



UNIVERSITEIT
VAN
AMSTERDAM

Submitted to: Pattern Recognition

IAS technical report IAS-UVA-05-01

Semi-supervised learning with Gaussian fields

J.J. Verbeek and N. Vlassis

Intelligent Autonomous Systems,
University of Amsterdam
The Netherlands

Gaussian fields (GF) have recently received considerable attention for dimension reduction and semi-supervised classification. This paper presents two contributions. First, we show how the GF framework can be used for regression tasks on high-dimensional data. We consider an active learning strategy based on entropy minimization and a maximum likelihood model selection method. Second, we show how a recent generalization of the Locally Linear Embedding algorithm for correspondence learning can also be cast into the GF framework, which obviates the need to choose a representation dimensionality.

Keywords: Gaussian fields, regression, correspondence learning, active learning, model selection.

IAS

intelligent autonomous systems

Contents

1	Introduction	1
2	Gaussian fields for unsupervised and semi-supervised learning	2
3	Active semi-supervised regression with Gaussian fields	4
3.1	Model selection	4
3.2	Query selection	6
3.3	Experimental results	7
3.3.1	Query selection.	7
3.3.2	Model selection.	10
4	Correspondence learning	10
4.1	Correspondence learning with the LLE algorithm	10
4.2	Correspondence learning with Gaussian fields	11
4.2.1	Model selection	12
4.3	Experimental results	13
5	Conclusion and discussion	14

Intelligent Autonomous Systems
 Informatics Institute, Faculty of Science
 University of Amsterdam
 Kruislaan 403, 1098 SJ Amsterdam
 The Netherlands

Tel (fax): +31 20 525 7461 (7490)
<http://www.science.uva.nl/research/ias/>

Corresponding author:

J.J. Verbeek
 tel: +31 20 525 7550
jverbeek@science.uva.nl
<http://www.science.uva.nl/~jverbeek/>

1 Introduction

Regression on the basis of high-dimensional data is a challenging problem in machine learning. In order to successfully learn regression functions either a limited class of regression models must be used (e.g. only linear models) or large quantities of data are required to reliably find a good model from a larger class. Recently several techniques have been proposed 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 is used to specify the assumption that nearest neighbors (according to some similarity measure) in the high-dimensional input space will have similar ‘outputs’. In case of dimension reduction [12, 15, 7] the outputs are the desired low-dimensional representations of the inputs, in the case of semi-supervised classification [5, 17, 1] the outputs are class labels.

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 all outputs. The GF density associated with a quadratic energy function is proportional to the exponent of the negative energy function (therefore it follows that the density is a Gaussian). The methods mentioned above return a vector of outputs that minimizes the energy function; this is the mean of the corresponding GF density.

In this paper we consider GFs defined on nearest neighbor graphs for two different 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 the pixel values of images of a face that looks in different directions and we want to map these images to pose parameters. If we consider the raw images as input vectors then the input equals the number of pixels in the images, but 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 (for which we do not know the pose parameters) can be exploited to identify the low-dimensional manifold structure of the data. Due to the low-dimensionality of the manifold, only few supervised examples are needed to successfully learn a regression function to predict the pose parameters. The amount of supervised data that can be acquired in practice is often limited. This motivates active learning approaches, which determine on the basis of a set of unsupervised data (and possibly some supervised data), which unsupervised data are expected to yield the most information if the supervised signal would be available for them.

The second problem is correspondence learning. In this problem two high-dimensional data sets are given that share the same underlying modes of variability. For example, each set can be a collection of images of different views of a particular object. The manifolds on which the two data sets lie, parameterized by the shared modes of variability, 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 now 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 from unsupervised data, which can then be aligned using only relatively few correspondences. Moreover, as opposed to earlier work on correspondence learning [3, 16], in the GF framework we do not need to specify a dimensionality of the underlying manifold. This is advantageous since the dimensionality is not always known for the problem at hand.

The main difference between the GF approach and more traditional approaches lies in how we specify a density over the outputs. For GFs, this density model is determined by the nearest neighbor graph on the inputs; in this manner the density is derived from —and reflects the

density of— the given inputs. More traditional techniques also define a density on the outputs, although often implicitly. They do this by limiting attention to a parametric class of functions (e.g. linear functions or feed-forward neural networks) that map inputs to outputs. The choice for a specific class of functions is often motivated by reasons of computational efficiency rather than by the suitability of this class for the given problem.

In the next section we introduce GFs defined on nearest neighbor graphs. In section Section 3 we focus on regression. First, we consider a maximum likelihood method to select an appropriate number of nearest neighbors to specify the neighborhood graph. Second, we consider an entropy minimization based query selection procedure for active learning. We present experimental results that show the effectiveness of both procedures. In Section 4 we turn to correspondence learning using GFs. We propose a new approach for the correspondence learning based on GFs, and compare it to an existing approach based on the LLE algorithm. We emphasize an attractive property of GFs for correspondence learning, namely that correspondences can be predicted without an explicit low-dimensional representation as required by the LLE based approach. We conclude with a discussion in Section 5.

2 Gaussian fields for unsupervised and semi-supervised learning

In this section we briefly summarize how Gaussian Fields have been used before for dimension reduction and semi-supervised classification. 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 nearest neighbors of each \mathbf{x}_i . Throughout this we define nearest neighbors on the basis of Euclidean distance. We note that it is not necessary to naively compute all pairwise squared distances to find the nearest neighbors, see e.g. [6].

Let y_i denote the scalar ‘output’ of \mathbf{x}_i , and let \mathbf{y} denote the n -dimensional vector with all outputs. Given the nearest neighbors, we define an energy function $E(\mathbf{y})$ over the outputs 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:

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

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 $\forall_{\mathbf{y}} E(\mathbf{y}) \geq 0$, however the converse is not true: matrices \mathbf{A} with negative entries exist with $\forall_{\mathbf{y}} E(\mathbf{y}) \geq 0$. For example, the Locally Linear Embedding (LLE) algorithm [12] uses

$$\mathbf{A} = \mathbf{W} + \mathbf{W}^\top - \mathbf{W}^\top \mathbf{W}, \quad (2)$$

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

$$E(\mathbf{y}) = \sum_{i=1}^n (y_i - \sum_{j=1}^n w_{ij} y_j)^2 = \|(\mathbf{I} - \mathbf{W}) \mathbf{y}\|_F^2 \geq 0, \quad (3)$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix (i.e. the square root of the sum of the square of all elements).

The LLE algorithm and the Laplacian Eigenmap algorithm [7] (the latter uses an adjacency matrix with non-negative entries) perform dimension reduction by solving for \mathbf{y} that minimizes $E(\mathbf{y})$. Note that there is a trivial solution $\mathbf{y} = \mathbf{1}c$, 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 \mathbf{y} is zero mean and has 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 under the constraint that \mathbf{y}_{d+1} should be orthogonal to the already extracted eigenvectors \mathbf{y}_i ($i = 1, \dots, d$). The resulting minimizing vectors \mathbf{y}_i are all eigenvectors of \mathbf{L} , the eigenvalues corresponding to later added features exceeding those corresponding to the already extracted features.

For semi-supervised classification, see e.g. [5, 17, 1], the value of some outputs y_i (i.e. the class labels, being 0 or 1 in the two class problems) of some data points \mathbf{x}_i are known and thus fixed to these values. The energy $E(\mathbf{y})$ is then minimized with respect to the remaining entries of \mathbf{y} . 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 points and \mathbf{y}_s the sub-vector for the supervised points. Let \mathbf{L}_{ss} , \mathbf{L}_{us} and \mathbf{L}_{uu} denote the corresponding blocks of \mathbf{L} :

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{ss} & \mathbf{L}_{su} \\ \mathbf{L}_{us} & \mathbf{L}_{uu} \end{pmatrix}. \quad (4)$$

Throughout we will use this subscripting to refer to blocks of matrices and vectors that correspond to supervised and unsupervised points. The energy can be expanded as:

$$E(\mathbf{y}) = \mathbf{y}_u^\top \mathbf{L}_{uu} \mathbf{y}_u + \mathbf{y}_s^\top \mathbf{L}_{ss} \mathbf{y}_s + 2\mathbf{y}_u^\top \mathbf{L}_{us} \mathbf{y}_s, \quad (5)$$

and for fixed \mathbf{y}_s , the minimizer \mathbf{y}_u^* of $E(\mathbf{y})$ with respect to \mathbf{y}_u , is characterized by the linear equation:

$$\mathbf{L}_{uu} \mathbf{y}_u^* = -\mathbf{L}_{us} \mathbf{y}_s. \quad (6)$$

Although \mathbf{L}_{uu} is sparse, its inverse may be full and for large n computing and even storing the inverse can be problematic.

Some authors [5, 1] constrain \mathbf{y} 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 coefficients in \mathbf{w} are set to minimize the squared error $\|\mathbf{y}_s - \mathbf{V}_s \mathbf{w}\|^2$, where \mathbf{V}_s contains the rows of \mathbf{V} corresponding to the supervised points. We note here that an alternative (and exact) method to solve (6), that avoids explicit inversion and that is in practice often much faster than computing a small eigenbasis, is to use the Cholesky decomposition $\mathbf{L}_{uu} = \mathbf{R}^\top \mathbf{R}$, then solve for $\mathbf{R}^\top \mathbf{z} = -\mathbf{L}_{us} \mathbf{y}_s$, 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.

In practice, the energy function $E(\mathbf{y})$ cannot be directly used to specify a GF since \mathbf{L} is singular (the constant vector is an eigenvector with eigenvalue zero). By adding a regularizer we ensure that it is positive definite and can be used as an inverse covariance matrix of the GF. Let $\mathbf{M} = \mathbf{L} + \alpha \mathbf{I}$, then it is easy to see that \mathbf{M} has the same eigenvectors as \mathbf{L} , with the eigenvalues increased by α . The regularizer replaces the infinite variance in the direction $\mathbf{1}$ from the GF distribution with variance α^{-1} .

For the given points \mathbf{X} we define the GF density over outputs \mathbf{y} as:

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; 0, \beta^{-1} \mathbf{C}) \propto \exp\left(-\frac{\beta}{2} \mathbf{y}^\top \mathbf{M} \mathbf{y}\right), \quad (7)$$

where β is a scale parameter that controls the smoothness of the density and \mathbf{C} denotes the inverse of \mathbf{M} . Note that in the GF formulation, fixing some of the outputs to \mathbf{y}_s induces a conditional density $p(\mathbf{y}_u | \mathbf{y}_s)$. The mean \mathbf{y}_u^* of this conditional density is given by (6) if we

replace the blocks of \mathbf{L} with the corresponding blocks of \mathbf{M} . Thus the methods considered above can be interpreted as computing the mean of the corresponding Gaussian field conditioned on the supervised points. In the next section we will exploit several other quantities associated with the GF density.

3 Active semi-supervised regression with Gaussian fields

Previous work [12, 7, 5, 17, 1] has considered the use of GFs for semi-supervised classification and dimension reduction; here we consider GFs for semi-supervised regression. The GF density $p(\mathbf{y})$ defined in the previous section can be regarded as a prior density over outputs that favors the outputs of neighboring inputs to be similar. Given the values of some of the variables, collected in \mathbf{y}_s , we obtain the posterior on the remaining variables $p(\mathbf{y}_u|\mathbf{y}_s)$ (in the previous section we already discussed how to compute the mean of the posterior). It is straightforward to generalize this setup to predict several response variables independently using the same Gaussian field; for clarity we focus here on the case with a single response variable.

In work by others on semi-supervised classification, the adjacency matrix \mathbf{A} that determines the energy function was specified directly by putting non-negative entries for nearest neighbors, or by using Gaussian kernels. 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 average squared difference 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. When \mathbf{A} is defined in the LLE-based manner (2) extrapolation is obtained: the data points towards the end of the spiral are assigned higher values to ensure that the average at the last supervised point matches the supervised value as good as possible. 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).

3.1 Model selection

The GF density (7) 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

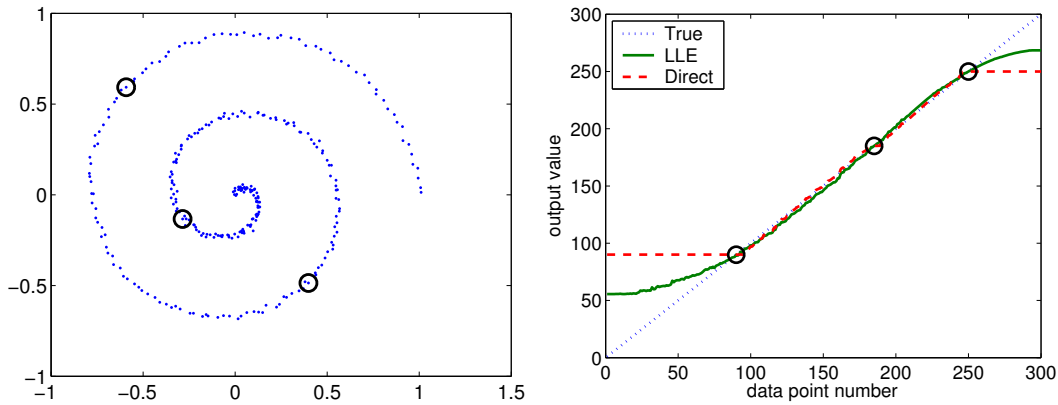


Figure 1: Input data distributed along spiral (left panel). True and predicted output for the data (right panel). The supervised points are encircled in both panels. Predictions are made using the direct and LLE based method to specify matrix \mathbf{A} .

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 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 , for particular k and β , is given by:

$$\log p(\mathbf{y}_s|\beta, k) = -\frac{1}{2} \left[\log |\mathbf{C}_{ss}| - n_s \log \beta + \beta \mathbf{y}_s^\top \mathbf{C}_{ss}^{-1} \mathbf{y}_s \right], \quad (8)$$

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:

$$\beta^* = \arg \max_{\beta} \log p(\mathbf{y}_s|\beta, k) = n_s / (\mathbf{y}_s^\top \mathbf{C}_{ss}^{-1} \mathbf{y}_s), \quad (9)$$

which can be substituted in (8) to find

$$l^* = \log p(\mathbf{y}_s|\beta^*, k) = -\frac{1}{2} \left[\log |\mathbf{C}_{ss}| + n_s + n_s \log (\mathbf{y}_s^\top \mathbf{C}_{ss}^{-1} \mathbf{y}_s / n_s) \right]. \quad (10)$$

Thus, to find the optimal number of neighbors k^* we can evaluate (10) for $k = \{1, 2, \dots, k_{max}\}$ and pick the value for which (10) is maximum. To evaluate (10), 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:

$$\frac{\partial l^*}{\partial \alpha} = \frac{1}{2} \text{Tr} \{ \mathbf{C}_{ss}^{-1} [\mathbf{C}^2]_{ss} \} - \frac{n_s \mathbf{y}_s^\top \mathbf{C}_{ss}^{-1} [\mathbf{C}^2]_{ss} \mathbf{C}_{ss}^{-1} \mathbf{y}_s}{\mathbf{y}_s^\top \mathbf{C}_{ss}^{-1} \mathbf{y}_s}. \quad (11)$$

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 $\mathbf{R}\mathbf{s}_i = \mathbf{z}$. Thus, to find the elements of $[\mathbf{C}^2]_{ss}$ we only need the Cholesky factor \mathbf{R} and solve the triangular linear systems to find the required \mathbf{s}_i . However, after each gradient step the Cholesky factors will need to be recomputed. This makes optimization over the regularizer computationally demanding. In our experiments we therefore fixed the regularizer at $\alpha = 10^{-11}$.

3.2 Query selection

In a semi-supervised active learning setting a set of unlabelled points is given and the learning system asks the user to label one or more of these points to gain prediction accuracy on the remaining unlabelled points. The question is now: for a given labelled set of points, which data points should be additionally labelled to maximize prediction accuracy?

To determine which query set of fixed size is the most informative, we may consider the entropy $H(\mathbf{y}_u|\mathbf{y}_s)$ 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 [8, 4]. 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 we would like the distribution $p(\mathbf{y}_u|\mathbf{y}_s)$ to be concentrated as tight as possible around its mean. The chain-rule of entropies tells us that $H(\mathbf{y}) = H(\mathbf{y}_s) + H(\mathbf{y}_u|\mathbf{y}_s)$. Thus, since the entropy of the complete field $H(\mathbf{y})$ is fixed, we can equivalently aim for the query set with the largest entropy. Note that to compute the entropy $H(\mathbf{y}_s) = \frac{1}{2} \log |\mathbf{C}_{ss}| - \frac{1}{2} n_s \log \beta + \text{const.}$ we can use the Cholesky factors of \mathbf{M} as above to find \mathbf{C}_{ss} .

To find the single variable with maximum variance we could compute all 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, like in [14], 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. A simple, and in practice very successful, scheme is to start with an initial random subset. Until some stopping criterion is met, randomly some variable y_i not in the current subset 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.

If we have some labelled points (the labels of which are collected in \mathbf{y}_t) and want to find an additional set of queries indexed by s . Thus, the outputs are divided in two disjoint sets: the already labelled \mathbf{y}_t and the sofar unlabelled \mathbf{y}_u and the new queries are a subset of the sofar unlabelled : $s \subset u$. We can proceed as above, by considering the conditional field $p(\mathbf{y}_u|\mathbf{y}_t)$ and compute the required entropies from the Cholesky decomposition of \mathbf{M}_{uu} , rather than from the decomposition of \mathbf{M} . (We can also use this to greedily, one-by-one, select an initial set of queries rather than randomly in the algorithm described above.) However, in an active learning system that computes a new query after the user has answered a previous query, the set of unlabelled points changes after each answered query. This would require to compute after each answered query the Cholesky factors of a different matrix \mathbf{M}_{uu} .¹ However, we can use the Cholesky factors of the whole matrix \mathbf{M} to find a query set which maximally reduces the entropy in the distribution on the unlabelled points, if some points have already been labelled. To see this note

¹Note that also to predict \mathbf{y}_u the Cholesky factor of \mathbf{M}_{uu} has to be computed.

that from the chain-rule of entropies we have that $H(\mathbf{y}_{u \setminus s} | \mathbf{y}_{sUt}) = H(\mathbf{y}) - H(\mathbf{y}_{sUt})$. Thus to find the next query set that maximally reduces the entropy in the remaining unlabelled points we find the query set s that, together with t , has the maximum entropy in the complete field. This can be done by evaluating the entropy $H(\mathbf{y}_{sUt})$ for different new query sets and picking the one that maximizes this entropy.

3.3 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 of a face that looks in different directions, varying both horizontally and vertically (see the images depicted in Fig. 2). The goal is to predict the pose of the face.

A neighborhood graph with 20 neighbors was used that was computed directly on the basis of the squared distance between the 40×40 pixel images. 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 2000 images are shown, the coordinates of the ten supervised images are indicated with the digits $1, \dots, 10$. Observe that the query selection procedure mainly requested supervision for relatively extreme views. In the same figure we also plotted four lines, the images nearest to the circles on the lines are plotted next to the figure. These images show that using only 10 supervised examples a reasonable prediction on the pose depicted in the other images can be made.

We continue with quantitative experimental results obtained using a data set of 698 images (64×64 pixels) of a face rendered by computer graphics software.² 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. We fixed the regularizer α at 10^{-11} .

In the experiments we used all 698 images. Some of the images were supervised and others not; the supervised images were selected either randomly or with by an active learning 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).

3.3.1 Query selection.

In Fig. 3 we show both for random query selection (dotted lines) and for active query selection (solid lines) the average squared errors over ten experiments 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

²The data, in [15] used for unsupervised learning, is available from <http://isomap.stanford.edu>.

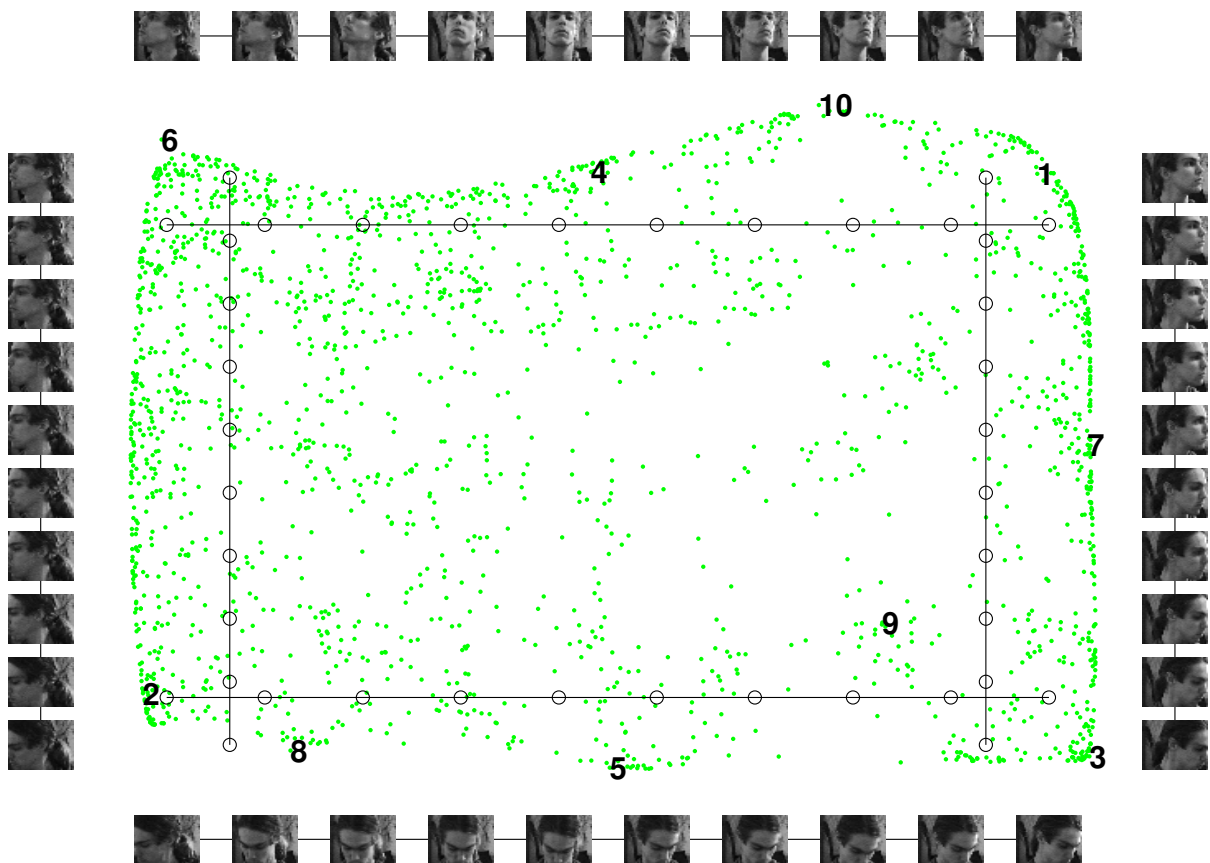
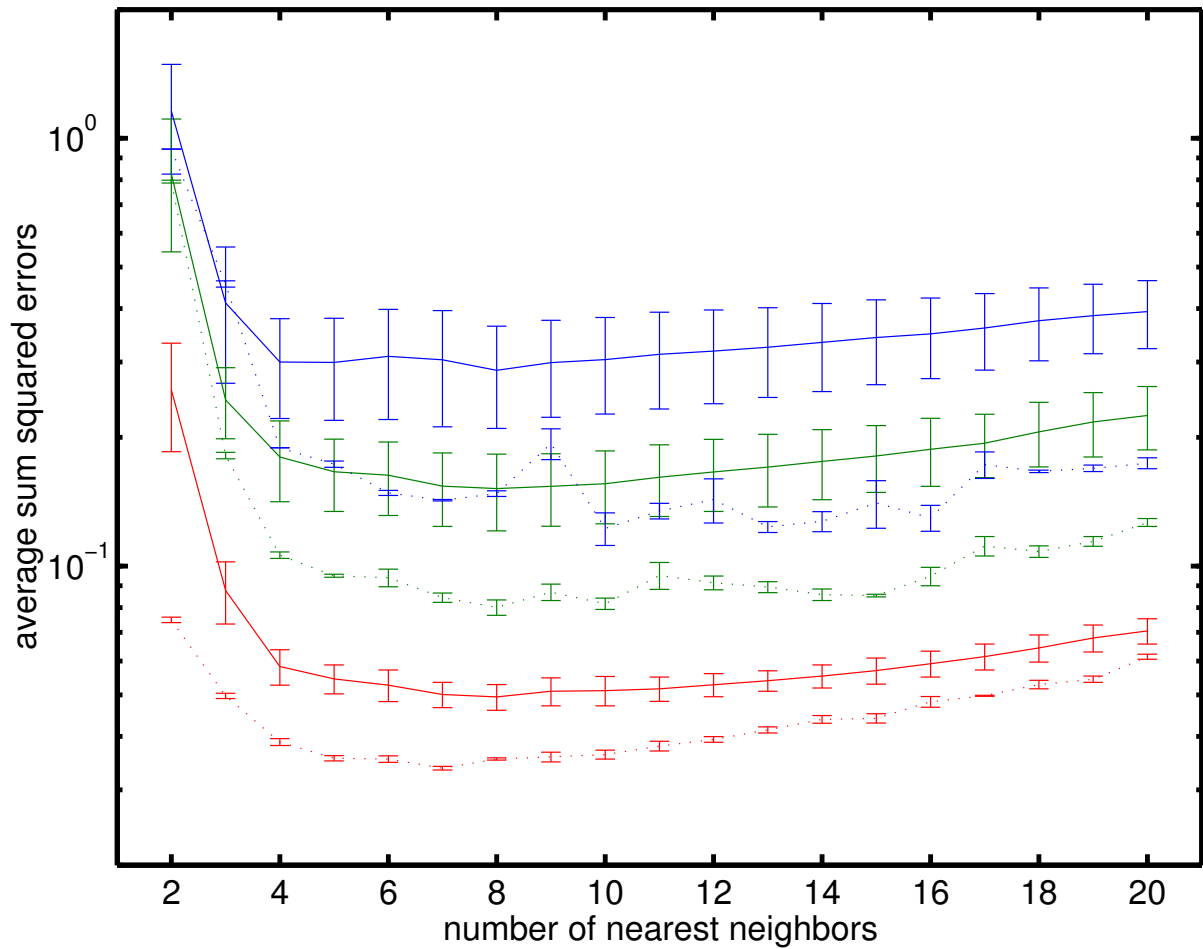


Figure 2: Predicted pose parameters of 2000 images, together with the pose parameters of the 10 supervised images. For 40 other points in the pose space we plotted next to the figure the image with the most similar predicted pose-parameters.



Average squared errors		Number of queries		
		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



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

the current query set. We terminated the iterations if no entropy increase was achieved after considering 20 unsupervised points in this manner.

We ran experiments for neighborhood graphs of different connectivity: $k = 2, 3, \dots, 20$. 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.

3.3.2 Model selection.

Next, we consider the performance of the maximum likelihood model selection procedure described in Section 3.1. 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 alternately k is selected using the labelled points (as described in Section 3.2), and a new query is selected depending on the current k and already labelled points (as described in Section 3.1). If instead a random query set is selected and labelled, we can do 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 experiment and with the value of k found by model selection. In each of the 20 experiments the optimal value of k was determined separately for using active and random queries: it is the number of neighbors for which the smallest prediction error was obtained (using active or random queries respectively).

For random queries using the optimal k does not give significantly better results than using model selection ($p = 0.1$). Thus model selection works well when using random queries. 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 ($p < 0.0001$).

4 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.

4.1 Correspondence learning with the LLE algorithm

The approach to correspondence learning taken in [3] is to perform dimensionality reduction for both sets simultaneously with LLE, explicitly aiming at discovering the underlying low-dimensional coordinates. The energy function adds the separate LLE energy function (3) of each set. The given correspondences are accounted for by constraining the low-dimensional coordinates of corresponding points to coincide, as we explain next.

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 graph Laplacians, c.f. (1) and (3), of the separate sets as used by LLE. We partition \mathbf{L}^a as

$$\mathbf{L}^a = \begin{pmatrix} \mathbf{L}_{ss}^a & \mathbf{L}_{su}^a \\ \mathbf{L}_{us}^a & \mathbf{L}_{uu}^a \end{pmatrix}, \quad (12)$$

and \mathbf{L}^b is partitioned accordingly. The vector \mathbf{y}^a is partitioned into a block for the points with correspondence \mathbf{y}_s^a and a block for the other points \mathbf{y}_u^a , and similarly for \mathbf{y}^b . Let

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{ss}^a + \mathbf{L}_{ss}^b & \mathbf{L}_{su}^a & \mathbf{L}_{su}^b \\ \mathbf{L}_{us}^a & \mathbf{L}_{uu}^a & 0 \\ \mathbf{L}_{us}^b & 0 & \mathbf{L}_{uu}^b \end{pmatrix}. \quad (13)$$

Taking into account the constraints $\mathbf{y}_s^a = \mathbf{y}_s^b \triangleq \mathbf{y}_s$, and letting $\mathbf{y} = [\mathbf{y}_s^\top \mathbf{y}_u^{a\top} \mathbf{y}_u^{b\top}]^\top$, the energy function can be written as:

$$E(\mathbf{y}) = \mathbf{y}^\top \mathbf{L} \mathbf{y}. \quad (14)$$

Analogous to the normal LLE algorithm, the eigenvectors of \mathbf{L} with smallest eigenvalues give the minimum energy assignments of \mathbf{y}_u^a , \mathbf{y}_u^b , 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 [16] 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 (making sure that correspondences have similar low dimensional coordinates) and mutual alignment of the factor analyzers fitted to each set of observations (making sure that each factor analyzer assigns approximately the same low-dimensional coordinates to data points).

4.2 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:

$$\mathbb{E}[(y_i - y_j)^2] = \mathbb{E}[y_i^2] + \mathbb{E}[y_j^2] - 2\mathbb{E}[y_i y_j] = \beta^{-1}(\mathbf{C}_{ii} + \mathbf{C}_{jj} - 2\mathbf{C}_{ij}). \quad (15)$$

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

$$\mathbf{C}_{ij} = \sum_t \mathbf{v}_t(i)\mathbf{v}_t(j)\lambda_t. \quad (16)$$

Let the eigenvalues be ordered from large to small: $\lambda_t \geq \lambda_{t+1}$, the LLE based method [3] is obtained (for embedding dimensionality d) as follows. We set $\lambda_t \leftarrow 1$ for $t \in \{1, \dots, d+1\}$ 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 (16) with the replaced λ_t .

4.2.1 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 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 all columns \mathbf{z}_j of \mathbf{Z} are independently and identically distributed 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:

$$p(\mathbf{Z}|k, \beta) = \sum_{j=1}^d \log p(\mathbf{z}_j|k, \beta) \quad (17)$$

$$= \sum_{j=1}^d -\frac{1}{2} \left[\log |\mathbf{C}_{ss}| - n_s \log \beta + \beta \mathbf{z}_j^\top \mathbf{C}_{ss}^{-1} \mathbf{z}_j \right] \quad (18)$$

$$= -\frac{1}{2} \left[d \log |\mathbf{C}_{ss}| - n_s d \log \beta + \beta \text{Tr}\{\mathbf{Z}^\top \mathbf{C}_{ss}^{-1} \mathbf{Z}\} \right]. \quad (19)$$

The maximization with respect to β can again be done in closed form yielding an expressions similar to (9) and (10):

$$\beta^* = \arg \max_{\beta} \log p(\mathbf{Z}|\beta, k) = n_s d / \text{Tr}\{\mathbf{Z}^\top \mathbf{C}_{ss}^{-1} \mathbf{Z}\}, \quad (20)$$

$$l^* = \log p(\mathbf{Z}|\beta^*, k) \quad (21)$$

$$= -\frac{1}{2} \left[d \log |\mathbf{C}_{ss}| + n_s d + n_s d \log \left(\text{Tr}\{\mathbf{Z}^\top \mathbf{C}_{ss}^{-1} \mathbf{Z}\} / (n_s d) \right) \right]. \quad (22)$$

To find the value of k for which l^* is maximal (and the corresponding β^*) we evaluate (22) for different values of k . To do this we need to find \mathbf{C}_{ss} , which can be computed with methods discussed in the previous section. Note that to compute the trace in (22) we can use $\text{Tr}\{\mathbf{Z}^\top \mathbf{C}_{ss}^{-1} \mathbf{Z}\} = \sum_{j=1}^d \mathbf{z}_j^\top \mathbf{C}_{ss}^{-1} \mathbf{z}_j$. Thus the required amount of computation grows only linearly in the number of input dimensions d .

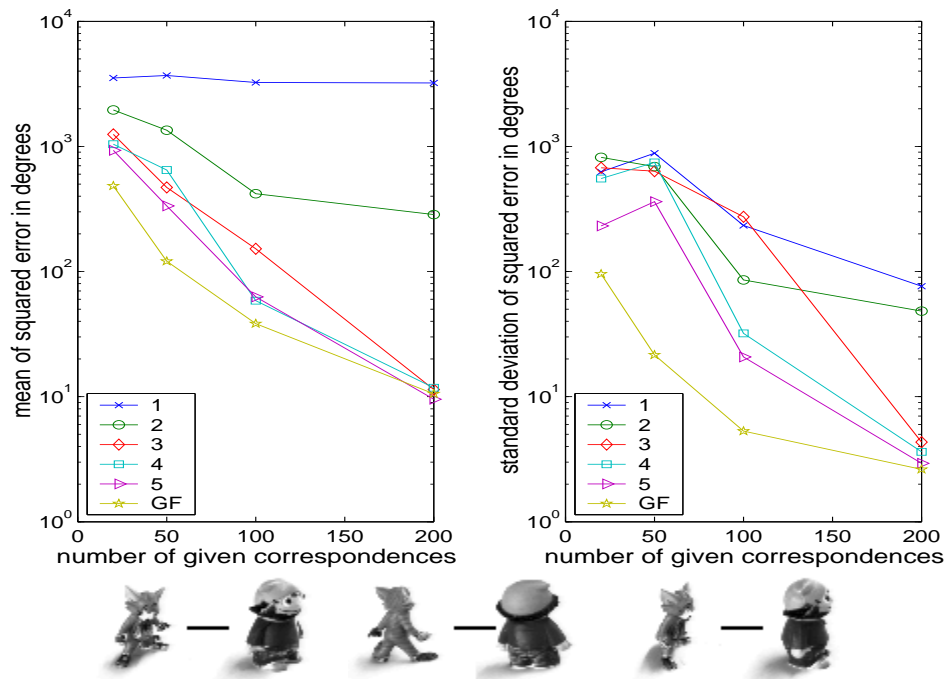


Figure 4: Mean (top left) and standard deviations (top right) of error against number of correspondences. Results are shown for the LLE based method using 1 up to 5 dimensions and the GF method. Three examples of corresponding images connected by bars (bottom).

4.3 Experimental results

We compare the performance of our GF approach with the LLE approach proposed in [3] 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.³ 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 some corresponding views, the task is to predict for each remaining image in set a which is the corresponding image in set b . We compare the position of the camera of the true correspondence and the predicted correspondence. The error we measure is the summed squared error in the longitude of camera ($0^\circ - 360^\circ$) and latitude ($0^\circ - 90^\circ$). 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.

³The data is available through G. Peters (Dept. of Computer Graphics, University Dortmund, Germany), see [11] and <http://ls7-www.cs.uni-dortmund.de/~peters>.

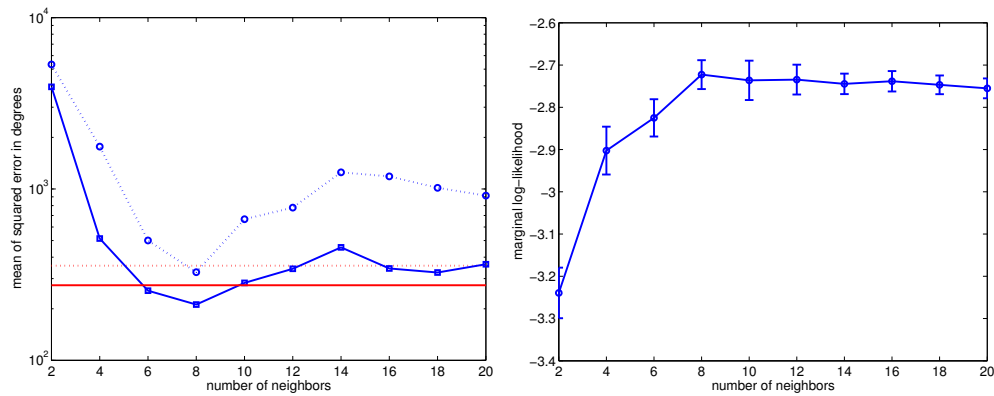


Figure 5: Mean error against the number of neighbors, for LLE based method (dotted) and GF (solid). Horizontal lines give the average obtained result if model selection is used (left panel). Mean and standard deviations of marginal log-likelihood against the number of neighbors (right panel).

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 $k = 2, 4, 6, \dots, 20$ neighbors the marginal log-likelihood l^* of the 2×64^2 observed pixel values of the correspondences. In each experiment, we also 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.

5 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 yields good generalization can effectively be estimated using a maximum-likelihood criterion and (ii) smaller errors can be obtained with an equal number of labeled examples if an entropy minimization principle is used to select which data points to label. 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 interesting alternative to using Cholesky decompositions, is to use iterative algorithms to solve linear systems, e.g. methods based on Lanczos iterations [2], to find the blocks of \mathbf{C} and \mathbf{C}^{-1} 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 can be found in relatively few iterations

if the iterations are started with the (approximate) solution before the last labelling [9].

The Gaussian field (GF) models considered here resemble Gaussian Processes (GP) [10]. 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 the outputs for all points in some input domain. The marginal Gaussian on the outputs of n elements $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ of the input domain can be obtained as follows: the mean vector is given by $[m(\mathbf{x}_1) \dots m(\mathbf{x}_n)]^\top$ 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 the GF considered in this paper. Therefore, in a GP the marginal on the outputs of several data points does not depend on other unsupervised data points. The marginal is simply obtained by removing the entries in the mean and covariance matrix that correspond to the variables over which we want to marginalize. The important difference between GP and the GF considered here is that for the GF the marginal distribution on the outputs of several data points does depend on other unsupervised points (for a fixed matrix \mathbf{M}). The covariance matrix of the marginal is given by the corresponding block of the inverse of \mathbf{M} and thus depends also on the parts of \mathbf{M} corresponding to unsupervised points. In this manner unsupervised data helps to identify the manifold structure of the data. The manifold structure is reflected in the covariance matrix, the entries of which are not simply given by (a transformation of) the Euclidean distance between data points, but also depend on other data points that indicate how distant the points are on the manifold.

Semi-supervised learning is a machine learning approach that is very relevant when working with high-dimensional data sets (e.g. images or text); in particular for nearest neighbor based methods, due to their flexible non-parametric nature. Unfortunately nearest neighbor methods need to access the training data to make predictions for future data. In future research we want to consider combinations of nearest-neighbor models and parametric models. In particular we want to use semi-supervised GF techniques (like those described here and in [18]) to define a prior on the partially observed variable of interest \mathbf{y} (e.g. a class label or response variable). This prior can then be used to bias parametric model learning algorithms, like the latent variable model proposed in [13], towards models that comply with the GF distribution on \mathbf{y} .

References

- [1] M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Machine Learning Journal*, 56(1–3):209–239, 2004.
- [2] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.
- [3] J. H. Ham, D. D. Lee, and L. K. Saul. Learning high dimensional correspondences from low dimensional manifolds. In *ICML 2003, workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, 2003.
- [4] 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.
- [5] T. Joachims. Transductive learning via spectral graph partitioning. In T. Fawcett and N. Mishra, editors, *Proceedings of the twentieth international conference on machine learning*, pages 290–297. AAAI Press, 2003.
- [6] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 741–750. ACM Press, New-York, NY, USA, 2002.

- [7] P. Niyogi M. Belkin. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003.
- [8] D. J. C. MacKay. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1991.
- [9] M. Mahdaviani, N. de Freitas, B. Fraser, and F. Hamze. Fast computational methods for visually guided robots. In *Proceedings of the International Conference on Robotics and Automation*, 2005. to appear.
- [10] R.M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.
- [11] G. Peters, B. Zitova, and C. von der Malsburg. How to measure the pose robustness of object views. *Image and Vision Computing*, 20(4):249–256, 2002.
- [12] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.
- [13] S. T. Roweis, L. K. Saul, and G. E. Hinton. Global coordination of local linear models. In T. G. Dietterich and S. Becker and Z. Ghahramani, editor, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, Cambridge, MA, USA, 2002.
- [14] 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.
- [15] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.
- [16] 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. MIT Press, Cambridge, MA, USA, 2004.
- [17] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003, workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, 2003.
- [18] X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: From gaussian fields to gaussian processes. Technical Report CMU-CS-03-175, CMU, 2003.

Acknowledgements

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.

IAS reports

This report is in the series of IAS technical reports. The series editor is Bas Terwijn (bterwijn@science.uva.nl). Within this series the following titles appeared:

J.M. Porta, M.T.J. Spaan, and N. Vlassis. *Value iteration for continuous-state POMDPs*. Technical Report IAS-UVA-04-04, Informatics Institute, University of Amsterdam, The Netherlands, December 2004.

W. Zajdel, A.T. Cemgil, and B.J.A. Kröse. *A hybrid graphical model for online multi-camera tracking*. Technical Report IAS-UVA-04-03, Informatics Institute, University of Amsterdam, The Netherlands, November 2004.

M.T.J. Spaan and N. Vlassis. *Perseus: randomized point-based value iteration for POMDPs*. Technical Report IAS-UVA-04-02, Informatics Institute, University of Amsterdam, The Netherlands, November 2004.

All IAS technical reports are available for download at the IAS website, <http://www.science.uva.nl/research/ias/publications/reports/>.