Large Margin Non-Linear Embedding

Alexander Zien ZIEN@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany

Joaquin Quiñonero Candela JQC@TUEBINGEN.MPG.DE
Friedrich Miescher Laboratory, Max Planck Society, 72076 Tübingen, Germany

Abstract

It is common in classification methods to first place data in a vector space and then learn decision boundaries. We propose reversing that process: for fixed decision boundaries, we "learn" the location of the data. This way we (i) do not need a metric (or even stronger structure) – pairwise dissimilarities suffice; and additionally (ii) produce low-dimensional embeddings that can be analyzed visually. We achieve this by combining an entropy-based embedding method with an entropy-based version of semi-supervised logistic regression. We present results for clustering and semi-supervised classification.

1. Introduction

Non-linear embedding (aka dimensionality reduction) can be useful for several purposes: eg, for data visualization, or as a pre-processing step to clustering or to supervised learning. In the latter case, using unlabeled data in the embedding essentially makes the entire procedure semi-supervised.

This paper proposes to integrate the embedding with a clustering or with a classification into a single method. To do so, we impose class boundaries in the embedding space and encourage the data points to keep some distance from them, i.e. implement a margin. This is in contrast to conventional classification approaches which fix the data in some (high-dimensional) space, often by means of a positive definite kernel function, and then optimize the (linear) class boundaries. Reversing this setup, that is fixing the boundaries and learning the points, has potential advantages: working in low-dimensional spaces may be easier, the data can be visualized, and the requirements on the input data are more relaxed (no metric nor kernel required).

To implement our approach, we extend an existing embedding technique by introducing the boundaries and a repulsive force on the data points. Popular techniques for non-linear embedding include for example Locally Linear Embedding (LLE) (Roweis & Saul, 2000), Isomap (Tenenbaum et al., 2000) Self-Organizing Maps (SOM) (Kohonen, 1988), and Stochastic Neighbor Embedding (SNE) (Hinton & Roweis, 2003). In the following, we will work with SNE, since it offers properties that we will take advantage of: it can use as input pairwise dissimilarities, and its cost function is expressed in terms of entropy.

Recently, a method called Parametric Embedding (PE) based on SNE has been proposed, that simultaneously embeds objects and their classes (Iwata et al., 2005). It is worth noting a fundamental difference between PE and our proposed method. The latter operates on pairwise relationships between the objects to be embedded, possibly augmented with "labels" (ie, class probabilities) for a fraction of them. The output consists of an embedding together with a prediction of class membership for each object. In contrast, PE requires information on the relationship between objects and classes for all objects *as input*, and does not produce predictions. It may therefore be considered purely a change of representation.

The paper proceeds as follows. In Section 2, we briefly review SNE as our starting point. We extend it step by step: we add a class boundary for binary clustering (Section 3); we take labeled data into account to achieve semi-supervised learning (Section 4); and we generalize it to multiple classes (Section 5). Then, in Section 6, we suggest a heuristic to optimize the resulting cost function and set parameter values. Next we perform experiments for clustering and semi-supervised classification (Section 7). The paper concludes with a discussion.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

2. Stochastic Neighbor Embedding

Consider a set of n objects, and suppose that for each object i we are given some dissimilarity d_{ij} to every other object j. By definition, d_{ii} will be ∞; we do not consider any point as its own neighbor. Given the point i, the asymmetric "probability" p_{ij} that point j is its neighbor can be, for example, computed as:

$$p_{ij} = \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)} \ . \tag{1}$$

In fact, the set of neighborhood probabilities $\{p_{ij}\}$ contains all the information that SNE uses about the original objects.

SNE aims at representing the original objects as points $\mathbf{y}_i$ in a low-dimensional Euclidean space (embedding). To achieve this, probabilistic neighborhoods are defined between those representations:

$$q_{ij} = \frac{\exp(-\|\mathbf{y}_i - \mathbf{y}_j\|^2)}{\sum_{k \neq i} \exp(-\|\mathbf{y}_i - \mathbf{y}_k\|^2)} \ . \tag{2}$$

In a good embedding, the induced neighborhood $Q_i = (q_{ij})_{j=1,\ldots,n}$ of any object i should resemble the corresponding original neighborhood $P_i = (p_{ij})_{j=1,\ldots,n}$. Notice that the Q_i's are invariant wrt rotation and translation of the embedding, and that its arbitrary scaling is imposed by choosing a variance of $1/2$ in the computations of the q_{ij} in (2). Hinton and Roweis (2003) propose to minimize the sum of Kullback-Leibler (KL) divergences (aka relative entropies) between the Q_i and the P_i:

$$\mathrm{C} = \sum_i KL\left(P_i \| Q_i\right) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \ . \tag{3}$$

Although SNE puts more effort in preserving relations of high similarity (due to the weighting of the log ratio by p_{ij}), it also tries to keep embedded points distant which are dissimilar in the original space. This is an improvement over non-linear embedding methods such as LLE (Roweis & Saul, 2000), where dissimilar objects may end up embedded as close neighbors.

The embedding is computed by minimizing C (3) with respect to the $\mathbf{y}_i$'s. The gradient of C,

$$\frac{\partial \mathrm{C}}{\partial \mathbf{y}_i} = \sum_j (\mathbf{y}_i - \mathbf{y}_j)(p_{ij} - q_{ij} + p_{ji} - q_{ji}) \ , \tag{4}$$

is very interpretable: $\mathbf{y}_i$ is pulled towards or pushed away from $\mathbf{y}_j$ according to the difference between the original neighborhood probabilities and those induced by the embedding. The minimization has to be performed with care because many local minima exist (Hinton & Roweis, 2003).

3. Enforcing a Boundary and a Margin

We make the assumption that two classes (or categories) of objects exist which are labeled "+1" and "−1", and we would like to discover them.

We divide the embedding space in two regions of different class membership by means of a *boundary* defined by $y_{i1} = 0$, where y_{i1} denotes the first component of $\mathbf{y}_i$. This choice is just for convenience; the location of the boundary is arbitrary due to the translation- and rotation-invariance of the SNE embedding.

Let M_i be the binomial probability distribution of class membership for object i, with $M_i(+1) = m_i$ and $M_i(-1) = 1 - m_i$. We estimate the probability that object i belongs to class "+1" by the logistic function

$$m_i = \frac{1}{1 + \exp\left(-\frac{1}{\lambda} y_{i1}\right)} \ . \tag{5}$$

Here, $\lambda > 0$ is a parameter that can be interpreted as the length scale of the first component of $\mathbf{y}$; alternatively, it can be seen as an inverse measure of steepness of the logistic probability distribution.

In order to discover structure we want the embedding to induce high class membership probabilities, i.e. that the m_i are either close to 0 or to 1. But we need to avoid the trivial solution with all points embedded far on the same side of the boundary – we want a certain balance around the boundary.

3.1. Generalized Jensen-Shannon Divergence

Unlike most measures of difference, the generalized Jensen-Shannon (JS) divergence (Lin, 1991) is applicable to more than two probability distributions. Consider our set of n class membership distributions M_i. Their generalized JS divergence is defined as[1]

$$JS(M_1, \ldots, M_n) = \underbrace{H\left(\frac{1}{n}\sum_i M_i\right)}_{B\text{: balancing}} \underbrace{- \frac{1}{n}\sum_i H(M_i)}_{A\text{: margin}} \ , \tag{6}$$

where $H(\cdot)$ is Shannon's entropy. Maximizing this measure is exactly what we need to satisfy our requirement that the embedded points should organize themselves around the boundary.

To see this, first consider the rightmost term in (6),

[1] The actual definition allows for weighing the probability distributions: $JS(M_1, \ldots, M_n) = H\left(\sum_i \pi_i M_i\right) - \sum_i \pi_i H(M_i)$. We use uniform weights, $\pi_i = 1/n$, corresponding to the assumption that the data points are iid.

that is the negative average of entropies:

$$\mathrm{A} = \frac{1}{n}\sum_i \left[m_i \log m_i + (1-m_i)\log(1-m_i)\right] \ . \quad (7)$$

For our binomial distributions, it will be maximum (ie, zero) when all m_i are either equal to 0 or to 1. In combination with the definition of the class probabilities m_i, this engenders a margin around the boundary, by encouraging points to move away from the boundary. The margin is "soft", ie. there may be points inside it, since a high entropy may be compensated for by a decrease in the SNE cost C. Minimizing such entropies has recently been suggested for semi-supervised learning (Grandvalet & Bengio, 2005).

Consider now the first term of the JS divergence (6), which is the entropy of the average $\bar{M}$ of the M_i:

$$\mathrm{B} = -\bar{m}\log\bar{m} - (1-\bar{m})\log(1-\bar{m}) \ , \quad (8)$$

where $\bar{m} = \frac{1}{n}\sum_i m_i$. It is maximum when the average probability $\bar{m}$ is 0.5, that is when the embedded points are distributed on both sides of the boundary in a balanced manner. In case there exists prior knowledge on the (unequal) relative class sizes, this may be incorporated by changing $\mathrm{B} = H(\bar{M}) = H(\frac{1}{2}) - KL\left(\bar{M}\middle\|\frac{1}{2}\right)$ into $\mathrm{B} = -KL\left(\bar{M}\middle\|\pi_{+1}\right)$, where π_{+1} is the prior probability of class "+1".

3.2. Cost Function and Derivative

We augment the SNE cost function (3) by subtracting the generalized JS divergence (6). This additional term will only affect the first component of the vectors $\mathbf{y}_i$. It is then enough to take derivatives of A and B with respect to y_{i1}:

$$\frac{\partial \mathrm{A}}{\partial y_{i1}} = \frac{1}{n\lambda^2}\sum_i m_i(1-m_i)y_{i1} \ . \quad (9)$$

Points get pushed away from the boundary, the more so, the closer they are to it. This creates a margin around the boundary, the size of which depends on λ. Let us now take the derivative of B:

$$\frac{\partial \mathrm{B}}{\partial y_{i1}} = \frac{1}{n\lambda}\log\left(\frac{1-\bar{m}}{\bar{m}}\right) m_i(1-m_i) \ . \quad (10)$$

Roughly, the $log((1-\bar{m})/\bar{m})$ factor pushes point y_i to the side of the boundary that contains less points. For example, if $\bar{m} < 0.5$, then y_i will be pushed towards the class "+1" side. This force is then moderated by the $m_i(1-m_i)$ factor. Points far from the boundary will be less affected than points near to it. Intuitively this makes sense, since we are already more certain about the class of points which are far from the boundary.

3.3. Fisher and the Margin

If we consider P given, the method outlined so far has only two parameters: the dimension of the space to embed into, and the scale parameter λ of the logistic function (5). In fact, λ determines the size of the (soft) margin around the boundary, through the term A (7) of the Jensen-Shannon divergence. The larger λ is, the further a point must be pushed away from the boundary to achieve an m_i close enough to 0 or to 1 to satisfy A.

It seems reasonable to select the value of λ that maximizes the margin. It would seem at first sight that λ should be large, since this maximizes the margin, in theory. However, λ needs to be commensurate to the scale of the $\mathbf{y}_i$'s computed by standard SNE embedding. Indeed, for too large a λ the derivative of the term A does not have enough strength to compete with that of the SNE cost term C.

We propose to use the Fisher criterion (see e.g. Bishop (1995)) as a measure of the margin:

$$\mathrm{J} = \frac{\left[\bar{y}_1(+1) - \bar{y}_1(-1)\right]^2}{\mathrm{var}\left(y_1(+1)\right) + \mathrm{var}\left(y_1(-1)\right)} \ , \quad (11)$$

where $\bar{y}_1(+1)$ is the mean of the first dimension of the points of class "+1", and $\mathrm{var}\left(y_1(+1)\right)$ is their variance. It has the advantage of being a *relative* criterion, independent of any scaling of the embedded points. It also directly matches our goal, which is to discriminate one class from another along the vector normal to the boundary. We choose the value of λ that gives an embedding with maximum value of J.

4. The Semi-Supervised Case

So far, we have concentrated on the unsupervised case, where no information about the class membership of our objects is available. However, it is not uncommon that for a fraction of the objects a label is known. This scenario is commonly referred to as "semi-supervised learning", and has gained much interest recently.

We want our method to be able to handle partially labeled data and to deal with noisy (or uncertain) labels. We suppose that for a subset of our objects we are given "soft" class membership labels in terms of probabilities of the object belonging to class "+1". If object i is labeled, then we are given its "supervised" class membership binomial probability distribution T_i, with $T_i(+1) = t_i$ and $T_i(-1) = 1 - t_i$. The soft label of point i is thus effectively its probability t_i of belonging to class "+1".

The label distribution T_i will have an effect on the

embedding of point i through its class membership distribution M_i, defined in (5). We will encourage M_i to match T_i as well as possible by minimizing the Kullback-Leibler divergence these two distributions.

We extend our method to deal with probabilistic labels by adding the following term to the cost function:

$$\begin{aligned} \mathrm{D} &= \sum_{i \in \mathcal{L}} KL\left(T_i \| M_i\right) \\ &= \sum_{i \in \mathcal{L}} H(T_i) - t_i \log m_i - (1-t_i)\log(1-m_i) \ , \end{aligned} \tag{12}$$

where $\mathcal{L} \subset \{1, \ldots, n\}$ is the set of objects with known labels. Given that the m_i depend on the embedded points only through their distance to the boundary, we compute derivatives with respect to y_{i1}:

$$\frac{\partial \mathrm{D}}{\partial y_{i1}} = \frac{1}{\lambda}(m_i - t_i) \ . \tag{13}$$

Suppose t_i is large (ie, close to 1) indicating that point i very likely belongs to class "+1". If m_i is smaller than t_i (for example if $\mathbf{y}_i$ is on the "-1" side of the boundary), then the gradient is smaller than zero. A gradient descent step to minimize D causes the value of y_{i1} to increase, pulling it towards the correct side of the boundary. The opposite (desired) behaviour takes place if t_i is smaller than m_i.

As we expected, minimizing the Kullback-Leibler divergence between the T_i and the M_i has the effect of pulling labeled points towards the side of the boundary of the class they are most likely to belong to.

The final cost function for semi-supervised embedding thus reads

$$\underbrace{\sum_i KL\left(P_i \| Q_i\right)}_{\text{C: } SNE} \underbrace{- JS(M_1, \ldots, M_n)}_{\text{A+B: } boundary} + \underbrace{\sum_{i \in \mathcal{L}} KL\left(T_i \| M_i\right)}_{\text{D: } labels} . \tag{14}$$

An additional weighting of the different terms of the cost function could of course be introduced. We prefer not to do so for two reasons. First, minimizing the number of parameters is desirable given the heuristic nature of our method. Also the effect of weighting of the A + B term is captured by varying λ. Second, all terms can being expressed as entropies, the unweighted summation may help us in our aim of giving an information theoretic interpretation to the whole cost function. We call the method of minimizing this cost Mbed.

5. Extension to Multiple Classes

We have so far concentrated exclusively on the case of two classes. Extending our framework to handle multiple classes is straightforward. Suppose we want to handle K different classes. We need to the class membership distribution M_i of point i to be a multinomial now. We achieve this by replacing the sigmoid function (5) by a "softmax". The probability that point i belongs to class k is now given by:

$$m_{ik} = \frac{\exp\left(\frac{1}{\lambda}\left\langle \mathbf{w}_k, \mathbf{y}_i \right\rangle\right)}{\sum_l \exp\left(\frac{1}{\lambda}\left\langle \mathbf{w}_l, \mathbf{y}_i \right\rangle\right)} \ , \tag{15}$$

where there is one projection vector $\mathbf{w}_k$ for each class.

The directions $\mathbf{w}_k$ have to be chosen with care, because they carry implications about the relationships between the classes. As an example, consider four classes that are to be embedded into two-dimensional space with their $\mathbf{w}_k$ being the four axis-parallel directions. Then each class can be neighbor to only two other classes, forcing two pairs of classes to have larger mutual distance. This can be seen as a geometric way of encoding prior knowledge about the structure of the classes.

However, when no such prior knowledge exists, it is usually preferred to operate in the most general setting, i.e. not to impose any structure. In our case this means that each class direction $\mathbf{w}_k$ should have unit length and should have the same angle to every other class direction $\mathbf{w}_l$. To realize this, at least $K-1$ dimensions are required for K classes.

For semi-supervised applications, the T_i have to be generalized to multinomials as well: now there exists a variable t_{ik} for each class k, indicating the probability with which the object i belongs to this class.

For computing the derivative of the cost function, it will be useful to note that

$$\frac{\partial m_{ik}}{\partial \mathbf{y}_i} = \frac{1}{\lambda} m_{ik} \left(\mathbf{w}_k^\top - \sum_l m_{il} \mathbf{w}_l^\top \right) \ . \tag{16}$$

The derivative of A (7) is minus the mean of the terms

$$\frac{\partial H(M_i)}{\partial \mathbf{y}_i} = \frac{1}{\lambda^2} \left[\left(\sum_l m_{il} \mathbf{w}_l^\top \right) \left(\sum_k \gamma_{ik} \right) - \sum_k \gamma_{ik} \mathbf{w}_k^\top \right] \ , \tag{17}$$

where $\gamma_{ik} := m_{ik} \left\langle \mathbf{w}_k, \mathbf{y}_i \right\rangle$. For B (8), we obtain

$$\frac{\partial H(\bar{M})}{\partial \mathbf{y}_i} = \frac{1}{\lambda n} \sum_k \left(\sum_l m_{il} \mathbf{w}_l^\top - \mathbf{w}_k^\top \right) m_{ik} \log \bar{m}_k \ . \tag{18}$$

Further, the derivative of D (12) is computed by

$$\frac{\partial KL\left(T_i \| M_i\right)}{\partial \mathbf{y}_i} = \frac{1}{\lambda} \sum_k \left(m_{ik} - t_{ik}\right) \mathbf{w}_k^\top \ . \tag{19}$$

Finally, we have to generalize the Fisher criterion (11) to multiple classes. From Multivariate Analysis of Variance (MANOVA) we adopt the Hotelling-Lawley trace, which quantifies the dissimilarity of mean vectors of multiple populations. It is defined as

$$\mathrm{J} = \mathrm{tr}\left[\Sigma_B \Sigma_W^{-1}\right] \quad , \tag{20}$$

where Σ_W is the matrix of covariance within the classes, and Σ_B is the matrix of covariance between the classes.

6. Optimization

Since our objective is non-convex, optimization is difficult. We resort to a search for local minima by conjugate gradient (CG) descent.[2]

As pointed out earlier, the value of λ should be chosen relative to the scale of the SNE embedding, which we estimate by the radius r of the sphere encompassing the $\mathbf{y}_i$'s after a crude SNE embedding (10 CG steps). We approximate r by the geometric mean of the component-wise standard deviations of the $\mathbf{y}_i$'s.

Assuming a Gaussian distribution of the SNE-embedded points, about 68% of them are within distance r of the origin, and about 95% within $2r$. Figure 1 shows the logistic function (5) for different values of λ as a multiple of the radius. It seems that 10 to $\frac{1}{10}$ is a reasonable range of values for λ. We implement a scheme that "anneals" λ by decreasing it logarithmically from 10 to 10^{-1}. In first experiments, 20 steps seemed to be a reasonable choice.

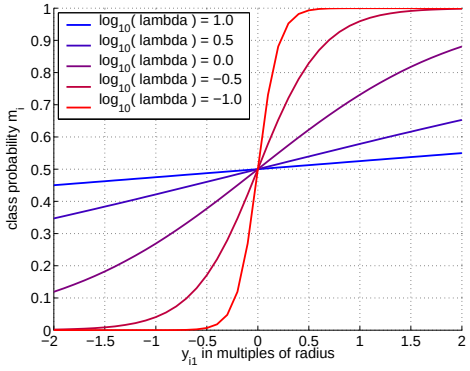

Figure 1. During the annealing the shape of the logistic function changes from that shown in blue to that in red.

For semi-supervised learning, the "confidence" in the labels is gradually increased. We want to start with rather low confidence in the labels, since absolute certainty ($t_{ik} \in \{0, 1\}$) corresponds to locations at infinity (for part D of the cost function). We choose to let the confidence correspond to a point placed at distance $2r$ along a direction $\mathbf{w}_k$ from the origin. Formally, based on (15), with $\mathbf{y}_* = r\mathbf{w}_k$ and using $\langle \mathbf{w}_k, \mathbf{w}_l \rangle = -1/(K-1)$ for $k \neq l$, we get

$$\begin{aligned} t_{*k} &= \frac{\exp\left(\frac{1}{\lambda}\langle \mathbf{w}_k, \mathbf{y}_* \rangle\right)}{\sum_l \exp\left(\frac{1}{\lambda}\langle \mathbf{w}_l, \mathbf{y}_* \rangle\right)} \\ &= \frac{\exp\left(\frac{1}{\lambda} r\right)}{(K-1)\exp\left(-\frac{1}{\lambda} r \frac{1}{K-1}\right) + \exp\left(\frac{1}{\lambda} r\right)} \end{aligned} \tag{21}$$

as desired confidence. For an object i with a given class label k, we thus set t_{ik} to t_{*k}, and t_{il} to $(1-t_{*k})/(K-1)$ for $l \neq k$.

Finally, the dimension d of the embedding space is set to K. The rationale is that we need at least $K-1$ dimensions to encode "uniform" class priors, where each class is equally similar to the rest. Adding one extra dimension appears to be useful as it gives an additional degree of freedom for fitting P. However, increasing the dimensionality too much tends to decrease the classification accuracy (apart from increasing computational expense).

The resulting algorithm is shown as Algorithm 1. A matlab implementation is available at `http://www.kyb.tuebingen.mpg.de/bs/people/zien/Mbed/`. In line 7 the value of λ appears to be set to twice the values suggested before. This is because the above discussion referred to the binary case with the logistic function defined by (5), while the algorithm uses the multi-class version (15). When $K = 2$, they relate via $m_{i1}(\lambda) = m_i(\lambda/2)$.

7. Experiments

We perform unsupervised and semi-supervised experiments, corresponding to the two kinds of applications of our method. Our example data sets consist of vectorial data and we compute pairwise Euclidean distances d_{ij} using (1) to obtain neighborhood probabilities. For a K-class problem, we set the length scale σ to the $1/K$ quantile of the distribution of all pairwise distances.

7.1. Unsupervised

As an illustrative example, we use Mbed to embed the digits 2, 4 and 8 from the test set of the USPS handwritten digits into a 3-dimensional space using 10 CG steps per annealing step. Figure 2 shows the first two components of the result (right), next to an SNE embedding into 2-dimensional space (left) as reference.

[2] We use the function "minimize" by Carl Rasmussen, available at `http://www.kyb.tuebingen.mpg.de/~carl`.

Algorithm 1 The Mbed algorithm.

Require: $P \in \mathbb{R}^{n \times n}$ {pairwise neighborhood matrix of objects}
Require: K {number of classes}
Require: $\mathcal{L}, t : \mathcal{L} \to \{1, \ldots, K\}$ {(optional) labels for semi-supervised learning}
Require: s_0, s {number of CG optimization steps in initial and subsequent annealing steps}
$d \leftarrow K$ {embed into as many dimensions as there are classes}
{**STEP 1: determine SNE radius**}
$Y_0 \leftarrow \mathcal{N}(0, 0.1\mathbf{I}) \in \mathbb{R}^{n \times d}$ {initialize the embedding with random locations from a normal distribution}
$Y_{SNE} \leftarrow minimize_Y(s_0, SNE(Y), Y_0)$ {perform s_0 steps of CG with the SNE objective function (3)}
$r \leftarrow radius(Y_{SNE})$ {compute the approximate radius of the embedded points}
{**STEPS 2-20: anneal** λ}
$Y_1 \leftarrow \mathcal{N}(0, 10^{-3} r\mathbf{I}) \in \mathbb{R}^{n \times d}$ {initialize the embedding with small scale random values}
for $j = 2$ to 20 **do**
  $\lambda \leftarrow 2 * r * 10/100^{(j-2)/(p-2)}$ {let λ decrease from $2r * 10$ to $2r/10$; cf Figure 1}
  $\tau \leftarrow \exp\left(\frac{1}{\lambda} r\right) / \left[(K-1)\exp\left(-\frac{1}{\lambda} r \frac{1}{K-1}\right) + \exp\left(\frac{1}{\lambda} r\right)\right]$ {cf (21)}
  $t^j_{ik} \leftarrow t_{ik}\tau + (1 - t_{ik})(1 - \tau)$ for all $i \in \mathcal{L}$
  $Y_j \leftarrow minimize_Y(s, Mbed(Y), Y_{j-1}, \lambda, t^j)$ {perform s CG steps with augmented objective function (14)}
end for
return Y_j that maximizes criterion J {cf (20)}

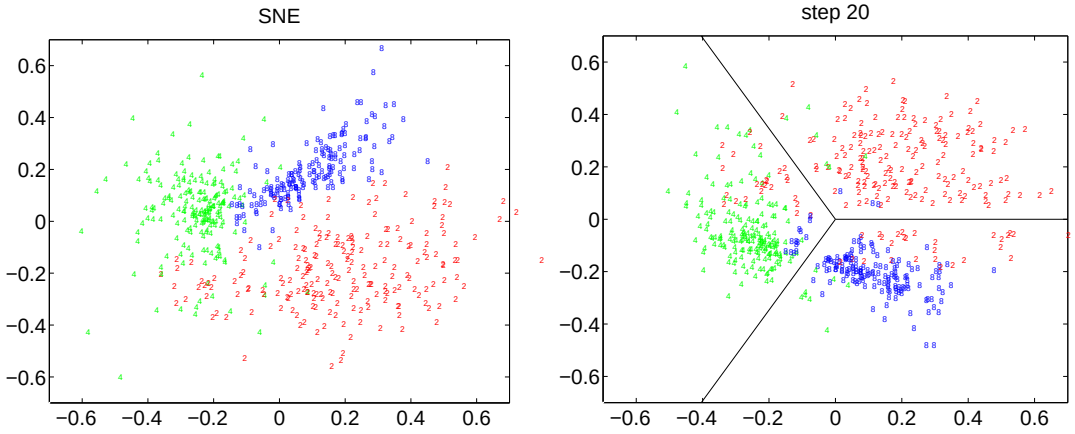

Figure 2. Embedding (SNE, left) and clustering (Mbed, right) of USPS digits 2, 4 and 8. Digits are color-coded for better visual impression.

Further, we cluster four datasets from Section 7.2 with Mbed and with traditional k-means, ignoring the labels. We give both methods the correct number of classes, and train with 10 re-initializations. Table 1 shows the percentage of pairs of points with inconsistent attributed classes: points with identical label assigned to different clusters, and vice-versa. Notice that this measure is not trivially comparable with the classical misclassification error used in Section 7.2.

For g50c, Mbed performs only slightly worse than k-means, which implements the optimal model for this data set. For g10n both algorithms are invalidated because the cluster assumption does not hold. TEXT is very high-dimensional and non-trivial; here Mbed benefits from performing dimensionality reduction. For COIL, the cluster assumption is strongly present which k-means benefits from, and the embedding dimensionality, 20, is too high for Mbed to perform excellently.

	g50c	**g10n**	**TEXT**	**COIL**
k-means	9.02%	50.09%	18.78%	4.62%
Mbed	9.68%	50.09%	10.03%	5.90%

Table 1. Percentage of inconsistently labeled pairs of points for Mbed and k-means in the purely unsupervised case.

7.2. Semi-Supervised

We extend the unsupervised example by introducing 50 labeled points Figure 3 shows the resulting embedding (right), compared to the unsupervised one (left). The number of misclassified points decreases. The bottom of the figure shows a series of annealing steps.

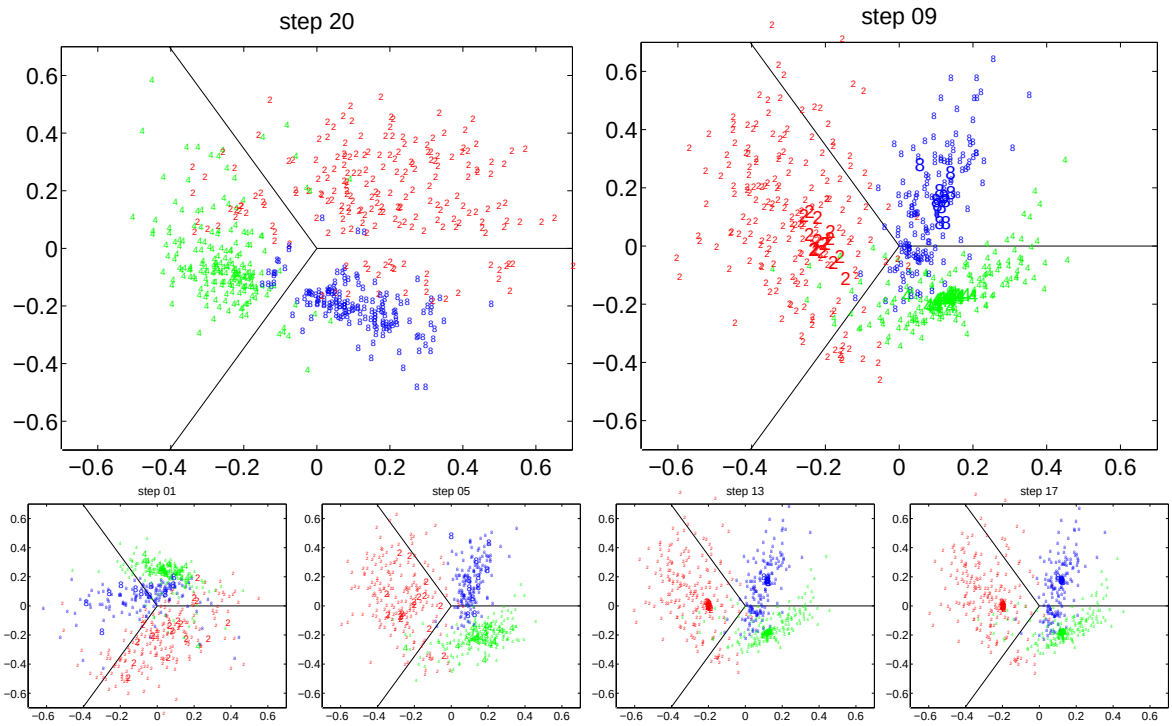

Figure 3. Clustering (right) and semi-supervised classification (left) of USPS digits 2, 4 and 8. In the clustering, step 20 was chosen by the margin criterion (20), in the classification it was step 9. Steps 1, 5, 13, and 17 of the semi-supervised annealing are shown at the bottom. Labeled objects are emphasized by larger font size.

	manifold	**TSVM**	**LDS**	**Mbed**
g50c	17.30%	6.87%	5.62%	5.32%
g10n	30.64%	14.36%	9.72%	38.30%
TEXT	11.71%	7.44%	5.13%	5.90%
USPS	21.30%	26.46%	15.79%	30.57%
COIL	6.20%	26.26%	4.86%	40.44%

Table 2. Average test error of Mbed and three reference methods on five data sets.

To evaluate Mbed thoroughly in semi-supervised applications, we adopt the benchmark used by Chapelle and Zien (2005) (including the splits in labeled and unlabeled points)[3]. Note the absence of free parameters in Mbed as specified in Algorithm 1: no cross-validation is necessary and all training labels are used. We perform 500 CG steps in total for each optimization: $s_0 = 10$ are used for the initial SNE, the remaining are distributed evenly over the annealing steps.

Table 2 lists the test errors achieved by Mbed, averaged over the ten splits. For comparison, results reported by Chapelle and Zien (2005) for three other methods are also supplied: "manifold", akin to the algorithm proposed by Belkin and Niyogi (2004); Transductive Support Vector Machine (TSVM), optimized as described by Joachims (1999); and Low Density Separation (LDS) (Chapelle & Zien, 2005). Note that the performances may be over-optimistic for "manifold" and TSVM, since for these methods model selection was carried out by minimizing the test error.

While Mbed yields excellent performances on the g50c and TEXT data sets, the results for the other sets are rather poor. We hypothesize two potential causes:

- For g10n, the "cluster assumption" is not valid since the class boundary cuts through the centers of two clusters (Chapelle & Zien, 2005). However, the unsupervised parts of our cost function act to conserve the clustering (C) and drive the clusters away from the class boundary (B, A).

- While the other data sets are binary, USPS and COIL have 10 and 20 classes, respectively. Thus, we need to embed into relatively high dimensional spaces which makes the problem of local optima severe, and optimization more difficult.

Further experiments have to be conducted to confirm that the above explainations hold, and that the observed behavior is not due to deficiencies of the cost function itself.

[3]The data sets are available `http://www.kyb.tuebingen.mpg.de/bs/people/chapelle/lds/`.

8. Discussion

We have proposed an alternative view on the problem of relating data points to class boundaries: in contrast to existing methods, the boundaries are fixed and the data points are arranged relative to them. This approach assumes the presence of meaningful structure in the data ("cluster assumption").

Among the merits of this approach are that (i) weak requirements are imposed on the input data (no metric nor even positive definite kernel is required); (ii) there is only one parameter, the number K of classes, for unsupervised learning, and no parameter for semi-supervised learning, where K is known; and (iii) a new representation for the data is found, which can be useful for further analysis.

There are also a few open problems with our method:

- λ is determined heuristically, which makes the class probabilities assigned to the objects meaningless. Investigating the relation between λ, the margin, the radius of the SNE embedding, and the misclassification rate may inspire a more principled approach. The connection between entropy and Bayesian inference may prove useful (Bishop, 1995). Ideally λ should be learnt, and posterior class probabilities be provided as output.
- The optimization is non-convex and thus difficult. The annealing may help escaping undesirable local minima, but probably better strategies could be devised.
- As is the case for many other semi-supervised learning methods, Mbed is too slow for large data sets. However, Mbed may profit from work on efficient optimization of the SNE cost function (Nam et al., 2004).
- Mbed is transductive, i.e. it gives predictions only for the unlabeled points used in training. Classifying a new point ideally requires rerunning the entire optimization, which is costly. Suboptimal approaches may be investigated, like optimizing only over the position of the new point.

However, the presented results indicate that the method may already be useful for moderately sized binary problems, especially when visualization is desirable.

Acknowledgements Thanks to Matthias Hein for discussion about the Jensen-Shannon divergence. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

Belkin, M., & Niyogi, P. (2004). Semi-supervised learning on Riemannian manifolds. *Machine Learning*, *56*, 209–239.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Chapelle, O., & Zien, A. (2005). Semi-supervised classification by low density separation. *Tenth International Workshop on Artificial Intelligence and Statistics*.

Grandvalet, Y., & Bengio, Y. (2005). Semi-supervised learning by entropy minimization. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*, vol. 17. Cambridge, MA: MIT Press.

Hinton, G., & Roweis, S. (2003). Stochastic neighbor embedding. *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press.

Iwata, T., Saito, K., Ueda, N., Stromsten, S., Griffiths, T. L., & Tenenbaum, J. B. (2005). Parametric embedding for class visualization. In L. K. Saul, Y. Weiss and L. Bottou (Eds.), *Advances in neural information processing systems 17*. Cambridge, MA: MIT Press.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *International Conference on Machine Learning*.

Kohonen, T. (1988). *Self-organization and associative memory*. Berlin, Germany: Springer.

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, *37*, 145–151.

Nam, K., Je, H., & Choi, S. (2004). Fast stochastic neighbor embedding: A trust-region algorithm. *Proc. Int'l Joint Conf. Neural Networks (IJCNN)* (pp. 123–128). Budapest, Hungary.

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

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