Large-Scale Semi-Supervised Learning

Jason WESTON [a]
[a] *NEC LABS America, Inc., 4 Independence Way, Princeton, NJ, USA 08540.*

Abstract. Labeling data is expensive, whilst unlabeled data is often abundant and cheap to collect. Semi-supervised learning algorithms that use both types of data can perform significantly better than supervised algorithms that use labeled data alone. However, for such gains to be observed, the amount of unlabeled data trained on should be relatively large. Therefore, making semi-supervised algorithms scalable is paramount. In this work we review several recent techniques for semi-supervised learning, and methods for improving the scalability of these algorithms.

Keywords. semi-supervised, unlabeled data, transduction, large-scale, scalability

Introduction

One of the major ongoing challenges for the field of machine learning is dealing with the vast amount of data being generated in target applications. A lot of recent sophisticated machine learning algorithms appear to perform well on relatively small problems but don't scale very well (in terms of computational time) to a large amount of training data. This is ironic because with increasing dataset sizes, machine learning algorithms are guaranteed improved performance, but this performance is never attained simply because of the computational burden of calculating the result.

Another issue is that even though there is a large amount of data available, supervised machine learning algorithms need more than this: they need data that is labeled with a supervised target. Classification, regression and structured learning algorithms often give excellent performance compared to non-machine learning alternatives such as hand-built systems and rules, as long as the labeled data provided is of sufficient quality. However, although unlabeled data, such as data on the web, is abundant, labeled data is not – it is in fact often quite expensive to obtain, both in terms of monetary cost and labeling time.

Semi-supervised learning [2] is a framework in machine learning that provides a comparatively cheap alternative to labeling a huge amount of data. The aim is to utilize both the small amount of available labeled data *and* the abundant unlabeled data together in order to give the maximum generalization ability on a given supervised task. Using unlabeled data together with labeled data often gives better results than using the labeled data alone.

In this article we will discuss methods for performing semi-supervised learning which aim to scale up to large datasets to really achieve these goals. We will start with a summary of some of the classical approaches to this problem, such as transductive support vector machines (TSVMs) [16], label propagation type algorithms [20], and cluster-assumption encoding distance metrics [3,4]. We will then outline several recent tech-

niques for improving the scalability of these algorithms, focusing in particular on fast to compute distance metrics [17] for kernel methods and a fast optimization algorithm for TSVMs [5].

1. Semi-Supervised Learning

1.1. Supervised, Unsupervised and Semi-Supervised Learning

We will study the standard setting for machine learning algorithms where one is given a (possibly labeled) training set of m examples. If one is given data that is unlabeled:

$$(x_1^*), \ldots, (x_U^*), \tag{1}$$

then we are studying an *unsupervised* learning problem. Typical algorithms for making use of such data include estimating the density $p(x)$, clustering, outlier detection and dimensionality reduction – each paradigm having different aims with respect to what they want to discover from the data. The examples x_i could be any type of objects but are typically vectors in a d dimensional space, allowing such algorithms to make use of many mathematical techniques.

Sometimes, we have more information about the data. If a teaching signal is somehow provided that has *labeled* the data we now have data that are L pairs of examples:

$$(x_1, y_1), \ldots, (x_L, y_L). \tag{2}$$

This is called the *supervised learning problem* and includes classification, regression and structured output learning, amongst others, depending on the kinds of labels provided.

We note that in supervised learning, there are two main families of methods: generative models and discriminative models. In a generative model one is interested in estimating the joint probability distribution:

$$p(x, y) = p(x|y)p(y) \tag{3}$$

so that one can make predictions

$$p(y|x) = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)dy}. \tag{4}$$

In a discriminative model one only estimates the probability of label assignment given an example:

$$p(y|x) \tag{5}$$

or (in two-class classification) whether

$$p(y|x) > 0.5, \tag{6}$$

i.e. one is only interested in which class an example belongs to. In this article we will mostly focus on classification tasks using discriminative models.

However, one problem is that often a teaching signal is expensive or difficult to obtain. This is because human labeling of data can cost both time and money. The label-

ing may require an expert depending on the problem and the difficulty of the labeling task. For example in bioinformatics labels can be difficult to obtain, e.g. the function of a protein can only be obtained through successful wet-lab experiments. In such cases, to supplement the (small amount of) labeled data (2) one may have access cheaply to a large unlabeled set of data (1). This setting is called the *semi-supervised learning* setting. The hypothesis is that using both sets of data together can help improve generalization on the learning task of interest.

1.2. Semi-Supervised Learning Paradigms

Naively, given the definition above, one could see semi-supervised learning as a way of mixing unsupervised learning and supervised learning together. So for example one could perform density estimation or clustering on the unlabeled data and classification on the labeled data, somehow combining these predictions into a shared model. Indeed, such combinations are possible, and several variants already exist. In the following paragraphs we describe three particular paradigms for semi-supervised learning that fall under this description.

Supervised setting The typical setting for measuring the success of semi-supervised learning is to treat it as a supervised learning problem where one has additional unlabeled data that could potentially be of benefit. One first trains a model to predict labels given the labeled training set and unlabeled data, and then measures the error rate on a separate labeled test set. However, other interpretations of semi-supervised learning are possible.

Unsupervised setting One can also be interested in the task of unsupervised learning where one has additional labeled data that could potentially be of benefit. So for example, one can learn a clustering of the data, where one is given some extra must-link or must-not-link constraints that are implicit in the labeled data. This paradigm has the advantage of being able to handle missing classes, which could be important in some problems, e.g. speaker identification or protein family detection, to name two.

Level-of-detail setting Finally, yet another way of looking at semi-supervised learning is to see training labels y_i for examples x_i as having various levels of detail (granularity). For example in text processing, at a very coarse level one could label whether example sentences are grammatically correct or not. However, a labeling with a finer level of detail would also label the parts-of-speech of the words in the sentence. An even finer level of detail could specify the syntactic parse tree of the sentence. Seen this way, semi-supervised learning should handle labels y_i that go between two extremes: from no label at all to very detailed label information. Each example can be labeled with a different detail-level and the learning algorithm has to be designed to deal with this fact.

1.3. The History of Semi-Supervised Learning

Semi-supervised learning probably started with the concept of "self-learning" in the 1960s and 1970s [11]. In this paradigm one makes predictions on unlabeled data, and then uses these predicted labels to train the model as if one had an increased set of labeled data. Vapnik also introduced the concept of transduction in the 1970s [16]. The transductive support vector machine (TSVM) [16] algorithm can be seen algorithmically as a kind of refinement of the self-learning approach: one also looks at the predictions

of ones model on an unlabeled set of data, but this time one tries to choose the labeling which gives the most confident classification of those data, assuming the classifier outputs a measure of confidence, in this case using the notion of margin. Generative model approaches using the EM algorithm were also developed in the 1970s [8]. In [1] the co-training algorithm was introduced that is a clever use of unlabeled data that has multiple "views" or redundant sets of features to perform a kind of self-learning with a high degree of confidence. In the 1990s and 2000s several semi-supervised methods have been introduced that fit more into a regularization-based view – the use of unlabeled data is loosely seen as a regularization term in the objective function of a supervised learning algorithm. Amongst these are three notable algorithm types: TSVM-like algorithms, propagation of labels in a graph-based approach [20] and change of representation methods [3]. We will discuss these methods further in Section 2.

One issue with many of these algorithms is that they do not scale that well. In this article we will describe some of the recent advances in methods for making these algorithms more efficient. Those methods are discussed in Section 3.

1.4. Why Semi-Supervised Learning?

Supervised data is expensive both in monetary cost and labeling time. Labeling sentences with parse trees or finding protein function or 3D structure, to give two examples of labeled data, require human expertise.

Unlabeled data, on the other hand, is cheap to collect in many domains: audio-based problems, vision problems, and text processing problems, to name a few. In fact, most sensory-based problems that humans are good at have an abundant supply of unlabeled data. However, there are also other kinds of data, not natural to a human's perception, which are also relatively easy to collect compared to labeled examples of that same data. So, returning to our bioinformatics example, knowing the function (label) of a protein is costly, but obtaining its unlabeled primary structure (sequence of amino acids) is cheap.

In our view, true AI that mimics humans would be able to learn from a relatively weak training signal. For example, in the field of natural language processing, linguists label sentences with parse trees, but humans learn from data which usually has significantly less detailed labels. This argument perhaps strengthens the importance of *semi-supervised learning* as a topic of study in the field of machine learning.

1.5. Why Large-Scale Semi-Supervised Learning?

Many of the semi-supervised algorithms developed so far do not scale as well as one would like. For example, standard TSVM algorithms scale like $(U+L)^3$ where $U+L$ is the total number of examples [4], or worse. Because of this many experiments published are on relatively small, simple datasets comprising of around 50 training examples and a few thousand unlabeled examples [4,5,14].

Unfortunately, it appears that the impact of one extra unlabeled example in terms of improvement in generalization is smaller than the impact of one extra labeled example. Hence, if one has a realistic number of labeled examples, say a few thousand, then for unlabeled data to make a significant impact, you are likely to need hundreds of thousands of examples.

Hence, one clearly needs semi-supervised learning algorithms that scale well.

1.6. How Does Unlabeled Data Help in a Supervised Task?

Unlabeled data somehow gives knowledge about the density $p(x)$ but tells you nothing about the conditional density one is interested in $p(y|x)$ unless some assumptions are made in a training algorithm that hold true for a particular dataset.[1]

There are several possible assumptions that one can make about the data, each leading to different algorithms.

The cluster assumption One can assume that examples in the same cluster have the same class label. This implies that one should perform *low density separation* [4]; that is, the decision rule one constructs should lie in a region of low density. Many algorithms, such as TSVM [16], either explicitly or implicitly employ this assumption.

The manifold assumption One can also assume that examples in the same manifold have the same class. This is somewhat similar to the cluster assumption, but motivates different algorithms [20].

Zipf's law effect One obvious way unlabeled data can help in language problems is that one gets to see words that one has never seen before because a finite training set cannot cover the language. Zipf's law states that in a corpus of natural language utterances, the frequency of any word is roughly inversely proportional to its rank in the frequency table. In language problems, we are effectively always seeing new features (i.e. new words) every time we look at new examples. Our conjecture is that effective use of this knowledge in algorithms should surely improve performance over supervised learning alone.

Non-i.i.d. data Many machine algorithms and toy datasets assume that data are identically and independently distributed (i.i.d.) but in real life data this may not be true. We conjecture that if you see a test set that is drawn from a (slightly) different distribution to the training set then semi-supervised learning might help compared to purely supervised learning which is unaware of the distribution of the test set. For example, one might train a parser on the Wall Street Journal, but then apply it to the novel Moby Dick.

2. Semi-Supervised Learning Algorithms

In this section we will review some of the state-of-the-art methods for semi-supervised learning. While we cannot describe all available methods, we instead focus on three classes of method: (i) graph-based approaches, (ii) change of representation approaches and (iii) margin-based regularization based approaches.

All of these methods (either explicitly or implicitly) are special cases of a regularization approach to semi-supervised learning. In supervised learning a regularization approach leads one to optimize the empirical risk (training error) plus an additional term that limits the complexity of the decision rule:

$$\min_\alpha \sum_{i=1}^{L} \ell(f(x_i, \alpha), y_i) + \gamma\Omega(\alpha) \tag{7}$$

[1] Note that in supervised learning, a similar issue applies: most algorithms assume $p(y|x)$ and $p(y|x')$ are similar when x is close to x'.

where $\ell(\cdot)$ is a loss function encoding the penalty for incorrectly labeling a training example, and α encodes the parameters of the decision function.

In semi-supervised learning one approach is to do exactly the same as in supervised learning, but add one additional term that encodes the assumption one has made about the data w.r.t. using unlabeled examples, e.g. one may wish to encode the cluster assumption. This leads one to minimize:

$$\min_\alpha \sum_{i=1}^{L} \ell(f(x_i, \alpha), y_i) + \gamma\Omega(\alpha) + \lambda\Lambda(x^*, \alpha). \tag{8}$$

The regularizer $\Lambda(x^*, \alpha)$ is often point-wise, i.e.:

$$\Lambda(x^*, \alpha) = \sum_{i=1}^{U} \ell^*(f(x_i^*), \alpha) \tag{9}$$

where $\ell^*(\cdot)$ is a function that encodes the assumption of choice.

We now discuss some specific semi-supervised learning algorithms.

2.1. Graph-Based Approaches

Several semi-supervised algorithms work by constructing a graph on the unlabeled data, and then regularize using this graph in some way.

The authors of [20] (see also [12,19]) studied the so-called label-propagation type algorithm. In this algorithm one first defines F as a $(U + L)$ x 2 matrix for two-class classification problems, where

$$f(\bar{x}_i) = \text{argmax}_j F_{ij} \tag{10}$$

is the class prediction for a point $\bar{x}_i$, and $\bar{x}$ is the set of both labeled and unlabeled examples (ordered so that the labeled examples are first). A $(U+L)$ x 2 label matrix Y is also defined, where $Y_{ij} = 1$ if example $\bar{x}_i$ is classified into class j and $Y_{ij} = 0$ otherwise (so $Y_{ij} = 0$ for unlabeled examples $i > L$).

One then solves the following quadratic program:

$$\min_F \sum_{i=1}^{L+U} ||F_i - Y_i||^2 + \lambda \sum_{i,j=1}^{L+U} W_{ij}||F_i - F_j||^2 \tag{11}$$

where W_{ij} are weighted edges on the graph that one constructs a priori on the given data. For example, one can define $W_{ij} = 1$ if example i is one of example j's k-nearest neighbors, although many other schemes are possible. The matrix W is usually normalized so that the sum of outgoing links from any node in the graph sum to 1.

Intuitively, the first term minimizes the training error, whilst the second term regularizes using the unlabeled data. If two unlabeled examples are close in input space, then this algorithm tries to give them the same label. Because one is enforcing this constraint on all unlabeled and labeled points at the same time, there is a "label-propagation" effect where neighbors-of-neighbors will also try to have the same label as well. This can be clearly seen when writing down the algorithm as an iterative algorithm [19]:

1. Iterate $F(t + 1) = \alpha W F(t) + (1 - \alpha)Y$ until convergence.

2. Label each point $\bar{x}_i$ as $\arg\max_j F_{ij}(t)$.

That is, on each iteration, the label of an example is influenced by the weighted combination of all the labels of its neighbors. One can also write the solution in closed form:

$$F = (I - \alpha W)^{-1} Y. \tag{12}$$

This algorithm is essentially a generalized version of k-nearest neighbor with extra semi-supervised regularization. This can be seen because the first iteration predicts the label of an unlabeled example using the example's neighbors. Subsequent iterations provide regularization. One issue then is that although the regularization term seems reasonable, k-nn is not necessarily the best base classifier one could choose. A further issue is that, if one solves the problem as a quadratic program or as a matrix inversion problem, the complexity is $(L + U)^3$. However, one could *approximate* the solution by only running a few iterations of the iterative version of the algorithm.

LapSVM A more recent use of this same regularizer is the Laplacian SVM [14]. In that work they add the same regularizer to the SVM [16] objective function:

$$\min_{w,b} \sum_{i=1}^{L} \ell(g(x_i), y_i) + \gamma ||w||^2 + \lambda \sum_{i,j=1}^{U} W_{ij} ||g(x_i^*) - g(x_j^*)||^2 \tag{13}$$

where $g(x) = w \cdot x + b$. Here, as in a standard SVM, one is solving a two-class classification problem $y_i \in \pm 1$ and the final predictions are given by $f(x) = \text{sign}(g(x))$. This algorithm gives excellent results, but still suffers from having to compute a matrix inversion in the specific optimization technique the authors propose.

2.2. Change of Representation-Based Approaches

Another approach to semi-supervised learning is to simplify the regularization of the optimization: instead of optimizing both the supervised learning part of the optimization with an additional regularizer *at the same time* one could decompose into a two-part strategy:

- Build a new representation of the data using unsupervised learning on the unlabeled data.
- Perform standard supervised learning using the labeled data and the new representation.

The aim here is to map the data $x_i \mapsto \phi(x_i)$ such that the distances in the new representation encode manifold or cluster assumptions about the data. One can then use a standard classifier such as SVM on this data.

One study of this technique is given in [3]. In that work, several representations are defined for SVM that try to encode the cluster assumption. One desires the relative distance in the new representation is relatively smaller between two points that are in the same cluster. SVM training does not necessarily require the actual feature vectors of training examples in order to learn, but only the inner products:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j). \tag{14}$$

These inner products are referred to as kernels. The re-representations proposed by the authors of [3] are referred to as cluster kernels.

Two proposed cluster kernels are:

1. The random walk kernel. One defines $W_{ij} = \exp(-||x_i - x_j||^2/2\sigma^2)$ so that the probability of "walking" along the graph between points i and j is defined as $P(x_i \rightarrow x_j) = \frac{W_{ij}}{\sum_p W_{ip}}$. One can then compute t steps of a random walk using $P \leftarrow P^t$ (see also [15]). The idea here is to define a kernel[2] based on P^t. This representation has a similar motivation to the label propagation algorithm we have seen before: if it is easy to "walk" between two examples, i.e. they are neighbors, or neighbors-of-neighbors, then they are likely to have a similar label.

2. The spectral clustering kernel. One performs spectral clustering [10] on the labeled and unlabeled data, and then builds a representation via the first k eigenvectors for training the SVM.

Both of these kernels have a complexity of $O((U + L)^3)$ to compute.

2.3. Margin-Based Approaches

Another way of encoding a cluster assumption type regularizer is via the margin (distance) of the examples from the decision rule of a classifier of the form

$$g(x) = w \cdot x + b. \tag{15}$$

The transductive SVM (TSVM) is an example of this approach. The idea is that the decision rule should lie in a region of low density (the cluster assumption) so an unlabeled example x tends not to have the value $g(x) = 0$. This is encoded with the following optimization problem:

$$\min_{w,b} \sum_{i=1}^{L} \ell(g(x_i), y_i) + \gamma||w||^2 + \lambda \sum_{i=1}^{U} \ell^*(g(x_i^*)). \tag{16}$$

The loss functions $\ell(\cdot)$ and $\ell^*(\cdot)$ are typically chosen to be the hinge loss function

$$\ell(g(x), y) = \max(0, 1 - yg(x)) \tag{17}$$

and the symmetric hinge loss function

$$\ell^*(|g(x)|) = \max(0, 1 - |g(x)|) \tag{18}$$

respectively. The idea is that labeled examples are "pushed" away from the hyperplane so that they are correctly classified and achieve a margin of 1. Unlabeled examples are also "pushed" away from the hyperplane in the same way. To make this work, typically a balancing constraint is added so that all the unlabeled examples do not get "pushed" to the same side:

[2]Technically, one has to symmetrize this matrix to make it a valid kernel as described in [3].

Table 1. A comparison of supervised and semi-supervised algorithms on two small datasets.

Algorithm	*USPS*	*Mac-Win*
k-NN	9.3%±0.32	29.1%±0.68
SVM	7.4%±0.31	28.1%±1.39
Label propagation	1.79%±0.07	30.2%±1.35
SVM + cluster kernel	5.9%±0.24	11.0%±0.45
TSVM	5.62%±0.2	11.1%±0.48

$$\frac{1}{U}\sum_{i=1}^{U} g(x_i^*) \;=\; \frac{1}{L}\sum_{i=1}^{L} y_i \,. \tag{19}$$

Unfortunately this optimization problem is non-convex because of the absolute value in the $\ell^*(\cdot)$ function. The implementation in the SVMLight software [9] works by solving successive SVM optimization problems – it improves the objective function with a heuristic method: it iteratively switches the labels of two unlabeled points x_i and x_j to improve the objective function. It is rather slow with anything more than a few thousand examples. Another implementation called ∇TSVM [4] proposed to optimize this function directly by gradient descent, with a complexity of $(L+U)^3$.

2.4. Experimental Comparison

An experimental comparison on some small datasets is given in Table 1. Two supervised algorithms (k-NN and SVM) are compared to one algorithm from each of the three semi-supervised approaches described above: label propagation, SVM + spectral clustering kernel and TSVM. USPS is an optical digit recognition dataset, we used 32 labeled examples and 4000 unlabeled examples in 256 dimensions. Mac-Win is a text classification dataset (with a bag-of-words representation) with 16 labeled examples and 2000 unlabeled examples in 7511 dimensions. The results are average over 30 splits of the data. For USPS, semi-supervised learning improves the error rate. For Mac-Win the problem may be too high dimensional for label propagation to work well, however it does perform well on USPS.

3. Large-Scale Semi-Supervised Learning Algorithms

Having reviewed several of the semi-supervised algorithms developed in the literature, we will now describe several recent advances in speeding up these algorithms.

3.1. Graph-Based Approaches

As already mentioned, if one performs *early stopping* of the iterative algorithm for label propagation, instead of a direct optimization of the given objective function, one can speed this algorithm up. Such an approximation might give a loss of accuracy in the final predictions.

However, several other speed-ups are possible. Firstly, the algorithm is much faster if the matrix W is very sparse. Secondly it was proposed in [6] to approximate the so-

lution in a different way. The idea is that some unlabeled examples can be expressed as a linear combination of other examples, thus reducing the number of variables one has to optimize. This reduces the complexity of the algorithm from $O(k(U+L)^2)$, where each point has k neighbors, to $O(m^2(U+L))$ where $m << (U+L)$.

3.2. Change of Representation Approaches

For change of representation methods, the problem has been split into two tasks: an unsupervised learning algorithm and a supervised learning algorithm. Clearly then in order to perform large-scale semi-supervised learning, one should choose algorithms from both of these two areas that also scale well.

Therefore, one should first choose the fatest clustering algorithm one can. For example, k-means [7] is generally faster than spectral clustering, the algorithm we used before. It has a complexity of $O(rkUd)$, with a problem of dimensionality d running for r iterations. Empirically, running time grows sublinearly with k, n and d.

Thus, the following k-means cluster kernel was proposed in [17]:

1. Run k-means N times.
2. $K_{bag}(x_i, x_j) = \frac{\text{number of times } x_i \text{ \& } x_j \text{ in same cluster}}{\text{number of runs}}$
3. Take the product between the original and bagged kernel:
$$K(x, x') = K_{orig}(x, x') \cdot K_{bag}(x, x') \tag{20}$$

The idea is that the original kernel K_{orig} is rescaled by the "probability" that two points are in the same cluster as approximated by K_{bag}. This method was empirically evaluated in the context of protein superfamily classification, where one has to predict whether a protein belongs to a target superfamily or not. Experimental results are shown in Table 2, averaged over 54 target families. This method is compared on a small dataset with $L = 3000$ and $U = 3000$ so that other methods are feasible to compute, and then its error rate is computed with a much larger set of $U = 30000$ unlabeled examples. This method seems to have a good performance compared to other approaches, while having far better scaling properties.

3.3. Margin-Based Regularization Approaches

A well-founded technique for optimizing TSVM that also has improved scaling properties was recently developed in [5]. The idea is to make use of the concave-convex procedure (CCCP) [18], a technique of non-convex optimization. In CCCP, one rewrites the cost function of interest $J(\theta)$ as the sum of a convex part $J_{vex}(\theta)$ and a concave part $J_{cav}(\theta)$. One then iteratively solves

$$\theta^{t+1} = \text{argmin}_\theta \left\{ J_{vex}(\boldsymbol{\theta}) + J_{cav}^{'}(\boldsymbol{\theta}^t) \cdot \boldsymbol{\theta} \right\}. \tag{21}$$

Each iteration of the CCCP procedure approximates the concave part by its tangent and minimizes the resulting convex function. $J(\theta^t)$ is guaranteed to decrease at each iteration, and the whole procedure is guaranteed to converge to a local minima.

Table 2. Comparison of semi-supervised methods on a protein prediction problem taken from [17]. Error rates are measured using the Receiver Operator Characteristic (ROC) and the ROC up to the first 50 false positives (ROC_{50}).

	ROC_{50}	ROC
L=3000 U =3000		
SVM	0.416	0.875
SVM + spectral clustering kernel	0.581	0.861
SVM + random walk kernel	0.691	0.915
SVM + k-means kernel ($k = 100$)	0.719	0.943
SVM + k-means kernel ($k = 400$)	0.671	0.935
TSVM	0.637	0.874
U = 30000		
SVM + k-means kernel ($k = 100$)	0.803	0.953
SVM + k-means kernel ($k = 400$)	0.775	0.955

Table 3. A comparison of CCCP-TSVM with existing TSVM methods on a variety of small-scale datasets, taken from [5].

dataset	classes	dims	points	labeled
g50c	2	50	500	50
Coil20	20	1024	1440	40
Text	2	7511	1946	50
Uspst	10	256	2007	50

	Coil20	g50c	Text	Uspst
SVM	24.64	8.32	18.86	23.18
SVMLight-TSVM	26.26	6.87	7.44	26.46
∇TSVM	17.56	5.80	**5.71**	**17.61**
CCCP-TSVM	**17.28**	**4.88**	**5.71**	18.08

For the TSVM, such an approach is quite simple to implement and only requires solving a sequence of convex problems similar to the SVM algorithm. Hence it has quadratic empirical complexity in the number of examples, like SVMs have, and even on small problems of 2000 unlabeled examples is around 133 times faster than SVMLight, whilst having similar, or better, accuracy. An empirical comparison with existing approaches is shown in Tables 3, 4 and Figure 1. CCCP-TSVM sofware is available at `http://www.kyb.tuebingen.mpg.de/bs/people/fabee/universvm.html`.

Finally, another type of speed-up for TSVM was given in [13]. The authors proposed a "multi-switch" version of the SVMLight-based approach. The idea is to swap the labels of many examples at once, rather than the pair of examples approach suggested by [9]. This method was developed for a very fast linear SVM software called SVMLin, available at `http://people.cs.uchicago.edu/~vikass/svmlin.html`.

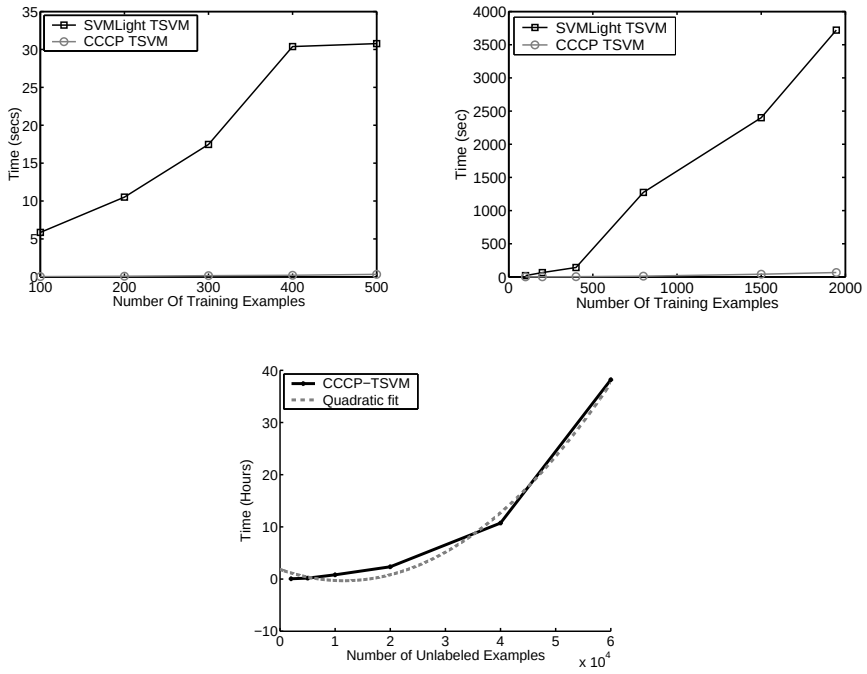

Figure 1. Training times for g50c (upper left), Text (upper right) and MNIST (lower) for different numbers of unlabeled examples, U. CCCP-TSVM are 133 times faster than SVMLight-TSVMs on the text dataset for $U = 2000$, and scale quadratically on larger datasets like MNIST.

Table 4. Large-scale results using CCCP-TSVMs [5] on the MNIST dataset, a 10-class digit recognition problem with $L = 1000$ labeled examples, and varying amount of unlabeled examples U.

Method	L	U	Test Error
SVM	1000	0	7.77%
TSVM	1000	2000	7.13%
TSVM	1000	5000	6.28%
TSVM	1000	10000	5.65%
TSVM	1000	20000	5.43%
TSVM	1000	40000	5.31%

4. Conclusion

In this article we have reviewed some of the main types of discriminative semi-supervised learning algorithms for classification that are available, and discussed some of the latest advances in making those algorithms scalable.

Semi-supervised learning is useful when labels are expensive, when unlabeled data is cheap and when $p(x)$ is useful for estimating $p(y|x)$, e.g. if either the manifold or cluster assumptions are true. We have reviewed several different algorithmic techniques for encoding such assumptions into learning, generally this is done by somehow "marrying" unsupervised learning into a supervised learning algorithm. Instances of this approach are graph-based approaches, change of representation based approaches and margin based approaches. All of these can somehow be seen as either explicitly or implicitly adding a regularizer that encourages that the chosen function reveals structure in the unlabeled data.

Large-scale learning is often realistic only in a semi-supervised setting because of the expense of labeling data. Moreover, the utility of an unlabeled example is less than a labeled one, thus requiring a relatively large collection of unlabeled data for its use to be effective. However, to make current algorithms truly large-scale, probably only linear complexity with respect to the number of examples will suffice. At least in the non-linear case, current approaches still fall short, leaving the field still open for further research.

References

[1] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, pages 92–100, 1998.

[2] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass., USA, 09 2006.

[3] O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. volume 15, pages 585–592, Cambridge, MA, USA, 2003. MIT Press.

[4] O. Chapelle and A. Zien. Semi-supervised classification by low density separation. pages 57–64, 01 2005.

[5] R. Collobert, F. Sinz, J. Weston, and L. Bottou. Large scale transductive svms. *Journal of Machine Learning Research*, 7:1687–1712, 08 2006.

[6] O. Delalleau, Y. Bengio, and N. Le Roux. Large-scale algorithms. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*, pages 333–341. MIT Press, 2006.

[7] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley & Sons, New York, 1973.

[8] D. Hosmer Jr. A Comparison of Iterative Maximum Likelihood Estimates of the Parameters of a Mixture of Two Normal Distributions Under Three Different Types of Sample. *Biometrics*, 29(4):761–770, 1973.

[9] T. Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning, ICML*, 1999.

[10] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, Cambridge, MA, 2002. MIT Press.

[11] H. Scudder III. Probability of error of some adaptive pattern-recognition machines. *Information Theory, IEEE Transactions on*, 11(3):363–371, 1965.

[12] J. Shrager, T. Hogg, and B. A. Huberman. Observation of phase transitions in spreading activation networks. *Science*, 236:1092–1094, 1987.

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

[14] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *International Conference on Machine Learning, ICML*, 2005.

[15] M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. *Neural Information Processing Systems 14*, 2001.

[16] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, New York, 1998.

[17] J. Weston, C. Leslie, D. Zhou, A. Elisseeff, and W. S. Noble. Semi-supervised protein classification using cluster kernels. volume 16, pages 595–602, Cambridge, MA, USA, 2004. MIT Press.

[18] A. L. Yuille and A. Rangarajan. The concave-convex procedure (CCCP). In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[19] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. volume 16, pages 321–328, Cambridge, MA, USA, 2004. MIT Press.

[20] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.