STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIII**, Number 2, 2008

ON SUPERVISED AND SEMI-SUPERVISED k-NEAREST NEIGHBOR ALGORITHMS

ZALÁN BODÓ AND ZSOLT MINIER

Abstract. The k-nearest neighbor (kNN) is one of the simplest classification methods used in machine learning. Since the main component of kNN is a distance metric, kernelization of kNN is possible. In this paper kNN and semi-supervised kNN algorithms are empirically compared on two data sets (the USPS data set and a subset of the Reuters-21578 text categorization corpus). We use a soft version of the kNN algorithm to handle multi-label classification settings. Semi-supervision is performed by using data-dependent kernels.

1. Introduction

Suppose the training data is given in the form $D = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, \ldots, \ell\} \cup \{\mathbf{x}_i \mid i = 1, 2, \ldots, u\}$ where the first set is called the labeled data, while the second is the unlabeled data set, which contains data drawn from the same distribution as the labeled points but there is no label information for them. Usually $\ell \ll u$. We will denote the size of the whole data set by $N = \ell + u$. The $\mathbf{x}_i \in X$ are called the *independent variables*, while the $y_i \in Y$ are the *dependent variables*, $X \subseteq \mathbb{R}^d$, $Y = \{1, 2, \ldots, K\}$, where K denotes the number of classes. In supervised classification we use only the first data set to "build" a classifier, while in semi-supervised classification we additionaly use the second data set that sometimes can improve predictions [11].

Semi-supervised learning (SSL) is a special case of classification; it is halfway between classification and clustering. The unlabeled data can be used to reveal important information. For example, suppose that in a text categorization problem the word "professor" turns out to be a good predictor for positive examples based on the labeled data. Then, if the unlabeled data shows

Received by the editors: September 15, 2008.

2000 *Mathematics Subject Classification.* 68T10, 45H05.

1998 *CR Categories and Descriptors.* I.2.6. [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE – *Learning*.

Key words and phrases. Supervised learning, Semi-supervised learning, k-nearest neighbors, Data-dependent kernels.

This paper has been presented at the 7th Joint Conference on Mathematics and Computer Science (7th MaCS), Cluj-Napoca, Romania, July 3-6, 2008.

that the words "professor" and "university" are correlated, then using both words the accuracy of the classifier is expected to improve. To understand how can one use the unlabeled data to improve prediction, consider the simplest semi-supervised learning method, called self-training or bootstrapping: train the classifier on the labeled examples, make predictions on the unlabeled data, add the points from the unlabeled set with the highest prediction confidence to the labeled set along with their predicted labels, and retrain the classifier. This procedure is usually repeated until convergence.

In order to be able to effectively use the unlabeled data to improve the system's performance some assumptions have to be conceived about the data. These are the smoothness assumption (SA), the cluster assumption (CA) and the manifold assumption (MA): SA says that points in a high density region should have similar labels, that is labels should change in low density regions, CA states that two points from the same cluster should have similar labels, while MA presumes that the data lies roughly on a low-dimensional manifold [7].

Most of the semi-supervised methods can be classified in the following four categories: generative models, low-density separation methods, graph-based methods and SSL methods based on change of representation. Methods belonging to the last category attempt to find some structure in the data which is better emphasized or better observable in the presence of the large unlabeled data set. These algorithms consist of the following following steps:

(1) Build the new representation – new distance, dot-product or kernel – of the learning examples.
(2) Use a supervised learning method to obtain the decision function based on the new representation obtained in the previous step.

Kernels referred in the first step are tools for non-linear extensions of linear algorithms like perceptron, linear support vector machines, kNN, etc. Kernel functions, or simply kernels were proposed for learning non-linear decision boundaries in 1964 in [1], but they became popular after the introduction of non-linear support vector machines (SVMs) in 1992 [6]. Kernel functions are symmetric functions of two variables, which return the "similarity" of two points in a high-dimensional space, without actually mapping the points to that space. More precisely, kernel functions return the dot product of two vectors in a so-called "feature" space:

$$k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})'\phi(\mathbf{z})$$

Any machine learning algorithm in which the input data appears only in the form of dot products can be extended to learn non-linear decision functions by simply using a positive semi-definite kernel function instead of the inner product of the vectors. This is called the "kernel trick". We call the matrix

containing the dot products of the data points – i.e. the Gram matrix – the kernel matrix or simply the kernel. Whether we are referring to the kernel function or the kernel matrix by the expression "kernel" will be clear from the context.

Data-dependent kernels are similar to semi-supervised learning machines: the kernel function does not depend only on the two points in question, but in some form it makes use of the information contained in the whole learning data available. That is the value of $k(\mathbf{x}, \mathbf{z})$ with data set D_1 is not necessarily equal to the value of $k(\mathbf{x}, \mathbf{z})$ with data set D_2, however the kernel function – or more generally the kernel construction method – is the same. This can be formalized as

$$k(\mathbf{x}, \mathbf{z}; D_1) \eqslantless k(\mathbf{x}, \mathbf{z}; D_2)$$

provided that the additional data sets are different, i.e. $D_1 \neq D_2$, where "$\eqslantless$" means "not necessarily equal" and ";" stands for conditioning. In SSL methods with change of representation data-dependent kernels are used.

We will use data-dependent kernels to construct a semi-supervised version of the kNN classifier. These methods then will be empirically compared to another semi-supervised kNN method, the label propagation (LP) algorithm.

The paper is structured as follows. Section 2 introduces the kNN and the "soft" kNN classifier. In Section 3 we present label propagation for binary and multi-class cases. Label propagation can be viewed as a semi-supervised kNN technique. Section 4 describes the kernelization of the kNN classifier and shortly presents three data-dependent kernels, namely the ISOMAP, the multi-type and hierarchical cluster kernels, used in the experiments. The experiments and the obtained results are presented in Section 5. The paper ends with Section 6 discussing the results obtained in the experiments.

2. K-nearest neighbor algorithms

The k-nearest neighbor classification was introduced by Cover and Hart in [10]. The kNN classifier determines the label of an unseen point $\mathbf{x}$ by simple voting: it finds the k-nearest neighbors of $\mathbf{x}$ and assigns to it the winning label among these.

$$\widetilde{f}(\mathbf{x}) = \operatorname*{argmax}_{c=1,2,\ldots,K} \sum_{\mathbf{z} \in N_k(x)} \operatorname{sim}(\mathbf{z}, \mathbf{x}) \cdot \delta(c, f(\mathbf{z}))$$

where the function f assigns a label to a point, $N_k(\mathbf{x})$ denotes the set of k-nearest neighbors of $\mathbf{x}$, K is the number of classes, the function $\operatorname{sim}(\cdot, \cdot)$ returns the similarity of two examples, and $\delta(a, b) = 1$ if $a = b$, 0 otherwise. The function $\operatorname{sim}(\cdot, \cdot)$ is used to give different weights for different points. One choice could be to use some distance metric $d(\cdot, \cdot)$ with the property of

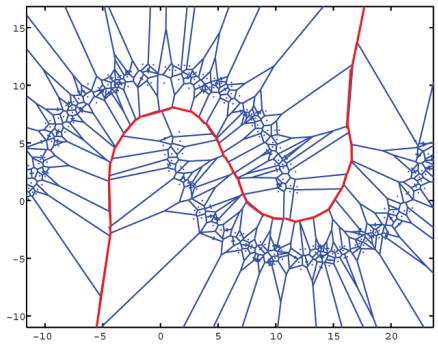

FIGURE 1. Figure showing the 1NN decision boundaries for the two-moons data set.

assigning a lower value to nearby points and a higher value to farther points to $\mathbf{x}$. Then one can choose for example

$$\mathrm{sim}(\mathbf{x}, \mathbf{z}) = \frac{1}{g(d(\mathbf{x}, \mathbf{z}))}$$

where $g(\cdot)$ is an adequate function. If the constant function $\mathrm{sim}(\mathbf{x}, \mathbf{z}) = 1$ is chosen, we arrive to simple kNN, where all the neighbors have the same influence on the predicted label.

In order to work efficiently implement the kNN method, no explicit form of the inductive classifier is built, since representing and storing the decision boundaries can become very complex. On Figure 1 the decision boundaries of a 1NN classifier are shown; we used the popular "two-moons" data set for this illustration. Here we have two classes: the positive class is represented by the upper crescent, while the points of the negative class lie in the lower crescent. The polygons represent the area in which an unseen point gets the label of the point which "owns" the respective cell. The red curve shows the decision boundary between the classes.

2.1. **Soft kNN.** In the soft version of the kNN we average the labels of the surrounding points. That is the prediction function becomes

$$f(\mathbf{x}) = \frac{1}{\sum_{\mathbf{z} \in N_k(\mathbf{x})} W_{\mathbf{zx}}} \sum_{\mathbf{z} \in N_k(\mathbf{x})} W_{\mathbf{zx}} f(\mathbf{z})$$

where $W_{\mathbf{zx}}$ denotes the similarity between $\mathbf{z}$ and $\mathbf{x}$. In case of binary classification, that is $Y = \{-1, 1\}$ or $Y = \{0, 1\}$ we use thresholding after computing the prediction by the above formula, e.g. using the value 0 or 0.5 for the threshold. Thus we arrive to the same decision function.

3. Label propagation

Label propagation was introduced in [23] for semi-supervised learning. It is a transductive graph-based semi-supervised learning technique, i.e. the labels are determined only in the desired points. We can call LP a semi-supervised kNN algorithm, because the label of a point is determined considering only the labels of its neighbors. The only and considerable difference between kNN and LP is that while in LP the labels propagate through the neighbors, and the label of an unseen point depends on the labels of the other unseen/unlabeled points too, the labels are static in kNN and only the labeled points count.

For binary class learning consider the vector $\mathbf{f} \in \{-1, 1\}^N$ of class labels, where $N = \ell + u$. Then the energy/cost function to be minimized is the following

$$E_1(\mathbf{f}) = \frac{1}{2} \sum_{i,j=1}^{N} W_{ij}(f_i - f_j)^2 \tag{1}$$

where f_i, $i = 1, \ldots, \ell$ is fixed according to the labeled training points. If $\mathbf{f}$ is divided as $\begin{bmatrix} \mathbf{f}_L \\ \mathbf{f}_U \end{bmatrix}$ where $\mathbf{f}_L$ and $\mathbf{f}_U$ denote the parts corresponding to the labeled and unlabeled examples, then the optimization problem can be written as

$$\min_{\mathbf{f}_U} E_1(\mathbf{f})$$

It is easy to check that

$$\begin{aligned} E_1(\mathbf{f}) &= \sum_{ij} W_{ij} f_i^2 - \sum_{ij} W_{ij} f_i f_j \\ &= \mathbf{f}'\mathbf{D}\mathbf{f} - \mathbf{f}'\mathbf{W}\mathbf{f} = \mathbf{f}'\mathbf{L}\mathbf{f} \end{aligned}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the graph Laplacian [9] of the similarity matrix of the points. For the sake of simplicity we divide the matrices into the following blocks:

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{LL} & \mathbf{W}_{LU} \\ \mathbf{W}_{UL} & \mathbf{W}_{UU} \end{bmatrix}; \quad \mathbf{D} = \begin{bmatrix} \mathbf{D}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_U \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{LL} & \mathbf{L}_{LU} \\ \mathbf{L}_{UL} & \mathbf{L}_{UU} \end{bmatrix}; \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{LL} & \mathbf{P}_{LU} \\ \mathbf{P}_{UL} & \mathbf{P}_{UU} \end{bmatrix}$$

We want to minimize $E_1(\mathbf{f})$, therefore we calculate its derivative and set to zero. Thus we obtain

$$\mathbf{f}_U = -\mathbf{L}_{UU}^{-1} \cdot \mathbf{L}_{UL} \cdot \mathbf{f}_L \tag{2}$$

or equivalently $(\mathbf{I} - \mathbf{D}_U^{-1}\mathbf{W}_{UU})^{-1}\mathbf{D}_U^{-1} \cdot \mathbf{W}_{UL} \cdot \mathbf{f}_L = (\mathbf{I} - \mathbf{P}_{UU})^{-1}\mathbf{P}_{UL} \cdot \mathbf{f}_L$.

The above energy function can be simply modified for the multi-class, multi-label case:

$$E_2(\mathbf{f}) = \frac{1}{2}\sum_{i,j=1}^{N} W_{ij}\|\mathbf{f}_{i\cdot} - \mathbf{f}_{j\cdot}\|_2^2$$

where now $\mathbf{f} \in \{0,1\}^{N\times K}$. One can observe that $E_2(\mathbf{f}) = \mathrm{tr}(\mathbf{f}'\mathbf{L}\mathbf{f})$. If we decompose $\mathbf{f}$ into column vectors

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}_{\cdot 1} & \mathbf{f}_{\cdot 2} & \cdots & \mathbf{f}_{\cdot K} \end{bmatrix}$$

then the problem can be rewritten as K independent constrained optimization problems involving vectors of size $N \times 1$.

$$\begin{aligned} \mathbf{f}'\mathbf{L}\mathbf{f} &= \begin{bmatrix} \mathbf{f}_{\cdot 1}{}' \\ \mathbf{f}_{\cdot 2}{}' \\ \vdots \\ \mathbf{f}_{\cdot K}{}' \end{bmatrix} \cdot \mathbf{L} \cdot \begin{bmatrix} \mathbf{f}_{\cdot 1} & \mathbf{f}_{\cdot 2} & \cdots & \mathbf{f}^{\cdot K} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{f}_{\cdot 1}{}'\mathbf{L}\mathbf{f}_{\cdot 1} & \cdot & \cdots & \cdot \\ \cdot & \mathbf{f}_{\cdot 2}{}'\mathbf{L}\mathbf{f}_{\cdot 2} & \cdots & \cdot \\ \vdots & \vdots & \ddots & \vdots \\ \cdot & \cdot & \cdots & \mathbf{f}_{\cdot K}{}'\mathbf{L}\mathbf{f}_{\cdot K} \end{bmatrix} \end{aligned}$$

from which it follows that

$$\mathrm{tr}(\mathbf{f}'\mathbf{L}\mathbf{f}) = \mathrm{tr}(\mathbf{f}_{\cdot 1}{}'\mathbf{L}\mathbf{f}_{\cdot 1}) + \ldots + \mathrm{tr}(\mathbf{f}_{\cdot K}{}'\mathbf{L}\mathbf{f}_{\cdot K})$$

that is we can minize now $\mathbf{f}_{\cdot i}{}'\mathbf{L}\mathbf{f}_{\cdot i}$ with respect to $(\mathbf{f}_U)_{\cdot i}$, $i = 1, 2, \ldots, K$ and from these solutions the solution of the original problem can be built. In our notation used above $\mathbf{f}_{\cdot i}$ denotes the ith column, while $\mathbf{f}_{j\cdot}$ denotes the jth row of $\mathbf{f}$.

By calculating the derivative of $E_2(\mathbf{f})$ with respect to $\mathbf{f}_U$, we arrive to the same formula as (2), but $\mathbf{f}$ is now a matrix, not a vector.

The iterative solution for LP is composed of the following steps:

(1) Compute $\mathbf{W}$ and $\mathbf{D}$.
(2) $i = 0$; Initialize $\mathbf{f}_U^{(i)}$.
(3) $\mathbf{f}^{(i+1)} = \mathbf{D}^{-1}\mathbf{W}\mathbf{f}^{(i)}$.
(4) Clamp the labeled data, $\mathbf{f}_L^{(i+1)} = \mathbf{f}_L$.
(5) $i = i + 1$; Unless convergence go to step 3.

Convergence means that the difference between the class assignment matrices $\mathbf{f}$ obtained in two consecutive steps drops below a predefined threshold. The value of the threshold greatly influences the running time of the algorithm. The difference between consecutive solutions can be measured by the Frobenius matrix norm. The convergence of the above algorithm is proven in [23]. On

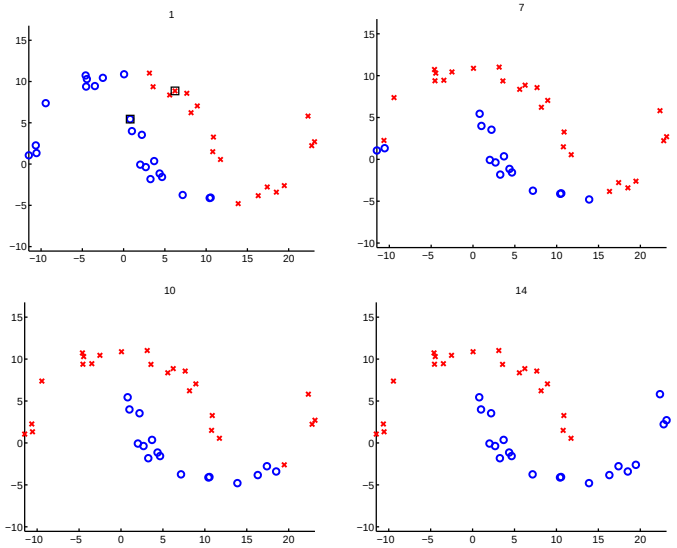

FIGURE 2. The propagation of labels (iteration 1, 7, 10 and 14). At the beginning – step 0, not shown separately here – only the 2 points put in the black squared frames are labeled.

Figure 2 the propagation of labels is illustrated on the two-moons data set, starting from only two labeled points.

If we decompose $\mathbf{f}$, we arrive to the simpler formula

$$\mathbf{f}_U^{(i+1)} = \mathbf{D}_U^{-1}\mathbf{W}_{UL}\mathbf{f}_L + \mathbf{D}_U^{-1}\mathbf{W}_{UU}\mathbf{f}_U^{(i)}$$

or by using the notation $\mathbf{A} = \mathbf{P}_{UL}\mathbf{f}_L = \mathbf{D}_U^{-1}\mathbf{W}_{UL}\mathbf{f}_L$ and $\mathbf{P}_{UU} = \mathbf{D}_U^{-1}\mathbf{W}_{UU}$, we obtain the update formula $\mathbf{f}_U^{(i+1)} = \mathbf{A} + \mathbf{P}_{UU}\mathbf{f}_U^{(i)}$, which can be included into the algorithm by replacing steps 3 and 4.

Consider now the the case of binary classification. Given the solution $\mathbf{f}$ according to the update formula we can write that

$$\begin{aligned} f_i &= (\mathbf{D}^{-1}\mathbf{W})_{i\cdot} \cdot \mathbf{f} \\ &= \frac{1}{\sum_{j=1}^{N} W_{ij}} \cdot \sum_{j=1}^{N} W_{ij} f_j \end{aligned}$$

that is the label of a point is equal to the weighted average of the other points' class labels. When the Gaussian kernel/similarity function is used (which is one of the most common choices in practice), which assigns an exponentially decreasing similarity to the more distant points, those weights can be considered to be equal to zero. Thus we get a kNN-like algorithm, where k changes

dynamically, so this is rather an εNN algorithm, where ε denotes a threshold above which similarity is considered to be 0.

In the multi-class case we can write

$$\mathbf{f}_{\cdot j} = \mathbf{P}\mathbf{f}_{\cdot j}$$

that is

$$f_{ij} = \mathbf{P}_{i\cdot}\mathbf{f}_{\cdot j}$$

which is equivalent to

$$f_{ij} = \frac{1}{\sum_{k=1}^{N} W_{ik}} \sum_{k=1}^{N} W_{ik} f_{kj}$$

for all $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, K$.

Label propagation can be considered as a constrained mincut problem, which is a very popular clustering technique [3]. Since graph mincut can be written as $(1/4) \cdot \mathbf{f}'\mathbf{L}\mathbf{f}$, where $\mathbf{f} \in \{-1, 1\}^N$, therefore label propagation is equivalent to searching for a mincut of the data graph, given that the labeled points are fixed.

4. Semi-supervised kNN

The k-nearest neighbor algorithm determines labels based on the labels of the nearest points. "Nearest" is defined using some metric, in the original formulation taking the Euclidean metric. The Euclidean distance can be rewritten in form of dot products as

$$\begin{aligned} \|\mathbf{x} - \mathbf{z}\|_2^2 &= \langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{z}, \mathbf{z} \rangle - 2 \cdot \langle \mathbf{x}, \mathbf{z} \rangle \\ &= k(\mathbf{x}, \mathbf{x}) + k(\mathbf{z}, \mathbf{z}) - 2 \cdot k(\mathbf{x}, \mathbf{z}) \end{aligned}$$

where $k(\cdot, \cdot)$ denotes in this case the linear kernel $k(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle = \mathbf{x}'\mathbf{z}$. Using the kernel trick, this can be replaced by any other positive semi-definite kernel. Thus the points are implicitly mapped to a – possibly higher dimensional – space, where their dot product is given by the kernel function $k(\cdot, \cdot)$.

In multi-label learning let us denote the decision function as $f : X \to [0, 1]^K$. For hard classification we can set a threshold, for example 0.5, but for soft classification we use the values of the output vector as class membership probabilities. The decision function is expressed in the same way as in the case of binary classification,

$$f(\mathbf{x}) = \frac{1}{\sum_{\mathbf{z} \in N_k(\mathbf{x})} W_{\mathbf{z}\mathbf{x}}} \sum_{\mathbf{z} \in N_k(\mathbf{x})} W_{\mathbf{z}\mathbf{x}} f(\mathbf{z})$$

with the difference that now the output is a $K \times 1$ vector, instead of a scalar value.

4.1. **Data-dependent kernels.** There are other methods to determine the nearest neighbors of a point by using the following data-dependent kernels.

4.1.1. *The ISOMAP kernel.* ISOMAP (ISOmetric feature MAPping) was introduced in [20] for dimensionality reduction using the manifold assumption. The ISOMAP kernel is defined as

$$\mathbf{K}_{\text{isomap}} = -(1/2)\mathbf{J}\mathbf{G}^{(2)}\mathbf{J}$$

where $\mathbf{G}^{(2)}$ contains the squared graph distances (shortest paths in the graph whose vertices are the original data points and the edges are among the nearest neighbors of each point) and $\mathbf{J}$ is the centering matrix, $\mathbf{J} = \mathbf{I} - \frac{1}{N} \cdot \mathbf{1} \cdot \mathbf{1}'$, $\mathbf{I}$ is the identity matrix, and $\mathbf{1}$ is the $N \times 1$ vector of 1's. $\mathbf{G}^{(2)}$ is not necessarily positive semi-definite so neither is the ISOMAP kernel. But since only the largest eigenvalues and the corresponding eigenvectors are important, we proceed in the following way. The kernel matrix can be decomposed into $\mathbf{USU}'$, where $\mathbf{U}$ contains the eigenvectors, while the diagonal matrix $\mathbf{S}$ holds the eigenvalues of the decomposed matrix [15, p. 393]. Then the ISOMAP kernel we will use is $\mathbf{K}_{\text{isomap}} = \mathbf{U}\widetilde{\mathbf{S}}\mathbf{U}'$, where $\widetilde{\mathbf{S}}$ is the diagonal matrix of the eigenvalues in which each negative eigenvalue was set to zero.

Informally, the ISOMAP kernel maps the points to the space, where their pointwise distances equal to the shortest path distances on the data graph in the input space. If the points are centered at each dimension, then $-(1/2) \cdot \mathbf{J}\mathbf{G}^{(2)}\mathbf{J}$ is equal to the dot products of the vectors mapped to the above-mentioned space [5, p. 262].

4.1.2. *The multi-type cluser kernel.* In [8] the authors develop a cluster kernel which connects several techniques together like spectral clustering, kernel PCA and random walks. The proposed cluster kernel is built following the steps described below:

(1) Compute the Gaussian kernel and store in matrix $\mathbf{W}$.
(2) Symmetrically normalize $\mathbf{W}$, that is let $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, where $\mathbf{D} = \text{diag}(\mathbf{W} \cdot \mathbf{1})$, and compute its eigendecomposition, $\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}'$.
(3) Determine a transfer function $\varphi(\cdot)$ for transforming the eigenvalues, $\widetilde{\lambda}_i = \varphi(\lambda_i)$, and construct $\widetilde{\mathbf{L}} = \mathbf{U}\widetilde{\mathbf{\Sigma}}\mathbf{U}'$, where $\widetilde{\mathbf{\Sigma}}$ contains the transformed eigenvalues on the diagonal.
(4) Let $\widetilde{\mathbf{D}}$ be a diagonal matrix with diagonal elements $D_{ii} = 1/\widetilde{L}_{ii}$, and compute $\mathbf{K} = \widetilde{\mathbf{D}}^{1/2}\widetilde{\mathbf{L}}\widetilde{\mathbf{D}}^{1/2}$.

The kernel type depends on the chosen transfer function. We discuss here three types of transfer functions as in [8]. In the following let λ_i represent the eigenvalues of matrix $\mathbf{L}$ defined in step (2).

The step transfer function is defined as $\varphi(\lambda_i) = 1$ if $\lambda_i \geq \lambda_{\text{cut}}$ and 0 otherwise, where λ_{cut} is a predetermined cutting threshold for the eigenvalues. This results in the dot product matrix of the points in the spectral clustering representation [17].

The linear step transfer function simply cuts off the eigenvalues which are smaller that a predetermined threshold, $\varphi(\lambda_i) = \lambda_i$ if $\lambda_i \geq \lambda_{\text{cut}}$, otherwise equals 0. Without normalization, that is with $\mathbf{D} = \mathbf{I}$ and similarly $\widetilde{\mathbf{D}} = \mathbf{I}$, the method would be equal to the data representation in KPCA space [18], since in that case we simply cut off the least significant directions to obtain a low rank representation of $\mathbf{L}$.

The polynomial transfer function is defined as $\varphi(\lambda_i) = \lambda_i^t$, where $t \in \mathbb{N}$ or $t \in \mathbb{R}$ is a parameter. Thus the final kernel can be written as

$$\widetilde{\mathbf{K}} = \widetilde{\mathbf{D}}^{1/2}\mathbf{D}^{1/2}\left(\mathbf{D}^{-1}\mathbf{W}\right)^t\mathbf{D}^{-1/2}\widetilde{\mathbf{D}}^{1/2} \tag{3}$$

where $\mathbf{D}^{-1}\mathbf{W} = \mathbf{P}$ is the probability transition matrix, where P_{ij} is the probability of going from point i to point j. This is called the random walk kernel, since (3) can be considered as a symmetrized version of the transition probability matrix $\mathbf{P}$.

4.1.3. *The hierarchical cluster kernel.* Hierarchical cluster kernels for supervised and semi-supervised learning were introduced in [4]. We used hierarchical clustering techniques to build ultrametric trees [21]. Then we used the distances induced by the clustering method to build a kernel for supervised and semi-supervised methods. The hierarchical cluster kernels are generalizations of the connectivity kernel [14].

The algorithm has the following steps:

-2. Determine the k nearest neighbors or an ϵ-neighborhood of each point and take all the distances to other points equal to zero.
-1. Compute shortest paths for every pair of points – using for example Dijkstra's algorithm.
0. Use these distances in clustering for the pointwise distance $d(\cdot,\cdot)$ in single, complete and average linkages distances [16, Chapter 3], [13, Chapter 4].
1. Perform an agglomerative clustering on the labeled and unlabeled data using one of the above-mentioned linkage distances.
2. Define matrix $\mathbf{M}$ with entries M_{ij} = linkage distance in the resulting ultrametric tree at the lowest common subsumer of i and j; $M_{ii} = 0$, $\forall i$.
3. Define the kernel matrix as $\mathbf{K} = -\frac{1}{2}\mathbf{JMJ}$.

method	**accuracy**
kNN (linear, Gaussian)	94.00 ($k_{kNN} = 1$)
LP (linear, Gaussian)	80.29 ($1/(2 \cdot \sigma^2) = 0.05$)
kNN + ISOMAP	95.71 ($k_{isomap} = 5, k_{kNN} = 5$)
kNN + mt. cluster kernel	95.00 (linstep, $1/(2 \cdot \sigma^2) = 0.05$, $\lambda_{cut} = 0.1$, $k_{kNN} = 4$)
kNN + h. cluster kernel	**96.64** (average linkage, $k_{isomap} = 4$, $k_{kNN} = 3$)

TABLE 1. Accuracy results obtained for the modified USPS handwritten digits data set.

The first three steps of the method – numbered with -2, -1 and 0 – are optional; they can be applied if the semi-supervised manifold assumption is expected to hold.

5. EXPERIMENTS

In the experiments we compared the methods of kNN, kNN with data-dependent kernels and label propagation. We used the data-dependent kernels presented in the previous section: the ISOMAP, the multi-type and the hierarchical clusters kernels. The methods were tested on two data sets: a modified version of the USPS *(United States Postal Service)* handwritten digits data set and a subset of Reuters-21578 [12]. The USPS data set is derived from the original USPS set of handwritten digits[1]. The set is imbalanced, since it was created by putting the digits 2 and 5 into one class, while the rest is in the second class. 150 images belong to each of the ten digits. Because the set was used as a benchmark data set for the algorithms presented in the book [7], it was obscured using a simple algorithm to prevent recognizing the origin of the data. The set contains 1500 examples and 2×12 splits of the data, where the first 12 splits contain 10 labeled, and 1490 unlabeled, while the second 12 splits contain 100 labeled and 1400 unlabeled examples. We used only the first split with 100 labeled points[2].

[1] `http://archive.ics.uci.edu/ml/datasets/`

[2] The modified USPS data set can de downloaded from `http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html`

method	**microBEP/macroBEP**
kNN (linear)	89.60 / 89.22 ($k_{kNN} = 14$)
kNN (Gaussian)	90.05 / 90.02 ($k_{kNN} = 17$)
LP (linear)	88.44 / 88.39
LP (Gaussian)	89.81 / 90.33 ($1/(2 \cdot \sigma^2) = 1$)
kNN + ISOMAP	88.37 / 87.99 ($k_{isomap} = 16, k_{kNN} = 16$)
kNN + mt. cluster kernel	**91.12 / 91.04** (linstep, $1/(2 \cdot \sigma^2) = 1$, $\lambda_{cut} = 0.01$, $k_{kNN} = 16$)
kNN + h. cluster kernel	87.46 / 87.42 (average linkage, $k_{isomap} = 3$, $k_{kNN} = 7$)

TABLE 2. Micro- and macro-averaged precision–recall breakeven point results for the modified Reuters-21578 text categorization corpus.

We also modified the Reuters-21578 text categorization corpus in order to make it smaller and to balance the categories. The original corpus[3] contains 12 902 documents – 9603 for training and 3299 for testing – categorized into 90 classes. We kept the following 10 categories: alum, barley, bop, carcass, cocoa, coffee, copper, cotton, cpi, dlr. Thus we were left with 626 training and 229 test documents. For representing documents we used the bag-of-words document representation [2, Chapter 2] with tfidf weighting [2, p. 29]. We stemmed the words of the documents using the Porter stemmer, and selected 500 terms with the χ^2 feature selection technique [22, 19].

For evaluation we used accuracy for the USPS data set and precision–recall breakeven point for the Reuters corpus.

The results are shown on Tables 1 and 2. For each method we searched for the parameters that result in the best performance on the test data (these parameters are shown in brackets). The best results for each data set were formatted with boldface.

[3]The 90 and 115-categories version of Reuters can be downloaded from the homepage of Alessandro Moschitti, `http://dit.unitn.it/~moschitt/corpora.htm`

6. Discussion

In this paper we compared kNN and some semi-supervised kNN methods on two data sets. We saw that semi-supervised kNN methods can outperform conventional kNN: for the USPS data set we obtained an improvement of 2.64% with the average linkage hierarchical kernel. However label propagation showed a very low performance on this data set. We considered the values between 1 and 5 for k, which means that the classes are separated quite well. For the Reuters corpus we found the multi-type cluster kernel with linear step function to provide the best performance, but the improvement was not so significant as for the USPS data set. This also shows that KPCA is able to remove irrelevant dimensions from the bag-of-words representation of the Reuters corpus. Label propagation with Gaussian kernel showed only little performance improvement to linear kNN. We see however that the best values of k for kNN were between 7 and 17, which could imply the intertanglement of the documents, that is one should search for a better initial representation than bag-of-words.

Acknowledgement

We acknowledge the support of the grants CNCSIS/TD-35 and CNCSIS/TD-77 by the Romanian Ministry of Education and Research.

References

[1] A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] Tijl De Bie. *Semi-Supervised Learning Based On Kernel Methods And Graph Cut Algorithms*. PhD thesis, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, 3001 Leuven (Heverlee), 2005.

[4] Zalán Bodó. Hierarchical cluster kernels for supervised and semi-supervised learning. In *Proceedings of the 4nd International Conference on Intelligent Computer Communication and Processing*, pages 9–16. IEEE, August 2008.

[5] Ingwer Borg and Patrick J. F. Groenen. *Modern multidimensional scaling, 2nd edition*. Springer-Verlag, New York, 2005.

[6] B. E. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *Computational Learning Theory*, 5:144–152, 1992.

[7] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. MIT Press, September 2006. Web page: http://www.kyb.tuebingen.mpg.de/ssl-book/.

[8] Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. Cluster kernels for semi-supervised learning. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 585–592. MIT Press, 2002.

[9] Chung. Spectral graph theory (reprinted with corrections). In *CBMS: Conference Board of the Mathematical Sciences, Regional Conference Series*, 1997.

[10] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, IT-13, 1967.
[11] Fabio G. Cozman and Ira Cohen. Risks of semi-supervised learning. In Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors, *Semi-Supervised Learning*, chapter 4, pages 55–70. MIT Press, 2006.
[12] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, 56:971–974, 2004.
[13] Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. John Wiley and Sons, 2001. 0-471-05669-3.
[14] Bernd Fischer, Volker Roth, and Joachim M. Buhmann. Clustering with the connectivity kernel. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *NIPS*. MIT Press, 2003.
[15] Gene H. Golub and Charles F. Van Loan. *Matrix Computations, 3nd Edition*. The Johns Hopkins University Press, Baltimore, MD, 1996.
[16] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.
[17] Andrew Y. Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.
[18] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352, 1999.
[19] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
[20] 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.
[21] Bang Ye Wu and Kun-Mao Chao. *Spanning Trees and Optimization Problems*. Chapman and Hall/CRC, Boca Raton, Florida, 2004.
[22] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *International Conference on Machine Learning*, pages 412–420, 1997.
[23] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

Department of Computer Science, Babeş–Bolyai University, Mihail Kogălniceanu nr. 1, RO-400084 Cluj-Napoca
E-mail address: {zbodo,minier}@cs.ubbcluj.ro