Semi-Supervised Methods for Handwritten Character Recognition using Active Learning

Leonidas Lefakis [a] Marco A. Wiering [a]

[a] *Department of Information and Computing Sciences, Utrecht University*
P.O.Box 80.089 3508TB Utrecht

Abstract

There are a number of supervised machine learning methods such as classifiers pretrained using restricted Boltzmann machines and convolutional networks that work very well for handwritten character recognition. However, they require a large amount of labeled training data to achieve good performance which unlike unlabeled data is often expensive to obtain. In this paper a number of novel semi-supervised learning methods for handwritten character recognition are presented based on the previous algorithms. These methods are oriented towards learning from as little labeled data as possible and for this goal they use unlabeled data and active learning. The proposed techniques are of varying complexity and involve simple K-means clustering, feature mapping with self organizing maps, dimensionality reduction with deep auto-encoders, and sub-sampling techniques. The presented algorithms outperform both a generic semi-supervised active learning algorithm and two well known supervised algorithms.

1 Introduction

A number of supervised learning methods exist for optical character recognition and handwritten text recognition [12]. Two such algorithms which attain a very high performance are convolutional networks [8] and neural networks pretrained by restricted Boltzmann machines (RBM) and fine-tuned using conjugate gradient learning [7]. Both these approaches reach an accuracy of close to 99% on the MNIST handwritten digit dataset. However, both supervised methods also require huge amounts of labeled data to achieve this performance due to their complexity. As is the case in most applications labeled data, though extremely useful, cannot always be easily obtained; the labels must usually be assigned manually, a costly and time-consuming process. Unfortunately this is also true in handwritten text recognition, labeled data is hard to come by, but fortunately there exists a large reserve of unlabeled data. This paucity of labeled data in conjunction with the abundance of unlabeled data leads us to seek methods that exploit both kinds of data sets. To this end we find it useful to re-examine the aforementioned supervised algorithms and to devise semi-supervised counterparts to these approaches.

Although there are many possible semi-supervised learning methods, they usually use a-priori labeled data that has been randomly selected to be labeled and which is not particularly significant with regards to the application. It would instead be more useful if we could obtain the labels of those data points which would aid the algorithm the most to achieve high performance. This can be accomplished using active learning [4]. Instead of randomly labeling a certain amount of data, a human expert is instructed by the algorithm to label specific data instances which it deems are the most important to solving the problem at hand.

As can be seen, both active and semi-supervised methods are fitting approaches to solving handwritten character recognition problems. The algorithms presented here draw from both these approaches, obtaining a significantly higher performance than supervised passive methods when the amount of labeled data is minimal, they also achieve significantly higher performance when compared with a simple semi-supervised active learning algorithm used as a benchmark. Furthermore these algorithms do not need any amount of originally labeled data; thus all labeled data is acquired from the active learning process and the need for randomly labeled original data is avoided.

Outline of this paper. Section 2 presents the supervised learning methods on which the semi-supervised methods are based. Section 3 describes the novel semi-supervised methods that we have developed. Section 4 describes experimental results, and Section 5 concludes this paper.

2 Supervised Methods

2.1 Convolutional Networks

Convolutional networks [8] use two basic techniques to create a distortion invariant classifier. They use feature detectors in order to create feature mappings of the data; these detectors consist of a set of weights that are applied to a group of units (the receptive field) in a small neighborhood in the data. By applying a variety of feature detectors to a series of receptive fields in the data a feature mapping of the data is obtained.

The other technique used by convolutional networks is that of sub-sampling. Feature maps record the presence and position of a certain feature. However the exact position of a specific feature is not always important. Convolutional networks employ sub-sampling which consists of averaging over the data of the feature maps belonging to the previous layer and multiplying the result by a training coefficient. As is the case with feature mappings, sub-sampling also uses weight replication. This leads to a reduction in the resolution of the feature maps and the network is thus less sensitive to shifts and distortions.

Typically a convolutional network will have a number of layers, with feature mapping and sub-sampling alternating between layers.

2.2 Pretrained Networks using RBM

Hinton and Salakhutdinov [7] have shown that the performance of a neural network can be significantly increased if the network's weights are initially pretrained using restricted Boltzmann machines (RBMs) and subsequently fine-tuned by conjugate gradient learning.

By pretraining the weights, a neural network (with many hidden layers) is able to overcome the initial weight problem. Networks of this kind will typically require initial weights close to an acceptable solution in order to attain the desirable performance and avoid local minima. Pretraining the weights ensures that the network will be trained using suitable initial weights.

As previously mentioned, this pretraining can be accomplished using restricted Boltzmann machines; each pair of consecutive layers in the network are treated as an RBM. The weights between each pair of consecutive layers are pretrained by training the corresponding RBM; the available data is presented to the RBM and the machine's weights are altered to decrease the energy of the network regarding the presented data. Once the RBM has been presented with the training data, a number of confabulations are produced and presented to the RBM and its weights are altered accordingly so that the energies of the confabulations are increased. The RBM thus learns to show a preference to real data as opposed to the confabulations.

This method is repeated for the weights between all the layers of the network. Once the network has been pretrained, the training data is used to train the entire network using the conjugate gradient method [11], this step is referred to as fine-tuning the network.

3 Semi-supervised Methods

3.1 Simplified Generic Algorithm

It is fairly straightforward to create a semi-supervised algorithm which uses active learning, using the simple k-means clustering algorithm and a k-nearest neighbor (k-NN) classifier. The initial data, which is unlabeled, is used to create a partitioning of the data by clustering with the k-means algorithm. In the case presented here the Euclidean distance has been used as a distance function but obviously any of a number of such functions may be used.

Using the clusters obtained, we then find those data points which are closest to each of the cluster centers. If the clusters obtained are meaningful with regards to the application then clearly so will these data points have an increased significance as compared to the other data points. After having found these data points, the algorithm presents them to the human expert requesting their labels.

Once these labels are acquired, the algorithm utilizes a simple 1-NN classifier to assign labels to the remaining data. Despite the relative simplicity of the algorithm, it performs considerably better than the more complicated supervised methods described above for applications with minimal labeled data due to its use of both semi-supervised and active learning. It remains nonetheless a rather simplified algorithm as will be seen in the experimental results. However, by using this simple algorithm as a basis and combining it with the techniques proposed in the following, it is possible to create semi-supervised classifiers using active learning which attain significantly higher performance.

3.2 Semi-supervised Learning using Auto-encoders

The method presented in the previous subsection can be considerably improved on with the help of auto-encoders. Auto-encoders [7] can also utilize pretraining by restricted Boltzmann machines to create a pre-trained bottlenecked network. The bottlenecked layer of the network is used to extract a noise-free representation of the data consisting of fewer dimensions than the original.

More specifically a symmetric bottlenecked neural network is trained so that its output is identical to the input it is presented with; each layer has the same amount of neurons as its mirror layer on the other side of the bottleneck. The bottleneck layer itself consists of as many neurons as the dimensionality of the data representation which we wish to acquire. Before the actual training of the network, the weights between the layers are pretrained using RBMs as analyzed in subsection 2.2.

Once the pretraining phase has finished and the weights have been initialized, the network is fine-tuned using the conjugate gradient method. For each data point presented to the network the desirable output is set to be equal to the input. As the network is able to recreate the data in the output from the values of the neurons in the bottleneck, by obtaining these values for each data point presented to the network we effectively acquire a noise-free representation of the data of reduced dimensionality.

In order to combine auto-encoders with semi-supervised and active learning, we originally train an auto-encoder using the available (unlabeled) training data. Once the auto-encoder is trained we obtain via the bottleneck neurons a new representation of the data with reduced dimensions. This data is then clustered using k-means as in the previous subsection. Having clustered the data the algorithm then finds the data points closest to the cluster centers, presents them for labeling to the expert and finally uses a 1-NN classifier to label the remaining data in the same manner as before.

3.3 A Convolutional Network Analogous

Although, as shown in the previous subsection, a semi-supervised algorithm can be derived in a relatively straightforward fashion from networks pretrained by RBMs, this is not the case when attempting to create a semi-supervised analogous of convolutional networks. In order to achieve this we must first redefine both the techniques of feature mapping and sub-sampling so that they may be used in an unsupervised or semi-supervised manner. In the following we will present three techniques, one for feature mapping and two for sub-sampling that will be used in the semi-supervised algorithm.

After partitioning the data based on the similarity function and using sub-sampling, the closest data points to each cluster center are once more chosen and presented to the expert for labeling. Once the labels of these data points are acquired, they are used to label the remainder of the data using a 1-NN classifier.

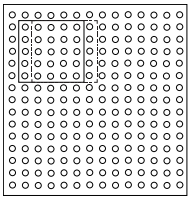
3.3.1 Feature Mapping

In order to create a feature mapping of the data, exploiting the available unlabeled data, we create feature detectors that can manipulate this unlabeled data. For this purpose we utilize a self organizing map (SOM) that will be trained using all available data to recognize the presence of certain features and thus serve as a detector of features. This self organizing map takes a small number of input units which are situated in a small neighborhood (i.e. the SOM's receptive field), and according to which feature is present in that field, activates the corresponding output unit.

The SOM is originally trained using as input neighborhoods of units in the training data. In the case of handwritten digits, these neighborhoods are in fact neighborhoods of pixels in different parts of the image. In the experiments presented below, the SOM used as a features detector has a 5×5 input layer . Presenting the SOM with 5×5 neighborhoods of units fields taken from the data, it is effectively trained to recognize the different features that occur in these 5×5 windows throughout the data. Consequently each neuron in the SOM's output layer becomes sensitive to a different feature.

Having trained the SOM, we subsequently scan each data point (image) using this SOM in order to obtain a feature mapping. For each data point the SOM is applied to a number of receptive fields on the data point, each receptive field shifted slightly in relation to the previous by one column or row, thus two consecutive receptive fields are overlapping. Figure 1(A) shows two such consecutive receptive fields, the receptive fields here are represented by the square windows which encompass the 5×5 neighborhoods of units. The SOM is applied continuously to consecutive receptive fields until the entire data point is scanned.

By scanning each data point with the SOM, a feature mapping of the data point is created by recording for each receptive field which neuron in the SOM's output field is activated. Thus the feature mapping of the



1	4	7	12	11	9	3	5	6	1
5	9	1	4	10	8	7	1	3	12
8	9	3	5	4	1	11	7	2	1
10	11	4	7	1	2	5	9	3	1
1	12	1	7	11	4	1	3	1	9
3	1	2	5	7	1	9	10	4	1
1	9	1	2	1	4	7	1	3	1
5	1	9	1	12	1	3	2	7	4
1	11	5	4	1	7	9	1	3	1
1	1	8	4	3	2	9	11	5	7

Figure 1: (A) Two consecutive receptive fields. (B) A feature mapping.

data consists of a list of numbers as can be seen in Figure 1(B) where each of these numbers corresponds to a specific neuron in the SOM's output field and denotes the presence of a specific feature in the corresponding receptive field.

3.3.2 Sub-sampling Technique A

Once a feature map of the data has been created, it is useful to find a way to execute sub-sampling. By using sub-sampling the algorithm will become even less sensitive to small distortions in the data as the exact positions of the occurring features will no longer be as important as their approximate positions.

To this effect, after creating the feature maps of the data, these feature maps are clustered in K different clusters, where the number of clusters K is equal to the number of clusters to which the data will be ultimately partitioned . For each one of these clusters k we follow the following procedure :

1. A specific region (neighborhood) of units is extracted from the data points that are partitioned to that specific cluster (In the experiments presented here 2×2 neighborhoods are chosen).

2. The center of the cluster formed by these subdimensions is calculated. This center is treated as an archetype for that subregion in the specific cluster k .

3. The above steps 1-2 are repeated scanning all subregions. The subregions are chosen in such a way that they do not overlap but rather are adjacent to each other. By this repetition we obtain a series of archetypal subregions. This collection of archetypal subregions together form an archetypal data point which is considered representative of the specific cluster.

4. Steps 1-3 are repeated for every cluster k.

5. Finally for each data point in the data set, we compare each of its 2×2 subregions with the respective archetypes obtained for each of the clusters. Each subregion of the data point is then replaced with the archetype with which it has the greatest similarity. Thus if a specific subregion in a data point is found to be most similar to the respective subregion in the archetypal data point of cluster k_i then it is replaced by that archetypal subregion. Another subregion of that data point may be found most similar to the archetypal subregion of another cluster.

The above procedure ultimately replaces the subregions of each data point by a number of archetypal subregions belonging to different clusters. Thus slight shifts or distortions in the original feature maps are lost as subregions exhibiting similar features will most likely be clustered together, and ultimately be replaced by a common archetype.

3.3.3 Sub-sampling Technique B

Another way to sub-sample the created feature maps in a way that takes advantage of all the available data is a method that resembles bagging for clustering [6]. However, although in bagging methods clustering is executed at each step over a different bootstrap sample, in the sub-sampling method presented here clustering is executed over different subdimensions of the data at each step. Thus the b^{th} time the partitioning clustering procedure is applied, it is applied over the entire learning set but only over specific subdimensions of the data. The rest of the data's dimensions are not taken into account by the specific clustering procedure.

Analytically the algorithm proposed is presented below.

1. The data is clustered using specific subdimensions , which differ at each iteration, are extracted from the data to create the learning set L^b, where each observation I_j^b belonging to L^b consists of the specific subdimensions of the data point I_j. These subdimensions are chosen to be a neighborhood of units encompassed by a window of specific size .

2. The partitioning clustering procedure P is applied to the training data (of the appropriate subdimensions) and the cluster labels $P^b(I_j^b; L^b)$ are obtained for each image I_j.

3. The cluster labels assigned to the data are permuted in order to create a maximum overlap with the original clustering of the data ($P^1(I_j^1, L^1)$)). More specifically the permutation t^b is chosen such that it maximizes (where I is the indicator function):

$$\sum_{j=1}^{N} I(t(P^b(I_j^b; L^b)) = P^1(I_j^1, L^1))$$

4. The procedure is repeated N times (where N is chosen by the user) and finally each observation is assigned a cluster label by majority voting.

Figure 2 shows the subdimensions used for clustering in two different clustering steps. The data point has already been scanned by a SOM and is represented by a feature map.

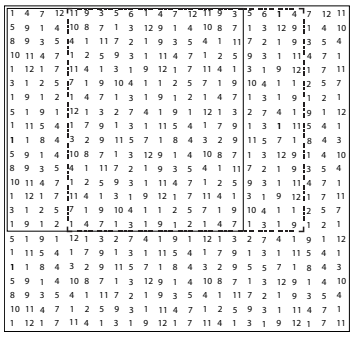


Figure 2: Two different sets of dimensions used for clustering.

Each of the clustering steps executed leads to a sub-sampling of the data. Data points with similar features present in approximately the same receptive fields will be more likely to be partitioned together by each of the clustering steps. Thus after the majority voting is carried out it is highly probable that these data points will be assigned to the same cluster label. By using this method, the significance of the exact position of each feature in the feature maps is reduced, giving more weight to the approximate position.

3.3.4 Clustering over Feature Maps

Once a feature mapping of the data has been created and sub-sampling has been executed, the data must be clustered. However as the data is represented by a list of numbers denoting the presence of features and their locations, the Euclidean distance function is no longer an appropriate similarity function for the clustering procedure. Instead the data is partitioned using a similarity function based on the distances between neurons in the SOM's output layer.

More specifically the distance between two data points I_i and I_j is set to be the sum of the distances between the pair of neurons activated in the SOM's outputs layer for each receptive fields in the SOM. More formally the distance between these two data points is calculated as

$$dist(I_i, I_j) = \sum_{k=1}^{M} d_{SOM}(I_i(k), I_j(k))$$

where $I_i(k), I_j(k)$ denote the neurons activated by the k^{th} neighborhood of units of the two data points and where the distance function $d_{SOM}(x, y)$ returns the distance between neurons x and y in the SOM.

4 Experiments and Results

4.1 Experimental Setup

The methods presented above are used for handwritten digit recognition using the MNIST dataset [9]. This dataset consists of sixty thousand images of handwritten digits (0 to 9). Due to the fact that clustering can be very time-consuming, a set of three thousands digits is drawn using a uniform distribution from the data to be used as a training set and the algorithms are tested on a number of test sets, each consisting of five hundred digits and drawn independently from the remaining data. Before using the images the values of their pixels are normalized so that these values fall within the range [0,1]. Each of the algorithms is trained six times in order to address a certain degree of randomness due to the choice of initial cluster centers which are chosen randomly from the training data. Each trained classifier is tested on the ten mentioned test sets.

In the case of the semi-supervised scheme using auto-encoders, a nine layered neural network is used to encode the data. This neural network has 784-1000-500-250-30-250-500-1000-784 neurons in the respective layers, with thirty being the number of neurons in the bottleneck and consequently the number of dimensions of the encoded data. A neural network of the same architecture was used in [7].

For the feature mapping in the case of the convolutional network analogous algorithm, a 4×3 two dimensional self organizing map was used for feature detecting. Lenet-5 a convolutional network presented in [8] uses six feature detectors to detect the presence of six different features. However, experiments run using different sizes of self organizing maps, showed that the best performance is attained using a 4×3 SOM. The topology of the SOM used was chosen to be hexagonal though no systematic experimentation was done in this direction; some preliminary experiments seem to indicate that the SOM's topology has little bearing on the algorithm's performance. The input layer of the SOM was set, as stated above, to be of size 5×5. Sub-sampling using technique A was done in 2×2 subregions as mentioned; these subregions were chosen so that two consecutive subregions were adjacent and not overlapping. Finally for the sub-sampling technique B nine separate sixteen by sixteen windows were chosen.

4.2 Results

Each of the proposed methods uses clustering of some form to obtain those data points which will be presented to the human expert. Thus the amount of data which will be labeled is determined according to the number of clusters the data is partitioned into, one for every cluster; thus when using 100 clusters to ultimately classify the digits to the ten classes, 100 data points will be labeled.

The algorithms presented in this paper, i.e. the generic semi-supervised classifier and the proposed classifiers, all make use of a k-NN classifier with k=1, experiments were run using higher values of k which did not yield a great increase in performance despite the fact that based on the methods used here, larger values of k meant a higher number of labeled data.

In Table 1 we can see the performance of the proposed algorithms when using one hundred clusters. The performance of a supervised method, pretrained neural networks using RBM, are also shown for comparison, the network was trained using one hundred digits. Experiments run using convolutional networks showed that they were not able to learn using such a small amount of data, after a number of epochs no increase in performance was observed. For comparison experiments were also conducted using a 1-NN classifier.

Table 1: Accuracy and standard deviation for 100 labeled examples

	Accuracy	Stand. dev.
Convolutional Network	11.23%	
Pretrained Network	66.40%	
1-NN Classifier	73.18%	
Generic Semi-supervised Algorithm	82.55%	1.52%
Using Auto-encoders	86.01%	1.54%
Using Feature Mapping	85.25%	1.37%
Using Feature Mapping and Sub-sampling A	85.93%	0.93%
Using Feature Mapping and Sub-sampling B	84.47%	1.83%

As can be seen the semi-supervised methods perform substantially better than the supervised methods. This is of course expected as the supervised methods take advantage of neither the unlabeled data nor the

active learning process. More importantly though it can be seen that by combining the proposed techniques with the simple generic algorithm, classifiers with higher performance are acquired. In Figure 3 the performances of two of the proposed algorithms as well as that of the generic algorithm are plotted against the number of clusters used in the clustering step. The performances shown are the performance of the simple generic algorithm, the performance when utilizing auto-encoders and the performance when using feature mapping without sub-sampling.

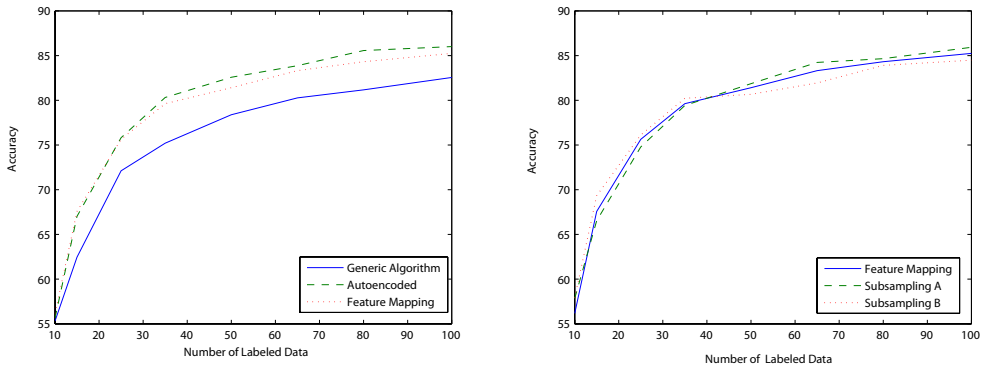


Figure 3: Performance according to the number of clusters

In Figure 3 we also see the performance of the semi-supervised algorithm when the two sub-sampling techniques are used. Again the performance is plotted against the number of clusters used. The performance using only feature mapping is also shown for comparison.

5 Related Work

In the past research has been conducted into applying semi-supervised learning as well as active learning to handwritten digit recognition. In [5] transductive support vector machines are used on the MNIST data set, the authors report an error rate of 16.81% when using 100 labeled examples and 2000 unlabeled examples, slightly fewer than the number of unlabeled examples used here. In [1] semi-supervised learning on manifolds is used, again on the MNIST data set with similar results.

As stated research has also been conducted into combining semi-supervised learning and active learning for handwritten digit recognition. In [2] LPBoost is combined with mixture kernels and used on the MNIST data set, the resulting classifier however is only used to distinguish between odd and even digits. Other approaches use clustering, as we have here, to obtain appropriate examples to be labeled. In [13] and [3] SVMs are combined with clustering, both these approaches apply clustering after first having trained an SVM.

Preclustering is used in [10] where the data is originally clustered using soft k-medoids clustering, subsequently data points are chosen either close to the cluster centers or close to the classification boundaries, to be labeled. This selection is made so that the expected future error of classification is minimized. The resulting algorithm is tested on the MNIST dataset, attaining high accuracy. The task was to distinguish between a given digit and the rest, which is simpler than digit recognition in general.

6 Conclusions

This paper presented a number of semi-supervised learning algorithms which were combined with active learning and which were shown to learn from unlabeled data to improve learning accuracies. Even the generic algorithm performed much better than supervised methods that did not learn from unlabeled data, and by combining this with a variety of techniques the proposed algorithms increased performance further.

Of the semi-supervised methods presented, as can be seen from Figure 3, the simple generic algorithm performs the worst as expected. The method using auto-encoders inspired by [7] performs better than the

simple feature mapping method inspired by [8], this difference of performance was found to be statistically significant as was the difference of performance of both these classifiers from the simple generic classifier.

As mentioned the experiment were conducted on only a subset of the MNIST dataset due to the computational cost of the algorithms. This computational cost is due almost exclusively to the clustering step of the classifiers, either using k-means or clustering over feature maps (which is in fact a form of the k-means algorithm which uses a specific distance function), the complexity of which is $O(nkdi)$ where n is the number of data points, d the number of dimensions, and i the number of iterations. Thus the scalability of the classifiers presented here depends almost exclusively on the scalability of the k-means algorithm.

In future work we want to apply these algorithms on other problems such as text categorization.

References

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

[2] Jinbo Bi, Glenn Fung, Murat Dundar, and Bharat Rao. Semi-supervised mixture of kernels via lpboost methods. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 569–572, Washington, DC, USA, 2005. IEEE Computer Society.

[3] Xuehua Shen Chengxiang. Active feedback – uiuc trec-2003 hard experiments.

[4] D. A. Cohn. Neural network exploration using optimal experiment design. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 679–686. San Mateo, CA: Morgan Kaufmann, 1994.

[5] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Large scale transductive SVMs. *Journal of Machine Learning Research*, 7:1687–1712, 2006.

[6] S. Dudoit and J. Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19, 2003.

[7] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE 1st International Conference on Neural Networks*, volume 86(11), pages 2278–2324, 1998.

[9] Y. LeCun, L.D. Jackel, L. Bottou, C. Cortes, J.S. Denker, J. Drucker, I. Guyon, U.A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks: The Statistical Mechanics Perspective*, pages 261–276, 1995.

[10] Hieu T. Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 79, New York, NY, USA, 2004. ACM Press.

[11] J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[12] A. Vinciarelli. Offline cursive handwriting: From word to text recognition. Technical Report IDIAP-RR 03-24, Dalle Molle Institute for Perceptual Artificial Intelligence, 2003.

[13] Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. Representative sampling for text classification using support vector machines. In *ECIR*, pages 393–407, 2003.