Large Scale Manifold Transduction

Michael Karlen[*†] MICHAEL.KARLEN@GMAIL.COM
Jason Weston[*] JASONW@NEC-LABS.COM
Ayse Erkan[*‡] NAZ@CS.NYU.EDU
Ronan Collobert[*] COLLOBER@NEC-LABS.COM
(∗) NEC Labs America, 4 Independence Way, Princeton, NJ 08540 USA
(†) Ecóle Polytechnique Fédérale de Lausanne, CH-1015, Lausanne, Switzerland
(‡) New York University, Computer Science Department, 715 Broadway New York, NY 10003 USA

Abstract

We show how the regularizer of Transductive Support Vector Machines (TSVM) can be trained by stochastic gradient descent for linear models and multi-layer architectures. The resulting methods can be trained *online*, have vastly superior training and testing speed to existing TSVM algorithms, can encode prior knowledge in the network architecture, and obtain competitive error rates. We then go on to propose a natural generalization of the TSVM loss function that takes into account neighborhood and manifold information directly, unifying the two-stage Low Density Separation method into a single criterion, and leading to state-of-the-art results.

1. Introduction

Several methods for improving discriminative classifiers using unlabeled data have been developed in the last few years. Perhaps the two most popular ways of utilizing the unlabeled data are:

(i) maximizing the *margin* on the unlabeled data as in Transductive Support Vector Machines (TSVM) so that the decision rule lies in a region of low density; and

(ii) learning the *cluster* or *manifold* structure from the unlabeled data as in cluster kernels (Chapelle et al., 2003), label propagation (Zhu & Ghahramani, 2002), and Laplacian SVMs (Belkin et al., 2006).

Both approaches can be seen as making the same *structure assumption* on the data, that the cluster or manifold structure in the data is correlated with the class labels of interest.

The Low Density Separation algorithm (LDS) (Chapelle & Zien, 2005) is a two-stage algorithm that combines both of these approaches, with improved results over using only one of the techniques, however the combination method is somewhat *ad-hoc*.

A serious problem with all these methods is that they suffer from an inability to scale to very large datasets, apart from in the linear case (Sindhwani & Keerthi, 2006). This is ironic because the potential gain of semi-supervised learning lies in the vast amounts of readily available unlabeled data. This performance gain is never attained simply because of the computational burden of calculating the result. In the conclusion of the article describing the LDS algorithm the authors state:

"We observe that the time (and to some degree, also space) complexities of all methods investigated here prohibit the application to really large sets of unlabeled data, say, more than a few thousand. Thus, work should also be devoted to improvements of the computational efficiency of algorithms, ideally of LDS."

In this work we propose a new method for semi-supervised learning which features the following improvements over existing approaches:

- A new regularizer for semi-supervised learning is proposed, that is a unification of the approaches in both *margin-based* and *manifold-based* regularization. As such it represents a clean version of

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

the LDS method in a single objective, rather than an *ad-hoc* two-stage approach. Experimental results show its good performance.

- We train our system using stochastic gradient descent and choose linear or multi-layer architectures rather than kernel methods. This results in far faster training and testing times than existing methods, and also allows semi-supervised learning to be performed *online*. Our method can easily scale to *millions* of examples.

- We show it is also possible to encode domain knowledge into our multi-layer architecture approach, resulting in excellent generalization performance. This is demonstrated by training semi-supervised convolutional networks for image data.

The rest of the article is as follows. Section 2 describes in detail existing margin and manifold based regularization approaches, and scalability of the resulting algorithms. Section 3 describes our proposed approach, Section 4 compares it experimentally to existing methods, and Section 5 concludes.

2. Existing Approaches

As stated in the introduction, two of the most popular loss functions (regularizers) for using unlabeled data are *margin*-based regularization as in TSVMs and *manifold*-based regularization. We will discuss each of these in turn.

2.1. TSVMs

The Transductive Support Vector Machine (TSVM) is an algorithm originally proposed by Vapnik (1998) to take advantage of both a labeled training set and an unlabeled test set during prediction time. It was named that way because Vapnik proved bounds on generalization performance given the availability of the test set that were superior to induction based on using the labeled training set alone. The idea of the algorithm was:

(i) Choose a nested set of functions $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \dots$ of increasing capacity.

(ii) For each possible labeling of the test examples, find the smallest subset $\mathcal{F}_k$ that can classify both training and testing data correctly.

(iii) Choose the labeling which required the smallest index k.

In terms of actual implementation it is known that the notion of margin – the distance of examples from the classifier's decision rule – is connected to the concept of capacity (Vapnik, 1998), so a simple algorithm is the following: *choose the decision rule that maximizes the margin on both labeled and unlabeled examples.*

The Support Vector Machine (Vapnik, 1998) for two-class classification already implements a margin based capacity control on *labeled* examples, using an optimization problem of the following form:

$$\min_{w,b} \gamma||w||^2 + \sum_{i=1}^{L} \ell(f(x_i), y_i) \tag{1}$$

where the family of functions are

$$f(x) = w \cdot x + b \tag{2}$$

and $\{(x_i, y_i), \dots, (x_L, y_L)\} \subset \mathbb{R}^d \times \{\pm 1\}$ are the labeled training examples, and the loss function $\ell(\cdot, \cdot)$ is the so-called hinge loss:

$$\ell(f(x), y) = \max(0, 1 - yf(x)). \tag{3}$$

To implement Transductive SVMs it is (almost) sufficient to take the SVM optimization problem (1) and add an extra term for the unlabeled examples:

$$\min_{w,b} \gamma||w||^2 + \sum_{i=1}^{L} \ell(f(x_i), y_i) + \lambda \sum_{i=1}^{U} \ell^*(f(x_i^*)) \tag{4}$$

where the U unlabeled examples use the so-called symmetric hinge loss function

$$\ell^*(f(x^*)) = \max(0, 1 - |f(x^*)|) \tag{5}$$

which, intuitively speaking, pushes the unlabeled examples far from the margin: the absolute value is necessary in equation (5) because one does not know which side of the hyperplane those examples should lie on, unlike the labeled examples, so effectively the classifier trains on its own predictions. This notion of *self-learning* (Chapelle et al., 2006) can cause disastrous consequences in some cases: especially when the dimensionality $d \gg L$ one might be able to classify all unlabeled examples as belonging to one class whilst still classifying the labeled data correctly, giving a low value of the objective function, but nonsense results. This is solved by introducing a so-called balancing constraint which attempts to keep some of the unlabeled examples in each class.

Many researchers seem to be believe that the TSVM objective function is a good choice for semi-supervised

learning. However, finding a solution to the non-convex problem is far from easy, and thus several implementations have been attempted thus far. We will now describe some of those specific implementations, and their key differences.

S^{3}VM The authors of (Bennet & Demiriz, 1998) proposed to use mixed integer programming to find the labeling with the lowest objective function. The optimization appears intractable for large datasets, as "the solver failed due to excessive branching" in those cases. Only the linear case was considered, and no balancing constraint was used.

SVMLight-TSVM In (Joachims, 1999) a heuristic algorithm was proposed that at first fixes the labels of the unlabeled examples and then iteratively switches those labels to improve the TSVM objective function, solving a convex SVM objective function at each step. The nonlinear case is implemented by solving in the dual, resulting in a kernel model of the form:

$$f(x) = \sum_{i=1}^{L} \alpha_i y_i K(x_i, x) + \sum_{i=1}^{U} \alpha_i^* K(x_i^*, x) + b \quad (6)$$

A balancing constraint enforces that the fraction of positive and negatives assigned to the unlabeled data should be the same fraction as found in the labeled data. According to the proof of convergence, the algorithm at worst case could look at all 2^U labelings, but this is rather unlikely. The algorithm can deal with a few thousand examples in the nonlinear case in practice, but is faster in the linear case.

VS^{3}VM In (Fung & Mangasarian, 2001) a concave-convex minimization approach was proposed that solves successive convex problems, usually requiring only 5-7 linear programs, where they chose the L_1 norm of w as a regularizer instead of the L_2 norm. They studied the linear case, with no balancing constraint. This method will scale like the linear solver used in each iteration.

∇TSVM More recently, the authors of (Chapelle & Zien, 2005) proposed to optimize TSVM by gradient descent in the primal. For the nonlinear case, Kernel PCA has to be performed so that optimization in the primal is possible. This algorithm is faster than SVMLight-TSVM at least for small datasets (Collobert et al., 2006), but still has cubic complexity $O((U+L)^3)$. This method also requires one to store the entire kernel matrix of $(U+L)^2$ elements in memory, which clearly becomes infeasible for large datasets. The authors introduced a balancing constraint that is amenable to gradient descent:

$$\frac{1}{U}\sum_{i=1}^{U} f(x_i^*) \;=\; \frac{1}{L}\sum_{i=1}^{L} y_i \,. \quad (7)$$

CCCP-TSVM The authors of (Collobert et al., 2006) proposed to apply the Concave-Convex procedure for non-convex problems to TSVMs, which can be seen as a nonlinear extension of VS^{3}VMs. It uses the same balancing constraint as ∇TSVM. This implementation is over 100 times faster than SVMLight-TSVM and 50 times faster than ∇TSVM (for $L+U=2000$), and appears to scale better as well. It has empirically quadratic complexity because it relies on the sparsity of the SVM solution for improved speed and memory requirements. However, it still takes around 40 hours on a modern machine to solve a problem with 60,000 unlabeled examples in the nonlinear case.

Large Scale Linear TSVMs The authors of (Sindhwani & Keerthi, 2006) recently proposed a large scale TSVM method for the linear case. They focused on text problems with large sparse feature vectors and train the model (2) directly in the primal. In particular, they use a label switching heuristic like SVMLight-TSVM, but switch multiple labels at once.

In the nonlinear case things are not so easy. One is restricted in the quest to reduce training time by the prediction speed of the model (6) . Moreover, computation grows as the training data grows (Steinwart & Scovel, 2005). Even if one tries tricks to keep a fixed number of basis functions these methods are still slow compared to multi-layer models (Burges, 1996).

2.2. Manifold-based regularization

A separate direction of research in semi-supervised learning is manifold-learning based regularization. The main idea in these approaches is to find a representation of the data which collapses points lying in the same manifold so that a classification algorithm can easily predict that they share the same class label.

These methods can be split into two categories: those which treat this as a two-stage problem: (i) learn an embedding and (ii) train a classifier in this new space, and those which try to do everything in a single step.

To train a two-stage classifier, in the first stage one employs any manifold-learning algorithm such as Isomap (Tenenbaum et al., 2000), Laplacian Eigenmaps (Belkin & Niyogi, 2003) or spectral clustering (Ng et al., 2002). The authors of (Chapelle et al., 2003) use such methods to build a kernel for SVMs

and call these kernels "cluster kernels". The "graph"-SVM method proposed in (Chapelle & Zien, 2005) also builds a kernel for SVM. In this method one embeds in a space where distances are the shortest paths on the graph weighted with the original distance measure, similar to the Isomap algorithm. Thus, points connected by regions of high density are close to each other in the new space.

To learn a single stage classifier, one has to introduce a regularizing term in the objective function which directly encodes behavior such as that described in the previous paragraph. The Laplacian Eigenmaps embedding algorithm in particular employs an objective function that is easily encoded in a classifier:

$$\sum_{ij} W_{ij}||f(x_i) - f(x_j)||^2 \tag{8}$$

Such a regularizer has been used both to generalize a Parzen-windows (Duda & Hart, 1973) type classifier resulting in a method called label propagation (Zhu & Ghahramani, 2002), and in SVMs. The SVM method is called Laplacian SVMs (LapSVM) (Sindhwani et al., 2005) and minimizes:

$$\min_{w,b} \sum_{i=1}^{L} \ell(f(x_i), y_i) + \gamma||w||^2 + \lambda \sum_{i,j=1}^{U} W_{ij}||f(x_i^*) - f(x_j^*)||^2 \tag{9}$$

We speculate here that forcing the Euclidean distance to be small if two points are assumed to be the same label might be a little stringent as for prediction it is only the sign of $f(x^*)$ that is important. Moreover, we also note that the lack of balancing constraint might mean in high dimensions that all the unlabeled examples can collapse to a single prediction.

In contrast, the LDS method (Chapelle & Zien, 2005) proposes to use both TSVM and manifold regularizers at once in a two-stage method. First, the Isomap-like embedding method of "graph"-SVM is used whereby data is clustered. Then, in the new embedding space, ∇TSVM is applied. The authors found that using both regularizers at once was better than using one type of regularizer alone.

In summary, we have discussed several algorithms which use two main types of regularizer: a clustering or an embedding that takes into account *structure* in the unlabeled data. Indeed TSVM is a kind of large margin clustering as has been exploited in (Xu et al., 2005) and is strongly related to classical techniques like competitive learning (Duda & Hart, 1973). In (Chapelle & Zien, 2005) the authors speculate that manifold-based regularization has a stabilizing effect on TSVM optimization. Without such neighborhood-based regularization TSVMs only compare unlabeled examples to the existing model, and not to each other. Using both approaches as in LDS is thus a smart idea, however it suffers from two problems: (i) the two-stage approach seems *ad-hoc* and (ii) the method is slow.

In the next Section we propose a new approach which remedies these problems.

3. Proposed Approach

We propose the following algorithm, named Manifold Transduction: minimize

$$\frac{1}{L}\sum_{i=1}^{L} \ell(f(x_i), y_i) + \frac{\lambda}{U^2} \sum_{i,j=1}^{U} W_{ij}\ \ell\big(f(x_i^*), y^*(\{i,j\})\big) \tag{10}$$

where

$$y^*(N) = \text{sign}(\sum_{k \in N} f(x_k^*)) \tag{11}$$

where the edge weights W_{ij} define pairwise similarity relationships between unlabeled examples x^*.

This objective, like TSVMs objective, is non-convex and there is no simple optimization scheme for solving it even for linear models such as kernel machines. Because of this fact, and the scalability problems with nonlinear kernel methods, we propose several novel algorithmic choices in its implementation:

(i) We minimize this function in the primal by stochastic gradient descent. This makes *online* semi-supervised learning possible for the first time.

(ii) In the nonlinear case we employ a multi-layer architecture to define $f(x)$. This makes both training and testing far faster than competing kernel methods such as TSVM.

(iii) We also make a specific recommendation for the implementation of an *online* balancing constraint.

We will now study this algorithm, and explain the reason for these choices in detail.

3.1. Objective function

In (10) we propose a new loss function for *unlabeled examples*:

$$\ell^*(f(x_i^*)) = \ell\Big(f(x_i^*), y^*(N)\Big) \tag{12}$$

where N is a set of examples that one believes share the same label, e.g. a set of neighboring examples. The

function y^* predicts the label of that set by taking the mean prediction.

For both labeled and unlabeled training data we use the hinge loss (3) as in SVMs.

In equation (10) we consider pairs of examples, weighted by the graph W_{ij}. If $W_{ii} = 1$ and $W_{ij} = 0$ for $i \neq j$ then we recover the TSVM loss function:

$$\ell^*(f(x_i^*)) = \ell\Big(f(x_i^*), \text{sign}(f(x_i^*))\Big) \qquad (13)$$

because we do not take neighborhood information into account.

Setting $W_{ij} = 1$ if x_i^* is among the k-nearest neighbors of x_j^*, and zero otherwise, our algorithm becomes a natural generalization of TSVM that regularizes using neighborhood information. This is a similar regularizer to the neighborhood-based manifold regularizers of Section 2.2 but based on *clustering* rather than *embedding*.

We make the assumption that if two examples are neighbors then they have the same *class label*, whereas manifold-based regularization assumes they are close in an *embedding space*. Our constraint is not as strict, but captures the prior we wish to encode. For example, if one class of data has more variance than the other, then the regularization of (9) might focus on that class, and ignore the other.

Extensions of our algorithm are also possible. First, in the multi-class case where $f(x^*)$ outputs a c-dimensional vector, we can define $y^*(N) = \text{argmax} \sum_{k \in N} f(x_k^*)$. Further, if the set N contains more than two examples then our algorithm takes into account a neighborhood in analogy to k-nearest neighbor. This is not easily possible with the approach of (9) which is limited to pairs.

3.2. Model: Multi-Layer Architecture

As already discussed, the issue that makes all the previously described algorithms computationally expensive in the nonlinear case is their choice of the kernel expansion (6). Instead we propose to use a multi-layer model of the form:

$$f(x) = \sum_{i=1}^{d} w_i^0 \; h_i(x) + b$$

where typically one chooses hidden units

$$h_i(x) = S\left(\sum_j w_j^i \; x_j + b^i\right)$$

Algorithm 1 Online Manifold Transduction

Input: labeled data (x_i, y_i) and unlabeled data x_i^*
repeat
  Pick a random labeled example (x_i, y_i)
  Make a gradient step to optimize $\ell(f(x_i), y_i)$
  Pick a random unlabeled example x_i^*
  Pick a random neighbor x_j^* of x_i^*
  Predict label $y^* = y^*(\{i, j\})$
  if fraction of recent assignments to class $y^* < p_{est}(y^*)$ (see Section 3.4) **then**
    Make a gradient step for $\ell(f(x_i^*), y^*)$
  end if
until stopping criteria is met.

where S is a non-linear squashing function. We use the Hard Tanh function:

$$S(x) = \begin{cases} 1 & \text{if } x \geq 1 \\ -1 & \text{if } x \leq -1 \\ x & \text{otherwise.} \end{cases}$$

In the multi-class case we define one output $f_i(x)$ for each class, but each function f_i shares the same hidden units h_j, as is often done in neural network models.

The flexibility of using multi-layer architectures also allows us to encode prior knowledge into our model. For example, convolutional neural networks (CNNs) (LeCun et al., 1998) have several layers of image patch based feature maps applied across the input image. Such networks have been shown to perform very well in digit, face and 3D object detection tasks.

3.3. Optimization: Stochastic Gradient

We optimize our objective *online*, in the primal, using stochastic gradient descent. Recent experimental comparisons show this approach often outperforms sophisticated optimizer schemes (Bottou, 2007). To simplify the hyperparameters we fix $\lambda = 1$ in our experiments, yielding the method described in Algorithm 1. If the model is multi-layered then we use backpropagation (see, e.g. (Duda & Hart, 1973)) during the gradient step. A typical stopping criteria is to use a validation set or to measure the objective function value.

3.4. Balancing Constraint

To implement a balancing constraint while learning *online* we keep a cache of (arbitrarily) the last $25c$ predictions $f(x_i^*)$ where c is the number of classes. This is dependent on c because if c is large the cache must also be large or the estimates will be too poor. We then try to make the next prediction *balanced* assuming we

have a fixed estimate $p_{est}(y)$ of the probability of each class, which without further information, can be estimated from the labeled data: $p_{trn}(y=i) = \frac{|\{i:y_i=i\}|}{L}$. We consider two alternatives:

1. **∇bal** Adding the term (7) to the objective function multiplied by a scaling factor as in ∇TSVMs. The disadvantage of such an approach is that the scaling factor is a further hyperparameter.
2. **ignore–bal** Count how many examples in the cache have been attributed to each class. If the next unlabeled example x^* is given a label y^* by the model that already has too many examples assigned to it, then we simply do not make a gradient step for this example.

We note that the quality of p_{trn} depends on the ratio of labeled examples L to the number of classes c, not the input dimensionality d. Thus it may be a good estimate in many real datasets. However, because in some of the small datasets used in (Chapelle & Zien, 2005) it is a poor estimate we consider improving this estimate by taking into account that we have access to unlabeled data. We suggest the following simple method p_{knn}: label the k nearest neighbors of each labeled example with its label. If k is large enough some labeled points will label the same examples, and so when we count the number of points assigned to each class, we achieve a smoothed version of p_{trn}.

4. Experiments

4.1. Small Scale Datasets

We first report results on three small-scale datasets, summarized in Table 1. We follow the methodology in (Chapelle & Zien, 2005; Collobert et al., 2006) and report the best mean test error for a *fixed* set of hyperparameters over 10 splits of the data. For our method, we test standard transduction (our regularizer with no neighborhood information), called TNN (Transductive Neural Network), and our method *with* neighborhood information, called ManTNN (Manifold Transduction Neural Network). We also compute a baseline Neural Network (NN).

For NN, TNN and ManTNN we fixed 50000 iterations of Algorithm 1 and for ManTNN we chose 10 nearest neighbors for all datasets. We also choose not to minimize $\ell(f(x_i^*), y^*)$ for the first $10L$ iterations so that the classifier first finds a good model with labeled data alone before using the unlabeled data. We thus have two free parameters: the choice of hidden units {0,50,100,150,200} and the choices of learning rate { 0.5, 0.1, 0.05, 0.01, 0.005, 0.001, 0.0005 }.

Table 1. Datasets used in the experiments. The first three are small-scale datasets using the same experimental setup as found in (Chapelle & Zien, 2005). Mnist1h and Mnist1k use the same experimental setup as in (Collobert et al., 2006). Mnist1k+Invar uses shifted versions of digits to make an unlabeled set of 630,000 examples.

data set	classes	dims	points	labeled
g50c	2	50	500	50
Text	2	7511	1946	50
Uspst	10	256	2007	50
Mnist1h	10	784	70k	100
Mnist1k	10	784	70k	1000
Mnist1k+Invar	10	784	630k	1000

Table 2. Test Error for various methods for enforcing the balancing constraint, see Section 3.4 for explanation. The "no bal" method does not use a balancing constraint. p_{trn} and p_{tst} balance using the training and testing set distributions respectively, and p_{knn} estimates the true distribution using a k-nn based method on the unlabeled data.

	Uspst			g50c		
	p_{trn}	p_{knn}	p_{tst}	p_{trn}	p_{knn}	p_{tst}
TNN						
no bal	22.3	–	–	6.5	–	–
∇bal	30.4	29.3	29.4	6.5	6.5	6.5
ignore-bal	19.1	16.1	12.5	6.1	6.3	6.3
ManTNN						
ignore-bal	15.6	11.9	8.5	5.9	5.7	5.5

Table 3. Transductive (T-) and Manifold Transduction (ManT-) versions of Neural Networks (NN) as well as a baseline NN are compared to existing methods on Small-Scale Datasets. Following (Chapelle & Zien, 2005) all methods apart from those marked (*) have test error rates reported for a *fixed* set of hyperparameters averaged over 10 splits, where that fixed set is chosen using the test error itself. All methods have 2 free hyperparameters. In comparison, the methods marked (*) have parameters optimized on each split using 5-fold cross-validation.

	g50c	Text	Uspst
SVM	8.32	18.86	23.18
SVMLight-TSVM	6.87	7.44	26.46
CCCP-TSVM	5.62	7.97	16.57
∇TSVM	5.80	5.71	17.61
LapSVM$^{(*)}$	5.4	10.4	12.7
LDS$^{(*)}$	5.4	5.1	15.8
Label propagation	17.30	11.71	21.30
graph	8.32	10.48	16.92
NN	8.54	15.87	24.57
TNN	6.34	6.11	16.06
ManTNN	5.66	5.34	11.90

Balancing constraint comparison We first compare the balancing constraint methods ∇bal and ignore-bal and three different strategies for computing the class distribution p_{trn}, p_{knn} and p_{tst} as described in Section 3.4. p_{trn} measures the training set distribution, p_{knn} estimates the unlabeled set distribution (which is what we are really interested in) by using a k-nn like method, and p_{tst} is the true distribution of the unlabeled data, which is inaccessible in a real situation. A comparison on two of the datasets is given in Figure 2 (results are similar for "text").

We could not get ∇bal to work well in an online situation whereas the simple ignore-bal heuristic gives good results. Due to the small dataset size the difference in test error between using p_{tst} and p_{trn} is actually quite large. Using p_{knn} seems to be a better estimate than p_{trn}. We therefore adopt ignore-bal and the p_{knn} method in the following experiments.

Comparison with other methods We compare TNN and ManTNN to several TSVM implementations as well as label propagation, graph, LapSVM and LDS on the small scale datasets. The results given in Table 3 show that both TNN and ManTNN are competitive with existing approaches, and ManTNN, which includes the manifold-based transductive regularizer, outperforms TNN, which uses transduction alone.

4.2. Large Scale Dataset: MNIST

We then compared our method to SVMs and TSVMs on a "semi-supervised version" of the MNIST digit database, following (Collobert et al., 2006), using either 100 or 1000 labeled examples and 70000 unlabeled examples, and a validation set of 1000 examples for choosing parameters. Error rates are measured on the MNIST test set.

We used a two-layer neural network as before (baseline NN, TNN, ManTNN), choosing from the same set of hidden units and learning rates. We use the validation set as a stopping criteria for Algorithm 1.

We also applied convolutional networks (CNNs), and transductive versions of them, to this task. We chose an architecture similar to (LeCun et al., 1998). There are 6 layers. The first is six 3x3 spatial convolutions (outputting 26x26x6 features to the next layer). The second is six spatial 2x2 spatial subsamplings (outputting 13x13x6 features). The third is sixteen 4x4 spatial convolutions (outputting 10x10x16 features). The fourth is sixteen 2x2 spatial subsamplings (giving 5x5x16 features). The fifth is fifty 5x5 spatial convolutions (giving 1x1x50 features). This is followed by a standard fully connected layer with n hidden units (chosen as in the two-layer net), followed by a linear layer yielding the final 10 outputs (class predictions). CNNs encode prior knowledge about spatial features within the image, which should give improved accuracy over a standard NN.

Table 4. Results on Large-Scale Datasets: MNIST with 100 or 1000 labels and 70,000 unlabeled examples. Test Error is reported for Transductive (T-) and Manifold Transduction (ManT-) versions of Neural Networks (NN) and convolutional networks (CNN), and compared to SVMs and TSVMs. *ManTCNN (p_{tst})* uses the test distribution as the balancing constraint, which if this information were available, would give improved performance.

	Mnist1h	Mnist1k
SVM	23.44	7.77
CCCP-TSVM	16.81	5.38
NN	25.81	10.70
TNN	18.02	6.66
ManTNN	7.30	2.88
CNN	22.98	6.45
TCNN	13.01	3.50
ManTCNN	6.65	2.15
ManTCNN (p_{tst})	*1.96*	*1.87*

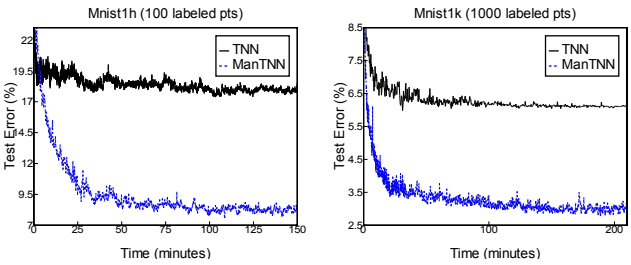

Figure 1. Test error versus training times for the TNN and ManTNN algorithms on Mnist (100 or 1000 labeled examples, 70000 unlabeled examples). These results compare favourably with the time to train the fastest TSVM algorithm (Collobert et al., 2006) which took 41.9 hours on the same machine.

The results are given in Table 4. The baseline NN performs slightly worse than SVM, but CNNs perform slightly better. Applying transduction, TNN is slightly worse than TSVMs, but TCNN is slightly better. Manifold Transduction outperforms all these methods, with ManTNN and ManTCNN performing almost as well as each other. The last row in the table shows ManTCNN trained with the true balancing constraint (knowing the test distribution). It appears that for only 100 labeled examples knowing this distribution could make results even better, although with 1000 labeled examples this is less important.

Training time for TNNs and ManTNNs are given in Figure 1. The results are shown for the best choice of hidden units and learning rate as chosen on the validation set. TNNs take around one hour to reach convergence, and ManTNNs (omitting the time to compute neighbors for ManTNN) take slightly longer. These times should be compared to the fastest TSVM implementation, CCCP-TSVMs, which took 41.9 hours on the same machine. Our code is not particularly optimized and is written in a scripting language with a C++ back-end. On MNIST, a nonlinear TNN with 200 hidden units can process 1 million unlabeled examples in an *online* fashion in 12.5 minutes.

Mnist1k+Invar We also performed experiments on MNIST1k with a larger unlabeled set of 630,000 examples by translating the original set by at most one pixel in each direction. TNN achieves a test error of 5.23% on the original test set, when choosing the training iteration that gives the minimum validation error, and ManTNN achieves a test error of 2.43%. Both methods outperform their counterparts trained with less unlabeled data using MNIST1k. Training time took 4.47 hours and 3.96 hours respectively for the two algorithms, including the computation time for generating the invariances.

5. Conclusions

In this article we introduced a large scale *nonlinear* method that elegantly combines the two main regularization principles for discriminative semi-supervised learning: transduction and neighborhood-based (manifold-based) regularization. Our future work will be to apply this approach to real large-scale nonlinear problems e.g. applications in vision and natural language processing.

References

Belkin, M., & Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: a geometric framework for learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, *7*, 2399–2434.

Bennet, K., & Demiriz, A. (1998). Semi-Supervised Support Vector Machines. *NIPS 12, 1998*. MIT Press, Cambridge, MA.

Bottou, L. (2007). `http://leon.bottou.org/projects/sgd`.

Burges, C. (1996). Simplified Support Vector Decision Rules. *ICML*, 71–77.

Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. Adaptive computation and machine learning. Cambridge, Mass., USA: MIT Press.

Chapelle, O., Weston, J., & Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. *NIPS 15* (pp. 585–592). Cambridge, MA, USA: MIT Press.

Chapelle, O., & Zien, A. (2005). Semi-supervised classification by low density separation. *AISTATS* (pp. 57–64).

Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Large scale transductive svms. *Journal of Machine Learning Research*, *7*, 1687–1712.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley & Sons.

Fung, G., & Mangasarian, O. (2001). Semi-supervised support vector machines for unlabeled data classification. *Optimization Methods and Software*, *15*, 29–44.

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

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*.

Ng, A. Y., Jordan, M., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press.

Sindhwani, V., & Keerthi, S. S. (2006). Large scale semi-supervised linear SVMs. *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 477–484). New York, NY, USA: ACM Press.

Sindhwani, V., Niyogi, P., & Belkin, M. (2005). Beyond the point cloud: from transductive to semi-supervised learning. *International Conference on Machine Learning, ICML*.

Steinwart, I., & Scovel, C. (2005). Fast rates to bayes for kernel machines. *NIPS*, *17*, 1345–1352.

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

Vapnik, V. N. (1998). *Statistical learning theory*. John Wiley and Sons, New York.

Xu, L., Neufeld, J., Larson, B., & Schuurmans, D. (2005). Maximum margin clustering. *Advances in Neural Information Processing Systems*, *17*, 1537–1544.

Zhu, X., & Ghahramani, Z. (2002). *Learning from labeled and unlabeled data with label propagation* (Technical Report CMU-CALD-02-107). Carnegie Mellon University.