LLE) algorithm[1]. The GF approach performs as good as or better than the LLE-based method, while the GF-based approach has one parameter less: it does not require the setting of a representation dimensionality. We end the paper with conclusions and a discussion in Section 6.

2 Gaussian fields defined over nearest neighbor graphs

In this section we review Gaussian fields defined over neighborhood graphs. Given a set of high-dimensional 'inputs' $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, the first step in defining the GF is to find the k nearest neighbors of each $\mathbf{x}_i$. Throughout this paper we define nearest neighbors on the basis of Euclidean distance (selecting the value of k is the topic of Section 4.1). Note that it is not necessary to exhaustively compute all pairwise distances to find the nearest neighbors, see e.g. [8].

Let y_i denote some scalar 'output' of $\mathbf{x}_i$, and let $\mathbf{y} = (y_1, \ldots, y_n)^\top$ denote the n-dimensional vector containing all outputs. Given the nearest neighbors, we define an energy function $E(\mathbf{y})$ by summing over all input pairs $\mathbf{x}_i$ and $\mathbf{x}_j$ which are nearest neighbors, the squared difference of their outputs $(y_i - y_j)^2$ times a weighting factor a_{ij}. If we let $a_{ij} \neq 0$ only for nearest neighbors, then:

$$E(\mathbf{y}) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(y_i - y_j)^2 = \mathbf{y}^\top (\mathbf{D} - \mathbf{A})\mathbf{y} = \mathbf{y}^\top \mathbf{L}\mathbf{y}, \tag{1}$$

where the degree matrix $\mathbf{D}$ is a diagonal matrix containing the row sums of the adjacency matrix $\mathbf{A}$. The matrix $\mathbf{L}$ is known as the graph Laplacian of $\mathbf{A}$. Note that if $\forall_{i,j} a_{ij} \geq 0$ then $\forall_{\mathbf{y}} E(\mathbf{y}) \geq 0$, i.e. $\mathbf{L}$ is positive definite. However the converse is not true: matrices $\mathbf{A}$ with negative entries exist with $\forall_{\mathbf{y}} E(\mathbf{y}) \geq 0$. For example, in the Locally Linear Embedding (LLE) algorithm [1] the following adjacency matrix is used

$$\mathbf{A} = \mathbf{W} + \mathbf{W}^\top - \mathbf{W}^\top \mathbf{W}, \tag{2}$$

where each row of $\mathbf{W}$ sums to 1 and $w_{ij} \neq 0$ only for nearest neighbors. Using this adjacency matrix we have $\mathbf{D} = \mathbf{I}$, and the energy $E(\mathbf{y})$ can be written as:

$$E(\mathbf{y}) = \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} w_{ij} y_j)^2 \geq 0. \tag{3}$$

Clearly, the corresponding Laplacian $\mathbf{L}$ is positive definite, while some entries of $\mathbf{A}$ can be negative (typically for pairs that are not neighbors in the graph but share one or more common neighbors).

The GF that we will define over the outputs is obtained by taking the exponent of the negative energy function. In this way the a zero mean Gaussian density is obtained, whose *inverse* covariance matrix is given by the coefficient matrix of the energy function. However, note that the energy functions defined in (1) and (3) do not define a proper GF, since $\mathbf{L}$ is singular (the constant vector $\mathbf{1}$ is an eigenvector with eigenvalue zero). By adding a regularizer we ensure positive definiteness:

$$E(\mathbf{y}) = \mathbf{y}^\top (\mathbf{L} + \alpha \mathbf{I}) \mathbf{y} = \mathbf{y}^\top \mathbf{M} \mathbf{y}. \tag{4}$$

It is easy to see that $\mathbf{M}$ has the same eigenvectors as $\mathbf{L}$, with the eigenvalues increased by α. In effect, the regularizer replaces the infinite variance in the direction $\mathbf{1}$ (if $\mathbf{L}$ is used as inverse covariance matrix) of the GF density with variance α^{-1}. Thus, for the given points $\mathbf{X}$ we define the GF density over outputs $\mathbf{y}$ based on the energy (4) as:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; 0, \beta^{-1}\mathbf{C}) \propto \exp -\frac{\beta}{2} E(\mathbf{y}), \tag{5}$$

where β is a scale parameter that controls the smoothness of the density and $\mathbf{C}$ denotes the inverse of $\mathbf{M}$. Note that although the inverse covariance matrix $\mathbf{M}$ is sparse, it generally induces a full covariance matrix $\beta^{-1}\mathbf{C}$.

The joint GF density over all outputs induces conditional and marginal distributions over sets of outputs, which will be used for active semi-supervised learning in Section 4. In semi-supervised learning the value of outputs y_i of some data points $\mathbf{x}_i$ are known and thus fixed to these values. Without loss of generality we can assume that all indices of supervised points are smaller than those of the unsupervised points. We can thus use the partition $\mathbf{y} = [\mathbf{y}_s^\top \mathbf{y}_u^\top]^\top$, where $\mathbf{y}_u$ denotes the sub-vector for unsupervised outputs and $\mathbf{y}_s$ the sub-vector for the supervised outputs. Let $\mathbf{M}_{ss}, \mathbf{M}_{us}$ and $\mathbf{M}_{uu}$ denote the corresponding blocks of $\mathbf{M}$:

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_{ss} & \mathbf{M}_{su} \\ \mathbf{M}_{us} & \mathbf{M}_{uu} \end{pmatrix}. \tag{6}$$

Throughout we will use this subscripting to refer to blocks of matrices and vectors that correspond to supervised and unsupervised points. Using the partitions, the energy (4) can be expanded as:

$$E(\mathbf{y}) = \mathbf{y}_u^\top \mathbf{M}_{uu} \mathbf{y}_u + \mathbf{y}_s^\top \mathbf{M}_{ss} \mathbf{y}_s + 2\mathbf{y}_u^\top \mathbf{M}_{us} \mathbf{y}_s, \tag{7}$$

from which it is easy to find the conditional density

$$p(\mathbf{y}_u | \mathbf{y}_s) \propto \exp -\frac{\beta}{2} \left(\mathbf{y}_u^\top \mathbf{M}_{uu} \mathbf{y}_u + 2\mathbf{y}_u^\top \mathbf{M}_{us} \mathbf{y}_s \right). \tag{8}$$

This conditional density can be used to predict the unsupervised outputs by computing the mean $\mathbf{y}_u^*$ (which equals the mode) of this density:

$$\mathbf{y}_u^* = -\mathbf{M}_{uu}^{-1} \mathbf{M}_{us} \mathbf{y}_s. \tag{9}$$

We note here that to solve (9) in practice, we can use the (sparse) Cholesky decomposition $\mathbf{M}_{uu} = \mathbf{R}^\top \mathbf{R}$, rather than inverting $\mathbf{M}_{uu}$ (which yields a full matrix). We then first solve for $\mathbf{R}^\top \mathbf{z} = -\mathbf{M}_{us} \mathbf{y}_s$, and then for $\mathbf{R}\mathbf{y}_u^* = \mathbf{z}$. Several quantities which need to be evaluated in the techniques developed in the next sections can be computed using Cholesky factors in a similar manner.

3 GFs for dimension reduction and classification

In this section we discuss related non-parametric methods for dimension reduction and semi-supervised classification.

Two recent approaches for unsupervised non-linear dimension reduction can be cast in the GF framework. The LLE algorithm [1] and the Laplacian Eigenmap algorithm [3] perform dimension reduction by minimizing $E(\mathbf{y})$, as defined in (3) and (1) respectively. Note that there is a trivial solution $\mathbf{y} = c\mathbf{1}$, which is an eigenvector of $\mathbf{L}$ with eigenvalue zero. In order to prevent the trivial solution we can minimize the energy under the constraint that the entries in $\mathbf{y}$ are zero mean and have unit variance. The minimizer is then given by the eigenvector of $\mathbf{L}$ with the second smallest eigenvalue. To extract $(d+1)$ rather than d features, one can minimize $E(\mathbf{y})$ under the constraint that $\mathbf{y}_{d+1}$ should be orthogonal to the already extracted eigenvectors $\{\mathbf{y}_i\}_{i=1}^d$. The resulting minimizing vectors $\mathbf{y}_i$ are all eigenvectors of $\mathbf{L}$, with increasing eigenvalues.

Given the relationship between $\mathbf{L}$ and the covariance matrix of the GF, these algorithms can thus be interpreted as computing the *principal components* of the GF density over the outputs. Within the space of all possible output vectors $\mathbf{y}$, the principal components span the linear subspace which contains most of the output variation under the GF density.

For two-class semi-supervised classification, the value of the known outputs in $\mathbf{y}_s$ is either 0 or 1, denoting the class membership. In [5], classification is performed by minimizing the energy $E(\mathbf{y})$, as defined in (1), with respect to $\mathbf{y}_u$ for fixed $\mathbf{y}_s$. The minimizer $\mathbf{y}_u^*$ is characterized by the linear equation:

$$\mathbf{L}_{uu}\mathbf{y}_u^* = -\mathbf{L}_{us}\mathbf{y}_s. \tag{10}$$

The unlabelled points are then assigned to class 0 if their value in $\mathbf{y}_u^*$ is smaller than 1/2, otherwise they are assigned to class 1. If we ignore the regularization with α, this approach computes the mean of the conditional density $p(\mathbf{y}_u|\mathbf{y}_s)$ and thresholds the values of the conditional mean at 1/2.

In other work on semi-supervised classification [6], $\mathbf{y}$ is constrained to be in the span of the m eigenvectors of $\mathbf{L}$ with smallest eigenvalues, i.e. $\mathbf{y} = \mathbf{V}\mathbf{w}$ where $\mathbf{V}$ contains the m eigenvectors as columns. The squared error $\|\mathbf{y}_s - \mathbf{V}_s\mathbf{w}\|^2$ is minimized w.r.t. $\mathbf{w}$, where $\mathbf{V}_s$ contains the rows of $\mathbf{V}$ corresponding to the supervised points; let $\mathbf{w}^*$ denote the minimizer. Classification is performed by computing $\mathbf{y}_u^* = \mathbf{V}_u\mathbf{w}^*$, the values in $\mathbf{y}_u$ are again thresholded at 1/2. In effect this approach first performs unsupervised non-linear dimension reduction using all data, as in [1,3]. Second, in the low dimensional representation obtained in the first step, a linear classifier is found from the supervised data.

In [4] the supervised labels are not used as a hard constraint. Rather, if we let $\boldsymbol{\gamma}_s$ denote the supervised labels, the following error function is minimized: [1]

$$E_J(\mathbf{y}) = \mathbf{y}^\top\mathbf{L}\mathbf{y} + (\mathbf{y}_s - \boldsymbol{\gamma}_s)^\top\boldsymbol{\Gamma}(\mathbf{y}_s - \boldsymbol{\gamma}_s). \tag{11}$$

The first term encourages solutions that are smooth in the sense of having similar values for nearest neighbors in the graph, while the second term favors solutions that comply with the supervised labels. The trade-off between smoothness and agreement with the supervised labels, as well as different misclassification costs for each supervised data point, can be encoded in the diagonal matrix $\boldsymbol{\Gamma}$.

[1] This is a slightly simplified account; in [4] solutions are constrained to be linear combinations of the smallest eigenvectors of $\mathbf{L}$ and to satisfy $\mathbf{y}^\top\mathbf{1} = 0$ and $\mathbf{y}^\top\mathbf{y} = n$.

4 Active semi-supervised regression with Gaussian fields

In this section we present two contributions in using GFs for semi-supervised regression. First, we propose a MAP criterion to automatically set the model parameters β and k. Second, we propose a minimum entropy query selection procedure for active learning. For clarity, we focus on regression using GFs for a single response variable. The presented material is straightforwardly generalized to prediction multiple response variables independently using the same Gaussian field.

In the related work on semi-supervised classification that we reviewed in Section 3 [4–6], the adjacency matrix $\mathbf{A}$ that determines the energy function was specified directly by assigning non-negative entries for nearest neighbors. As we argue next, for regression it is more useful to define $\mathbf{A}$ through a weight matrix $\mathbf{W}$, as in (2). We set $w_{ij} = 1/k$ if $\mathbf{x}_j$ is among the k nearest neighbors of $\mathbf{x}_i$ and $w_{ij} = 0$ otherwise. In this way the energy function sums for each point *the squared difference of its output and the average of its neighbors*, cf. (3). In comparison, if we directly set the entries $\mathbf{A}$ corresponding to nearest neighbors to $1/k$ then the energy function sums for each point *the sum of squared differences between its output and that of its neighbors*.

The apparently small difference to define $\mathbf{A}$ has an important consequence for extrapolation performance of regression using the GF. The effect is best understood through a simple example illustrated in Fig. 1. The input data consists of 300 data points along a spiral. The desired output of each point is its position if the data points are numbered along the spiral. For three data points (shown as circles) the desired output is specified. When $\mathbf{A}$ is specified directly, there is no force that drives extrapolation: if all points toward the end of the spiral are assigned the same output as the last supervised point, the contribution to the energy induced by these points is zero. In contrast, when $\mathbf{A}$ is defined in the LLE-based manner c.f. (2), the data points towards the end of the spiral are assigned higher values to ensure that the average at the last supervised point best matches the supervised value. This is necessary since the unsupervised points at the other side of the last supervised point are assigned smaller values for the interpolation between the supervised points.

Because of the lack of extrapolation in the direct method, we use the LLE based energy function (3) in the remainder of this paper, with $w_{ij} = 1/k$ when $\mathbf{x}_j$ is one of the k neighbors of $\mathbf{x}_i$. Note that the extrapolation effect does not appear in the case of semi-supervised classification, since then desired outputs are always in the range of the supervised outputs (i.e. between zero and one).

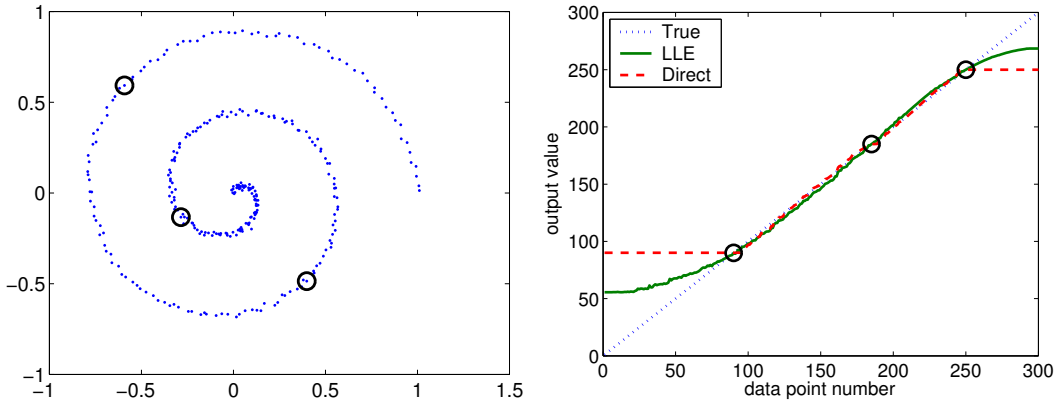


Fig. 1. Input data distributed along spiral (left panel). True and predicted output for the data (right panel). The supervised points are encircled in both panels. Predictions are made using the direct and LLE based method to specify matrix $\mathbf{A}$.

4.1 Model selection

The GF density (5) is parameterized by k, the number of nearest neighbors connected to each point, and the variance scale parameter β. Since it is hard to set these parameters by hand, it is desirable to find optimal values automatically. We can use the maximum a-posteriori (MAP) criterion to select appropriate values for k and β.

The posterior distribution on the parameters given the supervised outputs collected in $\mathbf{y}_s$ is $p(k, \beta|\mathbf{y}_s) \propto p(\mathbf{y}_s|k, \beta)p(k, \beta)$. Below we assume a uniform prior $p(k, \beta)$ over the parameters. In this case MAP parameter estimation reduces to maximum marginal likelihood parameter estimation, i.e. find $\arg\max_{k,\beta} p(\mathbf{y}_s|k, \beta)$. Note that if the weights w_{ij} are not fixed to $1/k$ but set to maximize the marginal likelihood, then a uniform prior on the parameters cannot be used. This is because the set of models with $k + 1$ neighbors then includes the set of models with k neighbors as a subset.

Since $p(\mathbf{y}|\beta, k)$ is Gaussian, the marginal likelihood of the supervised points is obtained relatively easily. Using n_s to denote the number of supervised points, the marginal log-likelihood of $\mathbf{y}_s$, given k and β, is given by:

$$\log p(\mathbf{y}_s|\beta, k) = -\frac{1}{2}\left[\log|\mathbf{C}_{ss}| - n_s \log\beta + \beta\mathbf{y}_s^\top \mathbf{C}_{ss}^{-1}\mathbf{y}_s,\right], \tag{12}$$

where $\mathbf{C}_{ss}$ depends on k. Through differentiation, the maximum of the marginal log-likelihood over β and the maximizer β^* can be analytically obtained as:

$$\beta^* = \arg\max_{\beta} \log p(\mathbf{y}_s|\beta, k) = n_s/(\mathbf{y}_s^\top \mathbf{C}_{ss}^{-1}\mathbf{y}_s), \tag{13}$$

which can be substituted in (12) to find

$$l^* = \log p(\mathbf{y}_s|\beta^*, k) = -\frac{1}{2}\left[\log|\mathbf{C}_{ss}| + n_s + n_s \log(\mathbf{y}_s^\top \mathbf{C}_{ss}^{-1}\mathbf{y}_s/n_s)\right]. \quad (14)$$

Thus, to find the optimal number of neighbors k^* we can evaluate (14) for $k = \{1, 2, \ldots, k_{max}\}$ and pick the value of k for which (14) is maximum.

To evaluate (14), we need to find $\mathbf{C}_{ss}$, which can be computed without explicitly computing all entries of $\mathbf{C}$. Recall that $\mathbf{C} = \mathbf{M}^{-1}$, then using the Cholesky decomposition $\mathbf{M} = \mathbf{R}^\top\mathbf{R}$ we can find the required entries of $\mathbf{C}_{ss}$ by noting that $\mathbf{C}_{ij} = \mathbf{e}_i^\top\mathbf{C}\mathbf{e}_j = \mathbf{e}_i^\top(\mathbf{R}^\top\mathbf{R})^{-1}\mathbf{e}_j \triangleq \mathbf{s}_i^\top\mathbf{s}_j$, where $\mathbf{e}_i$ is a vector with all zeros except for the i-th entry which equals 1, and $\mathbf{s}_i = \mathbf{R}^{-\top}\mathbf{e}_i$. Since $\mathbf{R}$ is triangular we can solve for $\mathbf{s}_i$ in time linear in n.

Maximization of the marginal likelihood over the regularizer α cannot be performed in closed form, but it is possible to use gradient based techniques. The derivative of l^* (the marginal log-likelihood maximized over β) with respect to the regularizer α equals:

$$\frac{\partial l^*}{\partial \alpha} = \frac{1}{2}\mathrm{Tr}\{\mathbf{C}_{ss}^{-1}[\mathbf{C}^2]_{ss}\} - \frac{n_s}{2}\frac{\mathbf{y}_s^\top\mathbf{C}_{ss}^{-1}[\mathbf{C}^2]_{ss}\mathbf{C}_{ss}^{-1}\mathbf{y}_s}{\mathbf{y}_s^\top\mathbf{C}_{ss}^{-1}\mathbf{y}_s}. \quad (15)$$

To find the required block of $\mathbf{C}^2$ it is not necessary to explicitly compute this product. To see this, note that $[\mathbf{C}^2]_{ij} = \mathbf{e}_i^\top\mathbf{C}\mathbf{C}\mathbf{e}_j \triangleq \mathbf{s}_i^\top\mathbf{s}_j$. Again using the Cholesky decomposition of $\mathbf{M}$ (ignoring β for clarity), we can write $\mathbf{s}_i = \mathbf{C}\mathbf{e}_i = (\mathbf{R}^\top\mathbf{R})^{-1}\mathbf{e}_i$. We can solve for $\mathbf{s}_i$ by first solving $\mathbf{R}^\top\mathbf{z} = \mathbf{e}_i$ and then for $\mathbf{R}\mathbf{s}_i = \mathbf{z}$. However, after each gradient step the Cholesky factors $\mathbf{R}$ will need to be recomputed. This makes optimization over the regularizer computationally demanding. In our experiments we therefore fixed the regularizer at $\alpha = 10^{-11}$.

4.2 Query selection

The amount of supervised data available in practice is often limited, since it is generally costly to acquire. This motivates the use of active learning approaches, which determine, on the basis of a set of unsupervised and supervised data, for which unsupervised data the supervised signal is most informative. In this way the user's effort is not wasted on answering uninformative queries, as could be the case if supervision is queried for a randomly selected inputs.

To determine which query set of fixed size is the most informative, we may consider the entropy $\mathcal{H}(\mathbf{y}_u|\mathbf{y}_s)$ of the conditional density of the unlabelled points

given a labelled query set, and aim for the query set that leads to the lowest conditional entropy (see also [9,10]). Intuitively, we want the uncertainty of our predictions on the unlabelled points, given the labelled points, to be as small as possible. The rationale is that, if we assume that the mean of the field will approach the true value of the response variable, then the distribution $p(\mathbf{y}_u|\mathbf{y}_s)$ should be concentrated as tight as possible around its mean. The chain-rule of entropies tells us that $\mathcal{H}(\mathbf{y}) = \mathcal{H}(\mathbf{y}_s) + \mathcal{H}(\mathbf{y}_u|\mathbf{y}_s)$. Since the entropy $\mathcal{H}(\mathbf{y})$ of the complete field is fixed, we can equivalently aim for the query set with the largest entropy. To compute the entropy $\mathcal{H}(\mathbf{y}_s) = \frac{1}{2}\log|\mathbf{C}_{ss}| - \frac{1}{2}n_s\log\beta + \text{const.}$ we use the Cholesky factors of $\mathbf{M}$ as above to find $\mathbf{C}_{ss}$.

To find the single variable with maximum entropy (i.e. maximum variance) we should compute all marginal variances: the diagonal elements of $\mathbf{C}$. However, already by selecting the maximum variance variable from a random subset of all variables, we have large probability of selecting one of the variables with largest variance. In fact, as in [11], we can compute confidence bounds of the type: With probability at least $1-\delta$ the variable with largest variance among m uniformly randomly selected variables is among the ε fraction of variables with largest variance, if $m \geq \lceil \log\delta / \log(1-\varepsilon) \rceil$. For example, if we use a random subset of 59 variables then with probability at least 95% the selected variable will be among the 5% of variables with the largest variances.

The number of possible query *sets* grows quickly with n_s and heuristic search methods are inevitable already when n_s is moderately large. Given some initial query set, a simple, and in practice very successful, scheme to improve the query set is the following. Until some stopping criterion is met, randomly some variable y_i not in the current query set is selected. An exchange is made if replacing one of the variables in the current query set with y_i yields a higher joint entropy of the query set. To generate the initial query set either a random subset can be used or a greedy procedure, that adds new elements to the query set one-by-one, which we describe below. When the size of the query set is not known in advance, it is also possible to use the greedy procedure by itself. In that case, when the user decides to label another data point, the query that optimally augments the current query set is determined.

Given an initial query set $s \subset \{1,\ldots,N\}$ we want to find the best query i to add to s, with $i \in u = \{1,\ldots,N\}\backslash s$ such that the entropy in the remaining variables $\mathcal{H}(\mathbf{y}_{u\backslash i}|\mathbf{y}_{\{i\}\cup s}) = \mathcal{H}(\mathbf{y}_u|\mathbf{y}_s) - \mathcal{H}(y_i|\mathbf{y}_s)$ is minimized. To do this we find the $i \in u$ for which $\mathcal{H}(y_i|\mathbf{y}_s)$ is maximum; to compute these entropies we use the Cholesky decomposition of $\mathbf{M}_{uu}$. Note that this decomposition is also used to predict $\mathbf{y}_u$ from $\mathbf{y}_s$. In cases where the Cholesky decomposition of $\mathbf{M}_{uu}$ is not available (because $\mathbf{y}_u$ is not predicted or predicted without use of the Cholesky decomposition), the decomposition of $\mathbf{M}$ can used instead. To see this, note that from the chain-rule of entropies we have that $\mathcal{H}(\mathbf{y}_{u\backslash i}|\mathbf{y}_{\{i\}\cup s}) = \mathcal{H}(\mathbf{y}) - \mathcal{H}(\mathbf{y}_{\{i\}\cup s})$. Thus to find the optimal query to add to s, we find the query

i that, together with s, has the maximum entropy $H(\mathbf{y}_{\{i\}\cup s})$ in the complete field. The entropies $H(\mathbf{y}_{\{i\}\cup s})$ are computed from the Cholesky factors of $\mathbf{M}$.

4.3 Experimental results

First we consider a face-pose estimation experiment to give a qualitative illustration of the techniques described above. The input data consists of a set of 2000 images (40×40 pixels) of a face that looks in different directions, varying both horizontally and vertically (see the images depicted in Fig. 2). [2] The goal is to predict the pose of unsupervised images given only 10 supervised images.

A neighborhood graph with 20 neighbors was used, based on Euclidean distance. The supervised data was obtained as follows: first the user has to place a random image in a two-dimensional plane. Then, using the minimum entropy principle a next query image is selected and the user places this image also in the plane. This process was repeated until ten images were placed in the plane. The two coordinates of the position in the plane of each image are considered as the supervised values of the two response variables which correspond to the horizontal and vertical viewing direction (pose) of the face.

In Fig. 2 the predicted pose parameters of all images are shown, while the coordinates of the 10 supervised images are indicated with the digits $1, \ldots, 10$. Observe that the query selection procedure mainly requested supervision for extreme poses. To illustrate the predictive power of the model, we sampled ten equally spaced points (depicted as circles) along four lines in the pose space. The images nearest to the sampled poses are plotted next to the figure. These images show that using only 10 supervised examples it is possible to accurately predict the pose depicted in the remaining unsupervised images.

We continue with quantitative experimental results obtained using a data set of 698 images (64×64 pixels) of a face rendered by computer graphics engine. [3] The face is viewed and lit from different directions and for all images the ground truth viewing direction (longitude and latitude) and lighting direction (longitude) are available. The task is to predict the viewing direction and lighting direction of unsupervised images. The three response variables were normalized to the range $[0, 3.5]$ yielding approximately unit variance.

In the experiments we used all 698 images, some of which were supervised. The supervised images were selected either randomly or by an active learning

[2] The data set is available from `http://www.science.uva.nl/~jverbeek`.

[3] The data set, also used in [2], is available from `http://isomap.stanford.edu`.

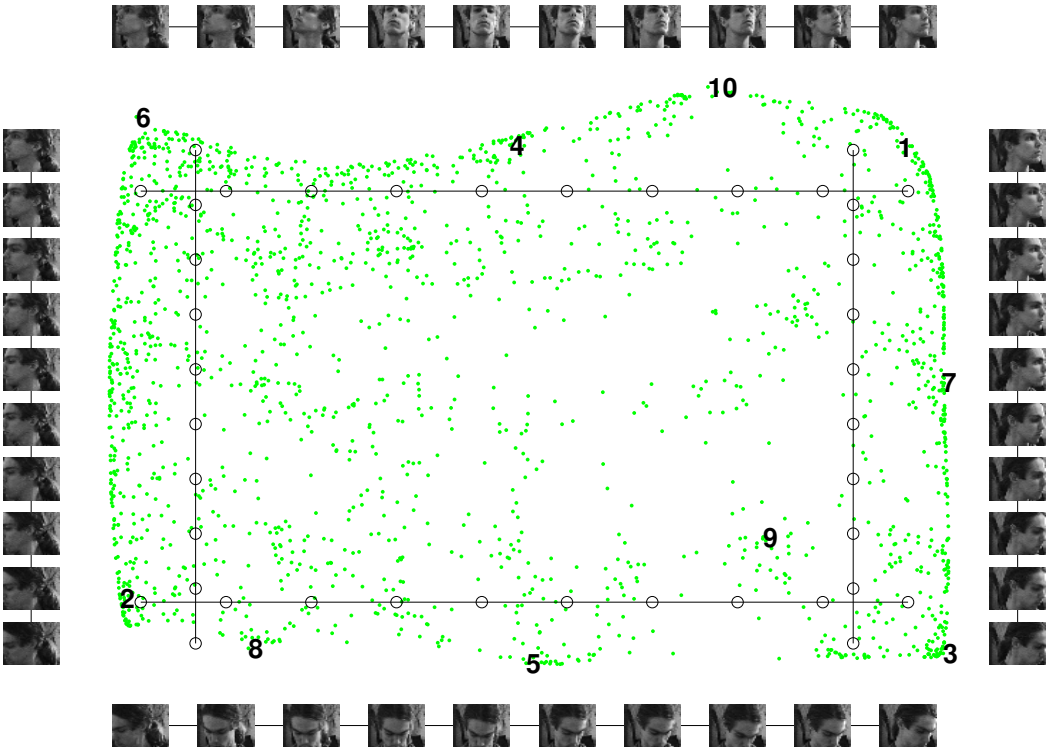
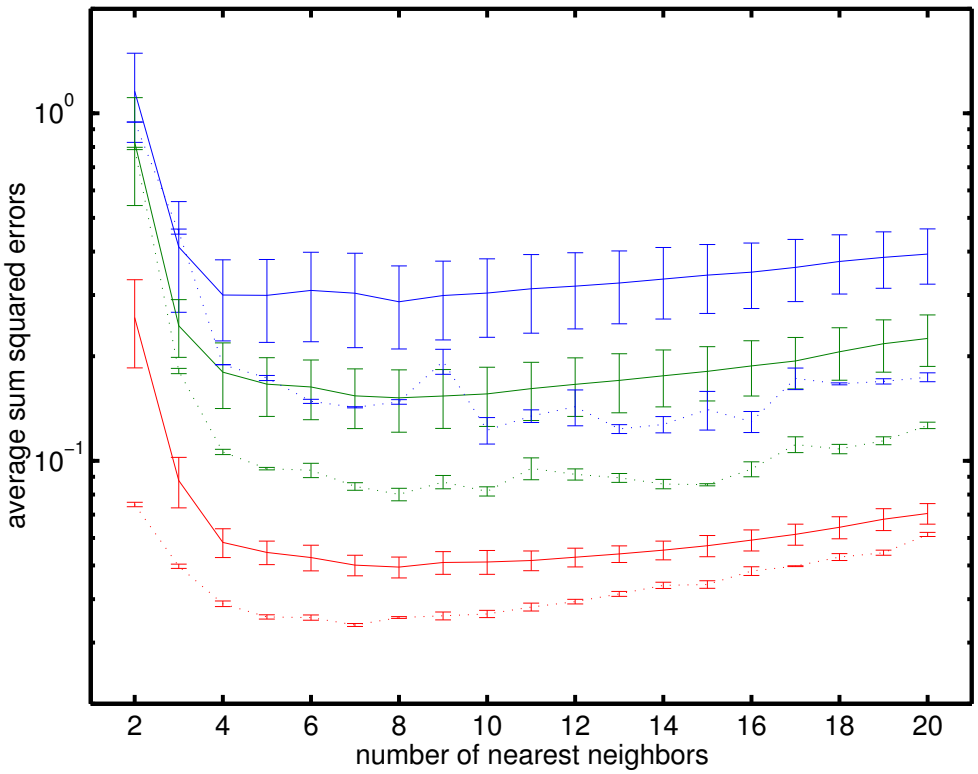


Fig. 2. Predicted pose parameters of 2000 images, together with the pose parameters of the ten supervised images. For 40 other points in the pose space, the image with the most similar predicted pose-parameters is shown.

scheme. The squared errors between the predictions and the ground truth are averaged over the unsupervised points and over the different response variables. All reported results are averages over 20 experiments and we used t-tests to assess the p-value significance levels of the obtained results presented in Fig. 3 (note the log scale).

4.3.1 Query selection

In Fig. 3 we show both for random query selection (solid lines) and for active query selection (dotted lines) the squared errors against the number of neighbors k used in the model. The different lines (from top to bottom) show results for $10, 20$ and 100 labelled points. We started the query selection procedure with the greedy query selection method, i.e. selecting one-by-one the queries that are expected to yield the most information given the other already selected query points. After the greedy initialization we improved the query set by an iterative procedure. We randomly selected one point not in the query set and computed whether the query set entropy increased if this point replaced any of the points in the current query set. We terminated the iterations if no entropy increase was achieved after considering 20 unsupervised points.



Average squared errors

	10	20	100
R O	0.28	0.15	0.05
R S	0.31	0.16	0.05
A O	0.12	0.08	0.03
A S	0.32	0.09	0.03

R/A: random / active queries
O/S: using optimal/selected k

Fig. 3. Errors on unsupervised points against number of neighbors, using 10, 20 and 100 randomly (solid lines) and actively (dotted lines) selected queries (top panel). Some example images (bottom right). Model selection results versus using optimal k value (table).

We ran experiments for neighborhood graphs of different connectivity: $k = 2, 3, \ldots, 20$. In all but 4 of the 57 combinations of k and a number of labelled points the active learning algorithm performs significantly ($p < 0.001$) better than using random queries. Also note that the standard deviation in the performance of the active learning approach is much smaller. We conclude that the active learning algorithm outperformed the usage of random queries.

4.3.2 Model selection

Next, we consider the performance of the maximum likelihood model selection procedure described in Section 4.1. Note that query selection depends on the number of neighbors that are used, and that model selection depends on the supervised queries. To combine model selection with query selection we use a greedy procedure: first a random query is labelled, and then alternatingly k is selected using the labelled points, and a new query is selected depending on the current k and already labelled points. If random queries are selected, we can perform model selection directly using all labelled points.

The table in Fig. 3 shows the average results that were obtained with the optimal value of k in each of the 20 experiments and with the value of k found by model selection. In each of the 20 experiments the optimal value of k was determined separately when combined with active and random queries: it is the number of neighbors for which the smallest prediction error was obtained (using active or random queries respectively).

Using random queries, model selection is near optimal: the optimal k does not give significantly better results than using the automatically selected k ($p = 0.1$). However, for actively selected queries and only 10 supervised points, the results for optimal k are significantly better ($p < 0.02$). The reason is that for few supervised points, model selection picks too small k which causes the query selection procedure to perform poorly. Comparing active and random queries, we found that for 10 queries there is no significant difference between active and random selection if model selection is used ($p = 0.4$). For more queries active selection is significantly better ($p < 10^{-4}$).

4.3.3 Comparison to staged unsupervised and supervised learning

In this experiment we compare the GF approach to the staged learning approach of [6] for semi-supervised classification as discussed in Section 3. The same data set as in the previous experiment was used, and each experiment was repeated 20 times with supervised points chosen uniformly random [4] in each experiment. We computed the prediction error on the remaining unsupervised points for different numbers of supervised points n_s, and numbers of nearest neighbors k to construct the graph. For the staged learning approach we have to set a number of eigenvectors m that will be used. In Tab. 1 we report the results for the *optimal* number of eigenvectors $m \in \{1, \ldots, 20\}$, i.e. the number that led to minimum average error. The results are printed bold if there is a significant difference in performance. The performance difference between

[4] We use random queries to avoid biasing results due to GF-based query selection.

n_s	k	4	6	8	10	12	14	16	18	20
10	*GF*	**5.1**	4.5	5.1	3.7	4.3	4.0	4.1	3.8	3.2
		±2.0	±2.0	±3.0	±1.3	±2.1	±2.2	±2.6	±1.7	±1.5
	SL	11.0 /1	6.3 /4	7.6 /4	3.3 /4	2.9 /4	2.8 /4	3.3 /4	4.5 /4	4.4 /4
		±1.3	±4.0	±7.5	±1.8	±1.3	±1.1	±2.4	±5.5	±10.1
20	*GF*	2.7	**2.2**	2.1	1.9	2.0	1.8	1.8	1.8	1.8
		±0.9	±0.5	±0.6	±0.5	±0.8	±0.5	±0.6	±0.5	±0.4
	SL	3.3 /8	3.1 /7	1.8 /8	1.6 /7	1.5 /7	1.7 /7	1.8 /7	1.8 /8	1.9 /7
		±3.6	±1.2	±0.9	±0.6	±0.3	±1.1	±0.9	±0.5	±0.6
100	*GF*	**1.0**	**0.8**	**0.8**	**0.8**	**0.7**	**0.7**	**0.7**	**0.8**	**0.8**
		±0.1	±0.1	±0.1	±0.1	±0.1	±0.1	±0.1	±0.1	±0.1
	SL	1.2 /16	1.0 /17	1.0 /16	0.9 /8	0.9 /7	0.9 /11	0.9 /11	1.0 /8	1.0 /9
		±0.2	±0.1	±0.1	±0.1	±0.0	±0.1	±0.1	±0.0	±0.1

Table 1
Prediction errors ×10 (average and st.dev.) obtained using GF and staged learning (SL). The optimal number of eigenvectors for SL is also given for each setting.

the two methods is mostly not significant ($p < 0.01$) for $n_s \in \{10, 20\}$. For $n_s = 100$ supervised points the GF method performs significantly ($p < 0.01$) better. Note that the GF method does not require the choice for a particular number of eigenvectors, and thus has one parameter less.

5 Correspondence learning

The second learning problem we consider is learning correspondences between two sets of data points. Each data set is assumed to be sampled from a different high-dimensional embedding of the same underlying low-dimensional manifold. The learning examples consist of (i) several supervised pairs of points (one from each data set) that are in correspondence, i.e. have the same (unknown) coordinate on the underlying manifold and (ii) many unsupervised points in each set for which the corresponding point in the other set is unknown. The goal is to predict other correspondences, i.e. for unsupervised points we want to predict which point in the other set corresponds to it.

5.1 Correspondence learning with the LLE algorithm

The approach to correspondence learning taken in [7] is to perform dimension reduction for both sets simultaneously with the LLE algorithm, explicitly

aiming at discovering the underlying low-dimensional coordinates. The energy function adds the LLE energy functions (3) of each set. As we explain next, the correspondences are accounted for by constraining the low-dimensional coordinates of corresponding points to coincide.

Without loss of generality we order the points in both sets in such a manner that the corresponding points have equal indices in each set, and all points with a correspondence have a smaller index than points without a correspondence. Let us use a and b to index the two data sets. Let $\mathbf{L}^a$ and $\mathbf{L}^b$ be the Laplacians, c.f. (1)–(3), of the separate sets as used by LLE. We use the partitions

$$\mathbf{L}^a = \begin{pmatrix} \mathbf{L}^a_{ss} & \mathbf{L}^a_{su} \\ \mathbf{L}^a_{us} & \mathbf{L}^a_{uu} \end{pmatrix}, \qquad \mathbf{y}^a = \begin{pmatrix} \mathbf{y}^a_s \\ \mathbf{y}^a_u \end{pmatrix}, \tag{16}$$

and similar partitions for $\mathbf{L}^b$ and $\mathbf{y}^b$. The two Laplacians are combined to form

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}^a_{ss} + \mathbf{L}^b_{ss} & \mathbf{L}^a_{su} & \mathbf{L}^b_{su} \\ \mathbf{L}^a_{us} & \mathbf{L}^a_{uu} & 0 \\ \mathbf{L}^b_{us} & 0 & \mathbf{L}^b_{uu} \end{pmatrix}. \tag{17}$$

Taking into account the constraints $\mathbf{y}^a_s = \mathbf{y}^b_s \triangleq \mathbf{y}_s$, and letting $\mathbf{y} = [\mathbf{y}_s^\top \mathbf{y}_u^{a\top} \mathbf{y}_u^{b\top}]^\top$, the sum of the energy functions, c.f. c.f. (1)–(3), of the two sets equals:

$$E(\mathbf{y}) = \mathbf{y}^\top \mathbf{L} \mathbf{y}. \tag{18}$$

Analogous to the normal LLE algorithm, the eigenvectors of $\mathbf{L}$ with smallest eigenvalues give the minimum energy assignments of $\mathbf{y}^a_u, \mathbf{y}^b_u$, and $\mathbf{y}_s$. The low-dimensional coordinates of each point are in the corresponding row of the matrix of which the columns are the smallest eigenvectors of $\mathbf{L}$. To predict a correspondence of an unsupervised point in set a, the nearest (using Euclidean distance in the low-dimensional representation) point from set b is found. Note that the dimensionality of the low-dimensional space, as well as relative scaling of the dimensions, influences the obtained results. In some cases an appropriate dimensionality for the embedding is known. In other cases, however, this dimensionality has to be estimated. Furthermore, in the above approach an appropriate number of nearest neighbors has to be determined for each set.

The semi-parametric correspondence learning setup in [12] differs in that it delivers a mixture of factor analyzers from which the high-dimensional corresponding observations can be reconstructed without storing the training data. However, in the semi-parametric approach some trade-off has to be made between satisfying the correspondences (ensuring that correspondences have

similar low dimensional coordinates) and mutual alignment of the factor analyzers fitted to each set of observations (ensuring that each factor analyzer assigns approximately the same low-dimensional coordinates to data points).

5.2 Correspondence learning with Gaussian fields

By casting the correspondence learning in the GF framework, we do not need an explicit representation in a low-dimensional space. Essentially, we compute quantities that are averaged over possible representations based on their likelihood under the GF distribution. As a first step we define a GF for each point set separately as in the previous sections, with inverse covariance matrices $\beta\mathbf{M}^a$ and $\beta\mathbf{M}^b$. We may view this as a single GF with two independent subfields. To account for correspondences, we constrain variables that are linked by the correspondences to have identical (but unknown) values. The constraints induce correlations between other variables in the subfields. The inverse covariance matrix $\beta\mathbf{M}$ is obtained in the same way as $\mathbf{L}$ above, where the two variables of each correspondence pair have been replaced by a single variable. Again we use $\mathbf{C}$ to denote the inverse of $\mathbf{M}$ of the combined field. The expected squared difference $(y_i - y_j)^2$ under the constrained GF distribution is:

$$\mathrm{E}[(y_i - y_j)^2] = \mathrm{E}[y_i^2] + \mathrm{E}[y_j^2] - 2\mathrm{E}[y_i y_j] = \beta^{-1}(\mathbf{C}_{ii} + \mathbf{C}_{jj} - 2\mathbf{C}_{ij}). \qquad (19)$$

Given a query point i from one set we can then find the point j in the other set with smallest expected squared difference (note that this point j is invariant for the value of β).

The LLE based method could be viewed as an approximation to our GF based method in the following sense. Let $\mathbf{v}_t$ and λ_t denote eigenvectors and the corresponding eigenvalues of $\mathbf{C}$, then we can write

$$\mathbf{C}_{ij} = \sum_t \mathbf{v}_t(i)\mathbf{v}_t(j)\lambda_t. \qquad (20)$$

Let the eigenvalues be ordered from large to small: $\lambda_t \geq \lambda_{t+1}$, the LLE based method [7] is obtained (for embedding dimensionality d) as follows. We set $\lambda_t \leftarrow 1$ for $t \in \{1, \ldots, d+1\}$ and $\lambda_t \leftarrow 0$ for $t > d+1$. The squared distance in the eigenvector embedding used by the LLE based method equals $\mathbf{C}_{ii} + \mathbf{C}_{jj} - 2\mathbf{C}_{ij}$, where the $\mathbf{C}_{ij}$ are computed as in (20) with the replaced λ_t.

5.2.1 Model selection

In the correspondence problem we can use a marginal likelihood model selection procedure (selecting an appropriate number of neighbors k and variance scale parameter β) similar to the procedure used in the previous section. However, here we consider the high-dimensional input vectors as the variables that we want to predict. For the n_s given correspondences both high-dimensional input vectors belonging to sets a and b are known.

The model selection method of the previous section is based on maximizing the marginal log-likelihood of the supervised data. Suppose that the input vectors from set a and b have dimensionality respectively d_a and d_b. Let $d = d_a + d_b$, then we define the $n_s \times d$ matrix $\mathbf{Z}$ as the matrix having in the i-th row the concatenation of the input vectors of the i-th correspondence pair. In the current setting, model selection will be based on maximizing the marginal log-likelihood of $\mathbf{Z}$. Assuming that the columns $\mathbf{z}_j$ of $\mathbf{Z}$ are independently and identically distributed [5] according to the GF distribution, this translates into maximizing the sum of the marginal log-likelihoods of the columns of $\mathbf{Z}$. Thus model selection is based on finding k and β that maximize:

$$p(\mathbf{Z}|k,\beta) = \sum_{j=1}^{d} \log p(\mathbf{z}_j|k,\beta) \tag{21}$$

$$= \sum_{j=1}^{d} -\frac{1}{2}\left[\log|\mathbf{C}_{ss}| - n_s \log\beta + \beta \mathbf{z}_j^\top \mathbf{C}_{ss}^{-1}\mathbf{z}_j\right] \tag{22}$$

$$= -\frac{1}{2}\left[d\log|\mathbf{C}_{ss}| - dn_s\log\beta + \beta \mathrm{Tr}\{\mathbf{Z}^\top\mathbf{C}_{ss}^{-1}\mathbf{Z}\}\right]. \tag{23}$$

The maximization with respect to β can again be done in closed form yielding an expressions similar to (13) and (14):

$$\beta^* = \arg\max_{\beta}\ \log p(\mathbf{Z}|\beta,k) = dn_s/\mathrm{Tr}\{\mathbf{Z}^\top\mathbf{C}_{ss}^{-1}\mathbf{Z}\}, \tag{24}$$

$$l^* = \log p(\mathbf{Z}|\beta^*,k) \tag{25}$$

$$= -\frac{1}{2}\left[d\log|\mathbf{C}_{ss}| + dn_s + dn_s\log\left(\mathrm{Tr}\{\mathbf{Z}^\top\mathbf{C}_{ss}^{-1}\mathbf{Z}\}/(dn_s)\right)\right]. \tag{26}$$

To find the value of k for which l^* is maximal (and the corresponding β^*) we evaluate (26) for different values of k. To do this we need to find $\mathbf{C}_{ss}$, which

[5] This assumption allows for straightforward generalization of the model selection method used in the previous section. Dropping this assumption requires non-trivial specification of conditional fields over columns of $\mathbf{Z}$ given other columns.

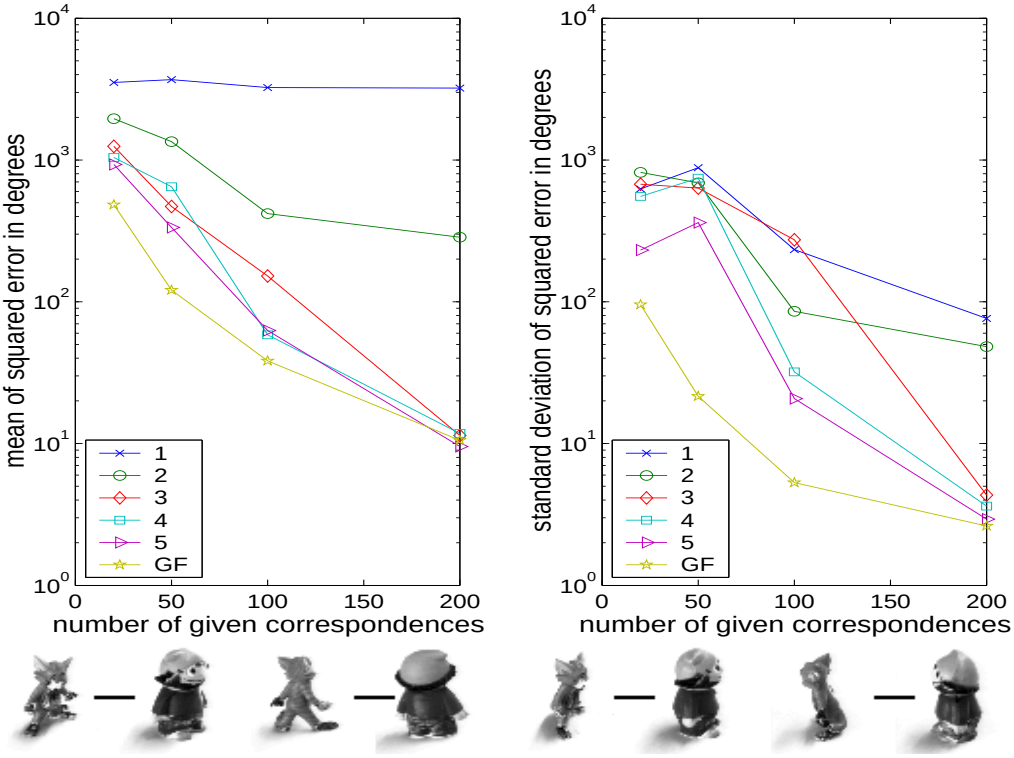


Fig. 4. Mean (top left) and standard deviations (top right) of error against number of correspondences. Results are shown for the LLE based method using 1 up to 5 dimensions and the GF method. Three examples of corresponding images connected by bars (bottom).

can be computed with methods discussed in the previous section. Note that to compute the trace in (26) we can use $\mathrm{Tr}\{\mathbf{Z}^\top \mathbf{C}_{ss}^{-1}\mathbf{Z}\} = \sum_{j=1}^{d} \mathbf{z}_j^\top \mathbf{C}_{ss}^{-1}\mathbf{z}_j$. Thus the required amount of computation depends only linearly on d.

5.3 Experimental results

We compare the performance of our GF approach with the LLE approach proposed in [7] using representations of different dimensionality. The data sets we used each contain 2500 views of a toy puppet recorded by positioning a camera on different positions on the hemisphere centered at the object. [6] For each view of one object there is a corresponding view of the other object recorded from the same camera position, see the images in Fig. 4. Given several corresponding views, the task is to predict for each remaining image in set a which is the corresponding image in set b and vice-versa. We compare the position of the camera of the true correspondence and the predicted correspondence.

[6] The data is available from `http://ls7-www.cs.uni-dortmund.de/~peters`.

The error we measure is the summed squared error in the longitude of camera ($0° - 360°$) and latitude ($0° - 90°$). The neighborhood graph (using $k = 20$ neighbors) was based on Euclidean distances when between the 64×64 pixel images (considering each pixel as a coordinate in the image space).

In Fig. 4 the mean and standard deviation of the errors are shown of ten experiments using different random sets of $20, 50, 100$ and 200 correspondences. First, we can see that for dimensionality smaller than three the LLE based method performs poorly; the differences with GF are significant with $p < 0.01$. The reason is that of the two degrees of freedom in the data, one (the longitude of the camera) is periodic. Therefore an additional dimension is required for a suitable representation. Second, the GF approach leads to equal or smaller errors and has less variance in the errors than the LLE based method using three or more dimensions and less than 100 correspondences (7 of the 9 differences are significant with $p < 0.05$). We conclude that the GF method (i) removes the need to find the optimal representation dimensionality and (ii) performs as good as or better than the LLE based method.

In Fig. 5 we show results for model selection, averaged over ten experiments, using different randomly selected sets of 50 correspondences. In each experiment we computed for $k = 2, 4, 6, \ldots, 20$ neighbors the marginal log-likelihood l^* of the 2×64^2 observed pixel values of the correspondences. In each experiment, we predicted the correspondences with the LLE based method and the GF method. The average prediction error and marginal log-likelihood are plotted against the number of neighbors in Fig. 5. Note that marginal likelihood gives a good indication of the prediction error, and that —on average— for $k = 8$ the highest marginal log-likelihood is obtained and both methods give the smallest prediction errors. Another interesting quantity is the average prediction error when using in each experiment the value of k for which the maximum marginal log-likelihood was achieved in that experiment. This is the error that would be obtained if we used the maximum likelihood criterion for model selection. The average of this error is given by the horizontal lines in the left panel of Fig. 5. In five experiments $k = 8$ was selected, three times $k = 10$ was selected and in two cases $k = 12$ was selected. For both the LLE-based and the GF approach, model selection leads to prediction errors near the error obtained using the optimal value of k.

6 Conclusion and discussion

In this paper we have shown how sparse GFs based on a neighborhood graph can be used for semi-supervised regression and correspondence learning. For semi-supervised regression we have shown that (i) a number of neighbors that

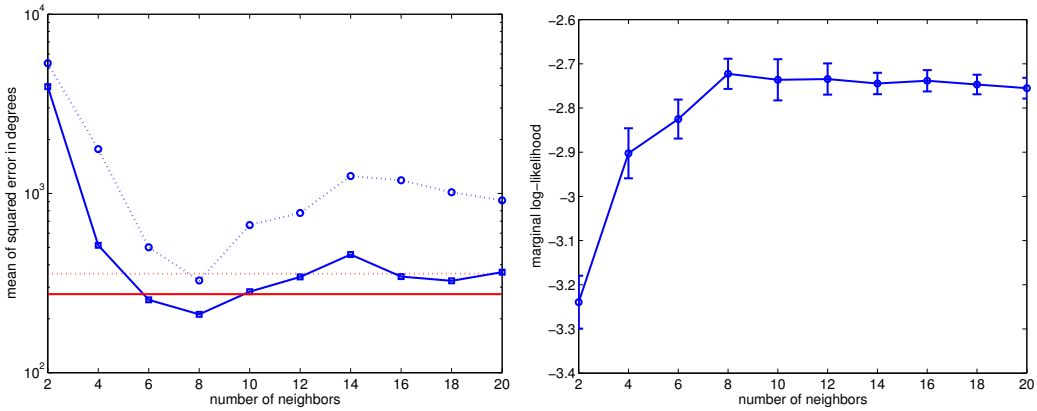


Fig. 5. Mean error against the number of neighbors, for LLE based method (dotted) and GF (solid). Horizontal lines give the average obtained result if model selection is used (left panel). Mean and standard deviations of marginal log-likelihood against the number of neighbors (right panel).

yields good generalization can effectively be estimated using a maximum-likelihood criterion (ii) smaller errors can be obtained with an equal number of labelled examples if an entropy minimization principle is used to select which data points to label (iii) comparable or smaller prediction errors are obtained as compared to a staged unsupervised-supervised learning approach. For correspondence learning we have shown that our GF approach, for which we do not have to fix a representation dimensionality, performs comparable or better than the LLE based method. Furthermore, also for correspondence learning a number of neighbors that gives accurate predictions can be determined through a maximum likelihood criterion.

An alternative to using Cholesky decompositions, is to use iterative algorithms, e.g. Lanczos iterations, to solve the linear systems as required for model selection and query selection. Such methods are attractive in active learning settings since the solution after labelling a new point will be similar to the solution before labelling the point. Therefore, accurate solutions are expected to be found in relatively few iterations if the iterations are started with the (approximate) solution before the last labelling.

The Gaussian field (GF) models considered here resemble Gaussian Processes (GP) [13]. A GP is given by a mean function $m(\mathbf{x})$ and a covariance function $C(\mathbf{x}, \mathbf{x}')$ which together define a Gaussian density over outputs for some input domain $\mathcal{X}$. The marginal Gaussian on the outputs of $\{\mathbf{x}_i \in \mathcal{X}\}_{i=1}^n$ has mean vector $[m(\mathbf{x}_1) \dots m(\mathbf{x}_n)]^\top$ and the $(i,j)^{th}$ entry of the covariance matrix is given by $C(\mathbf{x}_i, \mathbf{x}_j)$. Thus, the data are used to specify the covariance matrix rather than the inverse covariance function as in GFs. Therefore, in a GP the marginal on the outputs of several data points does not depend on other unsupervised data points. In contrast, in a GF the marginal distribution on the outputs of several data points does depend on other unsupervised points,

since the covariance matrix of the marginal is given by the corresponding block of the *inverse* of $\mathbf{M}$ and thus also depends on the parts of $\mathbf{M}$ corresponding to other points. In this manner unsupervised data modifies the covariance matrix to account for the manifold structure derived from the unsupervised data.

Acknowledgments JJV is supported by the Technology Foundation STW (AIF4997) applied science division of NWO and the technology program of the Dutch Ministry of Economic Affairs.

References

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

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

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[4] T. Joachims. Transductive learning via spectral graph partitioning. In T. Fawcett and N. Mishra, editors, *Proceedings of the International Conference on Machine Learning*, volume 20, pages 290–297. AAAI Press, 2003.

[5] X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, volume 20, pages 912–919. AAAI Press, Menlo Park, CA, USA, 2003.

[6] M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56(1–3):209–239, 2004.

[7] J. H. Ham, D. D. Lee, and L. K. Saul. Semisupervised alignment of manifolds. In Z. Ghahramani and R. Cowell, editors, *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, volume 10, pages 120–127, 2005.

[8] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 741–750. ACM Press, New-York, NY, USA, 2002.

[9] D. J. C. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1991.

[10] R. Hwa. Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 45–52, 2000.

[11] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the International Conference on Machine Learning*, volume 17, pages 911–918. Morgan Kaufmann, San Mateo, CA, USA, 2000.

[12] J. J. Verbeek, S. T. Roweis, and N. Vlassis. Non-linear CCA and PCA by alignment of local models. In S. Thrun and L. K. Saul and B. Schölkopf, editor, *Advances in Neural Information Processing Systems*, volume 16, pages 297–304. MIT Press, Cambridge, MA, USA, 2004.

[13] R.M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.