Learning by Combining Native Features with Similarity Functions

Mugizi Robert Rwebangira and Avrim Blum

February 2009
CMU-CS-09-107

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was sponsored by the National Science Foundation under contract no. IIS-0427206, National Science Foundation under contract no. CCR-0122581, National Science Foundation under contract no. IIS-0312814, US Army Research Office under contract no. DAAD190210389, and SRI International under contract no. 03660211. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: similarity, unlabeled data, semi-supervised

Abstract

The notion of exploiting data dependent hypothesis spaces is an exciting new direction in machine learning with strong theoretical foundations[66]. A very practical motivation for these techniques is that they allow us to exploit unlabeled data in new ways [2]. In this work we investigate a particular technique for combining “native” features with features derived from a similarity function. We also describe a novel technique for using unlabeled data to define a similarity function.

Contents

In this report we describe a new approach to learning with labeled and unlabeled data using similarity functions together with native features, inspired by recent theoretical work [2, 5]. In the rest of this report we will describe some motivations for learning with similarity functions, give some background information, describe our algorithms and present some experimental results on both synthetic and real examples. We give a method that given any pairwise similarity function (which need not be symmetric or positive definite as with kernels) can use unlabeled data to augment a given set of features in a way that allows a learning algorithm to exploit the best aspects of both. We also give a new, useful method for constructing a similarity function from unlabeled data.

1.1 Motivation

Two main motivations for studying the problem of learning with similarity functions are (1) Generalizing and Understanding Kernels and (2) Combining Graph Based or Nearest Neighbor Style algorithms with Feature Based Learning algorithms. We will expand on both of these below.

1.1.1 Generalizing and Understanding Kernels

Since the introduction of Support Vector Machines [62, 64, 65] in the mid 90s, kernel methods have become extremely popular in the machine learning community. This popularity is largely due to the so-called “kernel trick” which allows kernelized algorithms to operate in high dimensional spaces without incurring a corresponding computational cost. The ideas is that if data is not linearly separable in the original feature space kernel methods may be able to find a linear separator in some high dimensional space without too much extra computational cost. And furthermore if data is separable by a large margin then we can hope generalize well from not too many labeled examples.

However, in spite of the rich theory and practical applications of kernel methods, there are a few unsatisfactory aspects. In machine learning applications the intuition behind a kernel is that they serve as a measure of similarity between two objects. However, the theory of kernel methods talks about finding linear separators in high dimensional spaces that we may not even be able to calculate much less understand. This disconnect between the theory and practical applications makes it difficult to gain theoretical guidance in choosing good kernels for particular problems.

Secondly and perhaps more importantly, kernels are required to be symmetric and positive-semidefinite. The second condition in particular is not satisfied by many practically useful similarity functions(for example the Smith-Waterman score in computational biology [76]). In fact, in Section 3.1.1 we give a very natural and useful similarity function that does not satisfy either condition. Hence if these similarity functions are to be used with kernel methods, they have to be coerced into a “legal” kernel. Such coercion may substantially reduce the quality of the similarity functions.

From such motivations, Balcan and Blum [2, 5] recently initiated the study of general similarity functions. Their theory gives a definition of a similarity function that has standard kernels as a special case and they show how it is possible to learn a linear separator with a similarity function and give similar guarantees to those that are obtained with kernel methods.

One interesting aspect of their work is that they give a prominent role to unlabeled data. In particular unlabeled data is used in defining the mapping that projects the data into a linearly separable space. This makes their technique very practical since unlabeled data is usually available in greater quantities than labeled data in most applications.

The work of Balcan and Blum provides a solid theoretical foundation, but its practical implications have not yet been fully explored. Practical algorithms for learning with similarity functions could be useful in a wide variety of areas, two prominent examples being bioinformatics and text learning. Considerable effort has been expended in developing specialized kernels for these domains. But in both cases, it is easy to define similarity functions that are not legal kernels but match well with our desired notions of similarity (for an example in bioinformatics see Vert et al. [76]).

Hence, we propose to pursue a practical study of learning with similarity functions. In particular we are interested in understanding the conditions under which similarity functions can be practically useful and developing techniques to get the best performance when using similarity functions.

1.1.2 Combining Graph Based and Feature Based learning Algorithms.

Feature-based and Graph-based algorithms form two of the dominant paradigms in machine learning. Feature-based algorithms such as Decision Trees[56], Logistic Regression[51], Winnow[53], and others view their input as feature vectors and use feature values directly to make decisions. Graph-based algorithms, such as the semi-supervised algorithms of [7, 13, 15, 44, 63, 84, 85], instead view examples as nodes in a graph for which the only information available about them is their pairwise relationship (edge weights) to other nodes in the graph. Kernel methods [62, 64, 65, 66] can also be viewed in a sense as graph-based approaches, thinking of $\mathbb{K}(x, x')$ as the weight of edge (x, x').

Both types of approaches have been highly successful, though they each have their own strengths and weaknesses. Feature-based methods perform particularly well on text data, for instance, where individual keywords or phrases can be highly predictive. Graph-based methods perform particularly well in semi-supervised or transductive settings, where one can use similarities to unlabeled or future data, and reasoning based on transitivity (two examples similar to the same cluster of points, or making a group decision based on mutual relationships) in order to aid in prediction. However, they each have weaknesses as well: graph-based (and kernel-based) methods encode all their information about examples into the pairwise relationships between examples, and so they lose other useful information that may be present in features. Feature-based methods have trouble using the kinds of “transitive” reasoning made possible by graph-based approaches.

It turns out again, that similarity functions provide a possible method for combining these two disparate approaches. This idea is also motivated by the same work of Balcan and Blum[2, 5] that we have referred to previously. They show that given a pairwise measure of similarity $\mathbb{K}(x, x')$ between data objects, one can essentially construct features in a straightforward way by collecting a set $x_1, \ldots, x_n$ of random unlabeled examples and then using $\mathbb{K}(x, x_i)$ as the i^{th} feature of example x. They show that if $\mathbb{K}$ was a large-margin kernel function then with high probability the data will be approximately linearly separable in the new space. So our approach to combining graph based and feature based methods is to keep the original features and augment them (rather than replace them) with the new features obtained by the Balcan-Blum approach.

2.1 Background

We now give background information on algorithms that rely on finding large margin linear separators, kernels and the kernel trick and the Balcan-Blum approach to learning with similarity functions.

2.1.1 Linear Separators and Large Margins

Machine learning algorithms based on linear separators attempt to find a hyperplane that separates the positive from the negative examples; i.e if example x has label $y \in \{+1, -1\}$ we want to find a vector w such that $y(w \cdot x) > 0$.

Linear separators are currently among the most popular machine learning algorithms, both among practitioners and researchers. They have a rich theory and have been shown to be effective in many applications. Examples of linear separator algorithms are Perceptron[56], Winnow[53] and SVM [62, 64, 65]

An important concept in linear separator algorithms is the notion of “margin.” Margin is considered a property of the dataset and (roughly speaking) represents the “gap” between the positive and negative examples. Theoretical analysis has shown that the performance of a linear separator algorithm is directly proportional to the size of the margin (the larger the margin the better the performance). The following theorem is just one example of this kind of result:

Theorem In order to achieve error ϵ with probability at least $1 - \delta$, it suffices for a linear separator algorithm to find a separator of margin at leastγ on a dataset of size

$$O(\frac{1}{\epsilon}[\frac{1}{\gamma^2}\log^2(\frac{1}{\gamma\epsilon}) + \log(\frac{1}{\delta})]).$$

Here, the margin is defined as the minimum distance of examples to the separating hyperplane if all examples are normalized to have length at most 1. This bound makes clear the dependence on γ, i.e as the margin gets larger, substantially fewer examples are needed [16, 66] .

2.1.2 The Kernel Trick

A kernel is a function $\mathbb{K}(x, y)$ which satisfies certain conditions:

1. continuous
2. symmetric
3. positive semi-definite

If these conditions are satisfied then Mercer's theorem [55] states that $\mathbb{K}(x, y)$ can be expressed as a dot product in a high-dimensional space i.e there exists a function $\Phi(x)$ such that

$$\mathbb{K}(x, y) = \Phi(x) \cdot \Phi(y)$$

Hence the function $\Phi(x)$ is a mapping from the original space into a new possibly much higher dimensional space. The "kernel trick" is essentially the fact that we can get the results of this high dimensional inner product without having to explicitly construct the mapping $\Phi(x)$. The dimension of the space mapped to by Φ might be huge, but the hope is the margin will be large so we can apply the theorem connecting margins and learnability.

2.1.3 Kernels and the Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss Lemma[30] states that a set of n points in a high dimensional Euclidean space can be mapped down into an $O(\log n/\epsilon^2)$ dimensional Euclidean space such that the distance between any two points changes by only a factor of $(1 \pm \epsilon)$.

Arriaga and Vempala [1] use the Johnson-Lindenstrauss Lemma to show that a random linear projection from the ϕ-space to a space of dimension $\tilde{O}(1/\gamma^2)$ approximately preserves linear separability. Balcan, Blum and Vempala [5] then give an explicit algorithm for performing such a mapping. An important point to note is that their algorithm requires access to the distribution where the examples come from in the form of unlabeled data. The upshot is that instead of having the linear separator live in some possibly infinite dimensional space, we can project it into a space whose dimension depends on the margin in the high-dimensional space and where the data is linearly separable if it was linearly separable in the high dimensional space.

2.1.4 A Theory of Learning With Similarity Functions

The mapping discussed in the previous section depended on $\mathbb{K}(x, y)$ being a legal kernel function. In [2] Balcan and Blum show that it is possible to use a similarity function which is not necessarily a legal kernel in a similar way to explicitly map the data into a new space. This mapping also makes use of unlabeled data.

Furthermore, similar guarantees hold: If the data was separable by the similarity function with a certain margin then it will be linearly separable in the new space. The implication is that any valid similarity function can be used to map the data into a new space and then a standard linear separator algorithm can be used for learning.

2.1.5 Winnow

Now we make a slight digression to describe the algorithm that we will be using. Winnow is an online learning algorithm proposed by Nick Littlestone [53]. Winnow starts out with a set of weights and updates them as it sees examples one by one using the following update procedure:

Given a set of weights $w = \{w_1, w_2, w_3, \ldots w_d\} \in \mathbb{R}^d$ and an example $\{x = \{x_1, x_2, x_3, \ldots x_d\} \in \{0,1\}^d\}$

1. If ($w \cdot x \geq d$) then set $y_{pred} = 1$ else set $y_{pred} = 0$. Output y_{pred} as our prediction.

2. Observe the true label $y \in \{0,1\}$ If $y_{pred} = y$ then our prediction is correct and we do nothing. Else if we predicted negative instead of positive, we multiply w_i by $(1 + \epsilon x_i)$ for all i ; if we predicted positive instead of negative then we multiply w_i by $(1 - \epsilon x_i)$ for all i.

An important point to note is that we only update our weights when we make a mistake. There are two main reasons why Winnow is particularly well suited to our task.

1. Our approach is based on augmenting the features of examples with a plethora of extra features. Winnow is known to be particularly effective in dealing with many irrelevant features. In particular, suppose the data has a linear separator of L_1 margin γ. That is, for some weights $w^* = (w_1^*, \ldots, w_d^*)$ with $\sum |w_i^*| = 1$ and some threshold t, all positive examples satisfy $w^* \cdot x \geq t$ and all negative examples satisfy $w^* \cdot x \leq t - \gamma$. Then the number of mistakes the Winnow algorithm makes is bounded by $O(\frac{1}{\gamma^2} \log d)$. For example, if the data is consistent with a majority vote of just r of the d features, where $r \ll d$, then the number of mistakes is just $O(r^2 \log d)$ [53].
2. Experience indicates that unlabeled data becomes particularly useful in large quantities. In order to deal with large quantities of data we will need fast algorithms, Winnow is a very fast algorithm and does not require a large amount of memory.

3.1 Learning with Similarity Functions

Suppose $\mathbb{K}(x, y)$ is our similarity function and the examples have dimension k

We will create the mapping $\Phi(x) : \mathbb{R}^k \to \mathbb{R}^{k+d}$ in the following manner:

1. Draw d examples $\{x_1, x_2, \ldots, x_d\}$ uniformly at random from the dataset.
2. For each example x compute the mapping $x \to \{x, \mathbb{K}(x, x_1), \mathbb{K}(x, x_2), \ldots, \mathbb{K}(x, x_d)\}$

Although the mapping is very simple, in the next section we will see that it can be quite effective in practice.

3.1.1 Choosing a Good Similarity Function

The Naïve approach

We consider as a valid similarity function any function $\mathbb{K}(x, y)$ that takes two inputs in the appropriate domain and outputs a number between -1 and 1. This very general criteria obviously does not constrain us very much in choosing a similarity function.

But we would also intuitively like our similarity function to assign a higher similarity to pairs of examples that are more "similar." In the case where we have positive and negative examples it would seem to be a good idea if our function assigned a higher average similarity to examples that have the same label. One can formalize these intuitive ideas and obtain rigorous criteria for "good" similarity functions [2]. We discuss this in more detail in section 3.1.1

One natural way to construct a similarity function is by modifying an appropriate distance metric. A distance metric takes pairs of objects and assigns them a non-negative real number. If we have a distance metric $D(x, y)$ we can define a similarity function, $\mathbb{K}(x, y)$ as

$$\mathbb{K}(x, y) = \frac{1}{D(x, y) + 1}$$

Then if x and y are close according to distance metric D they will also have a high similarity score. So if we have a suitable distance function on a certain domain the similarity function constructed in this manner can be directly plugged into the Balcan-Blum algorithm.

Scaling issues

It turns out that the approach outlined previously has scaling problems, for example with the number of dimensions. If the number of dimensions is large then the similarity derived from the Euclidian distance between any two objects in a set may end up being close to zero (even if the individual features are boolean). This does not lead to a good performance.

Fortunately there is a straightforward way to fix this issue:

Ranked Similarity

We now describe an alternative way of converting a distance function to a similarity function that addresses the above problem. We first describe it in the transductive case where all data is given up front, and then in the inductive case where we need to be able to assign similarity scores to new pairs of examples as well.

Transductive Classification

1. Compute the similarity as before.
2. For each example x find the example that it is most similar to and assign it a similarity score of 1, find the next most similar example and assign it a similarity score of $(1 - \frac{2}{n-1})$, find the next one and assign it a score of $(1 - \frac{2}{n-1} \cdot 2)$ and so on until the least similar example has similarity score $(1 - \frac{2}{n-1} \cdot (n-1))$. At the end, the most similar example will have a similarity of $+1$, the least similar example will have a similarity of -1, with values spread linearly in between.

This procedure (we will call it "ranked similarity") addresses many of the scaling issues with the naïve approach as each example will have a "full range" of similarities associated with it and experimentally it leads to much to better performance.

Inductive Classification

We can easily extend the above procedure to classifying new unseen examples by using the following similarity function:-

$$\mathbb{K}_S(x,y) = 1 - 2\text{Prob}_{z \sim S}[d(x,z) < d(x,y)]$$

where S is the set of all the labeled and unlabeled examples.

So the similarity of a new example is found by interpolating between the existing examples.

Properties of the ranked similarity

One of the interesting things about this approach is that similarity is no longer **symmetric**, as the similarity is now defined in a way analogous to nearest neighbor. In particular, you may not be the most similar example for the example that is most similar to you. For example, if all points belong to a tight cluster except for an outlier y, the point x closest to y might have the property that $\mathbb{K}(x,y) = -1$ but $\mathbb{K}(y,x) = 1$.

This is notable because this is a major difference with the standard definition of a kernel (as a non-symmetric function is definitely not symmetric positive definite) and provides an example where the similarity function approach gives more flexibility than kernel methods.

Comparing Similarity Functions

One way of comparing how well a similarity function is suited to a particular dataset is by using the notion of a strongly (ϵ, γ)-good similarity function as defined by Balcan and Blum [2]. We say that $\mathbb{K}$ is a strongly (ϵ, γ)- good similarity function for a learning problem P if at least a $(1-\epsilon)$ probability mass of examples x satisfy $E_{x' \sim P}[\mathbb{K}(x', x) | l(x') = l(x)] \geq E_{x' \sim P}[\mathbb{K}(x', x) | l(x') \neq l(x)] + \gamma$ (i.e most examples are more similar to examples that have the same label).

We can easily compute the margin γ for each example in the dataset and then plot the examples by decreasing margin. If the margin is large for most examples, this is an indication that the similarity function may perform well on this particular dataset.

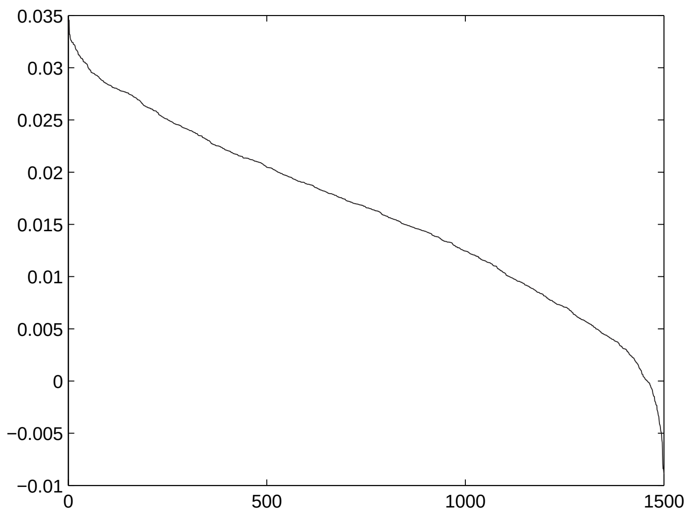

Figure 3.1: The Naïve similarity function on the Digits dataset

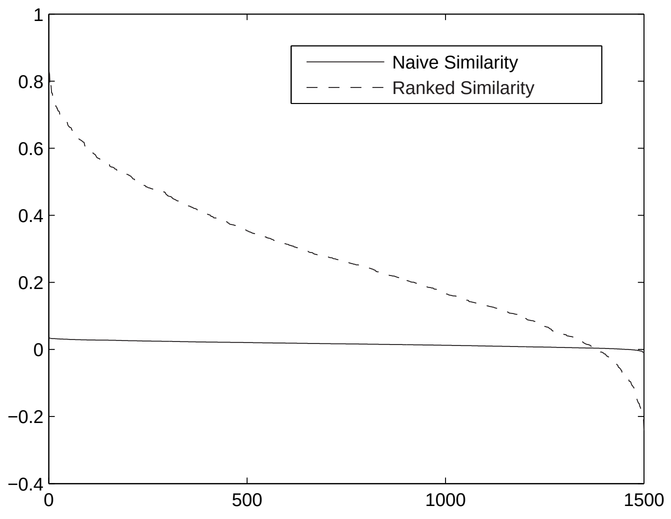

Figure 3.2: The ranked similarity and the naïve similarity plotted on the same scale

Comparing the naïve similarity function and the ranked similarity function on the Digits dataset we can see that the ranked similarity function leads to a much higher margin on most of the examples and experimentally we found that this also leads to a better performance.

4.1 Experimental Results on Synthetic Datasets

To gain a better understanding of the algorithm we first performed some experiments on synthetic datasets. On synthetic datasets we found that both naïve similarity and ranked similarity performed similarly, hence we only show naïve similarity for these datasets.

4.1.1 Synthetic Dataset:Circle

The first dataset we consider is a circle as shown in Figure 4.3 Clearly this dataset is not linearly separable. The interesting question is whether we can use our mapping to map it into a linearly separable space.

We trained it on the original features and on the induced features. This experiment had 1000 examples and we averaged over 100 runs. Error bars correspond to 1 standard deviation. The results are given in figure 4.4.

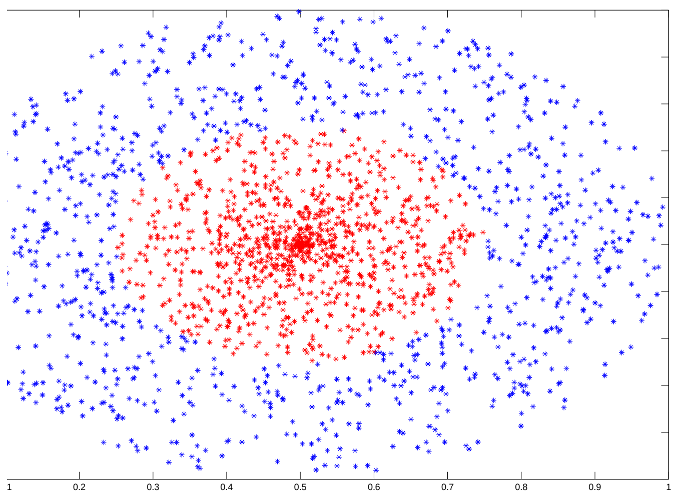

Figure 4.3: The Circle Dataset

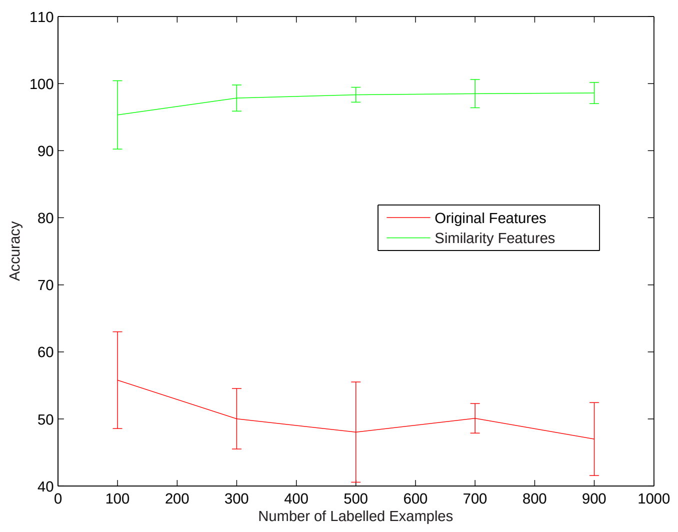

Figure 4.4: Performance on the circle dataset

4.1.2 Synthetic Dataset:Blobs and Line

We expect the original features to do well if the features are linearly separable and we expect the similarity induced features to do particularly well if the data is clustered in well-separated "blobs". One interesting question is what happens if data satisfies neither of these conditions overall, but has some portions satisfying one and some portions satisfying the other.

We generated this dataset in the following way:

1. We selected k points to be the centers of our blobs and randomly assign them labels in $-1, +1$.
2. We then repeat the following process n times:

a We flip a coin.

b If it comes up heads then we set x to random boolean vector of dimension d and $y = x_1$ (the first coordinate of x).

c If it comes up tails then we pick one of the k centers and flip r bits and set x equal to the result and set y equal to the label of the center.

The idea is that the data will be of two types, 50% is completely linearly separable in the original features and 50% is composed of several blobs. Neither one of the feature spaces by themselves should be able to represent the combination well, but the features combined should be able to work well.

As before we trained the algorithm on the original features and on the induced features. But this time we also combined the original and induced features and trained on that. This experiment had 1000 examples and we averaged over 100 runs. Error bars correspond to 1 standard deviation. The results are seen in figure 4.5.

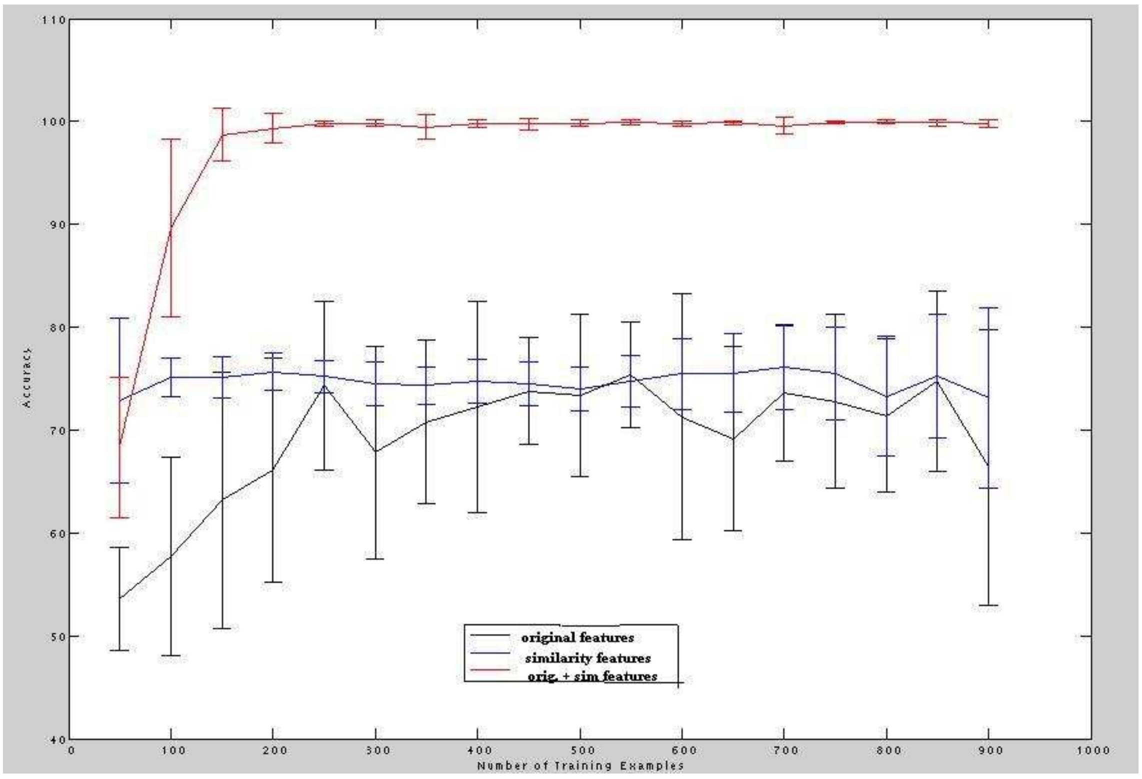

Figure 4.5: Accuracy vs training data on the Blobs and Line dataset

As expected both the original features and the similarity features get about 75% accuracy (getting almost all the examples of the appropriate type correct and about half of the examples of the other type correct) but the combined features are almost perfect in their classification accuracy. In particular this example shows that in at least some cases there may be advantages to augmenting the original features with additional features as opposed to just using the new features by themselves.

5.1 Experimental Results on Real Datasets

To test the applicability of this method we ran experiments on UCI datasets. Comparison with Winnow, SVM and NN (1 Nearest Neighbor) is included. We used ranked similarity for these experiments as it seemed to perform better.

5.1.1 Experimental Design

For Winnow, NN, Sim and Sim+Winnow each result is the average of 10 trials. On each trial we selected 100 training examples at random and used the rest of the examples as test data. We selected 200 random examples as landmarks on each trial.

5.1.2 Winnow

We implemented Balanced Winnow with update rule $(1 \pm e^{-\epsilon X_i})$. ϵ was set to $.5$ and we ran through the data 5 times on each trial (cutting ϵ by half on each pass).

5.1.3 Boolean Features

Experience suggests that Winnow works better with boolean features, so we preprocessed all the datasets to make the features boolean. We did this by computing a median for each column and setting all features less than or equal to the median to 0 and all features greater than or equal to the median to 1.

5.1.4 Booleanize Similarity Function

We also wanted the booleanize the similarity function features. We did this by selecting for each example the 10% most similar examples and setting their similarity to 1 and setting the rest to 0.

5.1.5 SVM

For the SVM experiments we used Thorsten Joachims SVM_{light} [46] with the standard settings.

5.1.6 NN

We assign each unlabeled example the same label as that of the closest example in Euclidean distance.

In Table 5.1 below, we present the results of these algorithms on a range of UCI datasets. In this table, n is the total number of data points, d is the dimension of the space, and nl is the number of labeled examples.We highlight all performances within 5% of the best for each dataset in bold.

5.1.7 Results

In Table 5.1 below, we present the results of these algorithms on a range of UCI datasets. In this table, n is the total number of data points, d is the dimension of the space, and nl is the number of labeled examples.We highlight all performances within 5% of the best for each dataset in bold.

Dataset	n	d	nl	Winnow	SVM	NN	SIM	Winnow+SIM
Congress	435	16	100	**93.79**	**94.93**	**90.8**	**90.90**	**92.24**
Webmaster	582	1406	100	**81.97**	71.78	72.5	69.90	**81.20**
Credit	653	46	100	**78.50**	55.52	61.5	59.10	**77.36**
Wisc	683	89	100	**95.03**	**94.51**	**95.3**	**93.65**	**94.49**
Digit1	1500	241	100	73.26	88.79	**94.0**	**94.21**	**91.31**
USPS	1500	241	100	71.85	74.21	**92.0**	86.72	**88.57**

Table 5.1: Performance of similarity functions compared with standard algorithms on some real datasets

We can observe that on certain types of datasets such as the Webmaster dataset (a dataset of documents) a linear separator like Winnow performs particularly well, while standard Nearest Neighbor does not perform as well. But on other datasets such as USPS(a dataset comprised of images) Nearest Neighbor performs much better than any linear separator algorithm. The important thing to note is that the combination of Winnow plus the similarity features always manages to perform almost as well as the best available algorithm.

6.1 Concatenating Two Datasets

In the section 5.4 we looked at some purely synthetic datasets. An interesting idea is to consider a “hybrid” dataset obtained by combining two distinct real datasets. This models a dataset which is composed of two disjoint subsets that are part of a larger category.

We ran an experiment combining the Credit dataset and the Digit1 dataset. We combined the two datasets by padding each example with zeros so they both ended up with the same number of dimensions as seen in the table below:

Credit (653×46)	Padding (653×241)
Padding (653×46)	Digit1 (653×241)

Table 6.2: Structure of the hybrid dataset

We ran some experiments on the combined dataset using the same settings as outlined in the previous section:

Dataset	n	d	nl	Winnow	SVM	NN	SIM	Winnow+SIM
Credit+Digit1	1306	287	100	72.41	51.74	75.46	74.25	**83.95**

Table 6.3: Performance of similarity functions compared with standard algorithms on a hybrid dataset

7.1 Discussion

For the synthetic datasets (Circle and Blobs and Lines) the similarity features are clearly useful and have superior performance to the original features. For the UCI datasets we observe that the combination of the similarity features with the original features is never significantly worse than the best algorithm on any particular dataset. On the“hybrid” dataset the combination of features does significantly better than either on its own.

8.1 Conclusion

In this report we explored techniques for learning using general similarity functions. We experimented with several ideas that have not previously appeared in the literature:-

1. Investigating the effectiveness of the Balcan-Blum approach to learning with similarity functions on real datasets.
2. Combining Graph Based and Feature Based learning Algorithms.
3. Using unlabeled data to help construct a similarity function.

From our results we can conclude that generic similarity functions do have significant potential for practical applications. They are more general than kernel functions and can be more easily understood. In addition by combining feature based and graph based methods we can often get the “best of both worlds.”

9.1 Future Work

One interesting direction would be to investigate designing similarity functions for specific domains. The definition of a similarity function is so flexible that it allows great freedom to experiment and design similarity functions that are specifically suited for particular domains. This is not as easy to do for kernel functions which have stricter requirements.

Another interesting direction would be to model some realistic theoretical guarantees relating the quality of a similarity function to the performance of the algorithm.

Bibliography

[1] Rosa I. Arriaga and Santosh Vempala. Algorithmic theories of learning. In *Foundations of Computer Science*, 1999. 2.1.3

[2] M.-F. Balcan and A. Blum. On a theory of learning with similarity functions. *ICML06, 23rd International Conference on Machine Learning*, 2006. (document), 1.1.1, 1.1.2, 2.1.4, 3.1.1, 3.1.1

[3] M.-F. Balcan, A. Blum, and S. Vempala. Kernels as features: On kernels, margins and low-dimensional mappings. *ALT04, 15th International Conference on Algorithmic Learning Theory,* pages 194—205.

[4] Maria-Florina Balcan and Avrim Blum. A pac-style model for learning from labeled and unlabeled data. In *In Proceedings of the 18th Annual Conference on Computational Learning Theory (COLT*, pages 111–126. COLT, 2005.

[5] M.F. Balcan, A. Blum, and S. Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 65(1):79–94, 2006. (document), 1.1.1, 1.1.2, 2.1.3

[6] R. Bekkerman, A. McCallum, and G. Huang. categorization of email into folders: Benchmark experiments on enron and sri corpora,. Technical Report IR-418, University of Massachusetts,, 2004.

[7] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006. 1.1.2

[8] G.M. Benedek and A. Itai. Learnability with respect to a fixed distribution. *Theoretical Computer Science*, 86:377—389, 1991.

[9] K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems 10*, pages 368—374. MIT Press, 1998.

[10] T. De Bie and N. Cristianini. Convex methods for transduction. In *Advances in Neural Information Processing Systems 16*, pages 73—80. MIT Press, 2004.

[11] T. De Bie and N. Cristianini. Convex transduction with the normalized cut. Technical Report 04-128, ESAT-SISTA, 2004.

[12] A. Blum. Empirical support for winnow and weighted majority algorithms: results on a calendar scheduling domain. *ICML*, 1995.

[13] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts.

In *Proceedings of the 18th International Conference on Machine Learning*, pages 19—26. Morgan Kaufmann, 2001. 1.1.2

[14] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, July 1998.

[15] A. Blum, J. Lafferty, M. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. *ICML04, 21st International Conference on Machine Learning,* 2004. 1.1.2

[16] Avrim Blum. Notes on machine learning theory: Margin bounds and luckiness functions. http://www.cs.cmu.edu/ avrim/ML08/lect0218.txt, 2008. 2.1.1

[17] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *IEEE Computer Vision and Pattern Recognition Conference*, June 1998.

[18] U. Brefeld, T. Gaertner, T. Scheffer, and S. Wrobel. Efficient co-regularized least squares regression. *ICML06, 23rd International Conference on Machine Learning,* 2006.

[19] A. Broder, R. Krauthgamer, and M. Mitzenmacher. Improved classification via connectivity information. In *Symposium on Discrete Algorithms,* January 2000.

[20] J. I. Brown, Carl A. Hickman, Alan D. Sokal, and David G. Wagner. Chromatic roots of generalized theta graphs. *J. Combinatorial Theory, Series B*, 83:272—297, 2001.

[21] Vitor R. Carvalho and William W. Cohen. Notes on single-pass online learning. Technical Report CMU-LTI-06-002, Carnegie Mellon University, 2006.

[22] Vitor R. Carvalho and William W. Cohen. Single-pass online learning: Performance, voting schemes and online feature selection. In *Proceedings of International Conference on Knowledge Discovery and Data Mining (KDD 2006)*.

[23] V. Castelli and T.M. Cover. The relative value of labeled and unlabeled samples in pattern-recognition with an unknown mixing parameter. *IEEE Transactions on Information Theory*, 42(6):2102—2117, November 1996.

[24] C.Cortes and M.Mohri. On transductive regression. In *Advances in Neural Information Processing Systems 18*. MIT Press, 2006.

[25] S. Chakrabarty, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, 1998.

[26] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`.

[27] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[28] F.G. Cozman and I. Cohen. Unlabeled data can degrade classification performance of generative classifiers. In *Proceedings of the Fifteenth Florida Artificial Intelligence Research Society Conference*, pages 327—331, 2002.

[29] I. Dagan, Y. Karov, and D. Roth. Mistake driven learning in text categorization. In *EMNLP*, pages 55—63, 1997.

[30] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical report, 1999. 2.1.3

[31] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1—38, 1977.

[32] Luc Devroye, Laszlo Györfi, and Gabor Lugosi. *A Probabilistic Theory of Pattern Recognition (Stochastic Modelling and Applied Probability)*. Springer, 1997. ISBN 0387946187. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0387946187`.

[33] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[34] M. Dyer, L. A. Goldberg, C. Greenhill, and M. Jerrum. On the relative complexity of approximate counting problems. In *Proceedings of APPROX'00, Lecture Notes in Computer Science 1913*, pages 108—119, 2000.

[35] Y. Freund, Y. Mansour, and R.E. Schapire. Generalization bounds for averaged classifiers (how to be a Bayesian without believing). To appear in Annals of Statistics. Preliminary version appeared in Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics, 2001, 2003.

[36] Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1048—1053, Edinburgh, Scotand, August 2005. URL `http://www.cs.technion.ac.il/~gabr/papers/fg-tc-ijcai05.pdf`.

[37] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51(2):271—279, 1989.

[38] Steve Hanneke. An analysis of graph cut size for transductive learning. In *the 23rd International Conference on Machine Learning*, 2006.

[39] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2001. ISBN 0387952845. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0387952845`.

[40] Thomas Hofmann. Text categorization with labeled and unlabeled data: A generative model approach. In *NIPS 99 Workshop on Using Unlabeled Data for Supervised Learning*, 1999.

[41] J.J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:550—554, 1994.

[42] M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM Journal on Computing*, 22:1087—1116, 1993.

[43] M. Jerrum and A. Sinclair. The Markov chain Monte Carlo method: An approach to approximate counting and integration. In D.S. Hochbaum, editor, *Approximation algorithms for* NP-*hard problems*. PWS Publishing, Boston, 1996.

[44] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the* 20^{th} *International Conference on Machine Learning (ICML)*, pages 290—297, 2003. 1.1.2

[45] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the* 16^{th} *International Conference on Machine Learning (ICML)*, 1999.

[46] Thorsten Joachims. *Making large-Scale SVM Learning Practical*. MIT Press, 1999. 5.1.5

[47] David Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4), 1996.

[48] J. Kleinberg. Detecting a network failure. In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pages 231—239, 2000.

[49] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *40th Annual Symposium on Foundations of Computer Science*, 2000.

[50] J. Kleinberg, M. Sandler, and A. Slivkins. Network failure detection and graph connectivity. In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms*, pages 76—85, 2004.

[51] Paul Komarek and Andrew Moore. Making logistic regression a core data mining tool: A practical investigation of accuracy, speed, and simplicity. Technical Report CMU-RI-TR-05-27, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2005. 1.1.2

[52] John Langford and John Shawe-Taylor. PAC-bayes and margins. In *Neural Information Processing Systems*, 2002.

[53] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 1988. 1.1.2, 2.1.1, 2.1.5, 1

[54] D. McAllester. PAC-bayesian stochastic model selection. *Machine Learning*, 51(1):5—21, 2003.

[55] Ha Quang Minh, Partha Niyogi, and Yuan Yao. Mercer's theorem, feature maps, and smoothing. In *COLT*, pages 154–168, 2006. 2.1.2

[56] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. 1.1.2, 2.1.1

[57] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press, 1998.

[58] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380—393, April 1997.

[59] Joel Ratsaby and Santosh S. Venkatesh. Learning from a mixture of labeled and unlabeled examples with parametric side information. In *Proceedings of the 8th Annual Conference on Computational Learning Theory*, pages 412—417. ACM Press, New York, NY, 1995.

[60] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323—2326, 2000.

[61] Sebastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *International Conference on Computer Vision (ICCV'98)*, pages 492—499, January 1998.

[62] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2002. 1.1.1, 1.1.2, 2.1.1

[63] Bernhard Schölkopf and Mingrui Wu. Transductive classification vis local learning regularization. In *AISTATS*, 2007. 1.1.2

[64] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521813972. 1.1.1, 1.1.2, 2.1.1

[65] John Shawe-Taylor and Nello Cristianini. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, 1999. 1.1.1, 1.1.2, 2.1.1

[66] John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 44:1926–1940, 1998. (document), 1.1.2, 2.1.1

[67] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 731—737, 1997.

[68] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularized approach to semi-supervised learning with multiple views. *Proc. of the 22nd ICML Workshop on Learning with Multiple Views*, 2005.

[69] Dan Snow, Paul Viola, and Ramin Zabih. Exact voxel occupancy with graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2000.

[70] Nathan Srebro. personal communication, 2007.

[71] Josh Tenenbaum, Vin de Silva, and John Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2000.

[72] S. Thrun, T. Mitchell, and J. Cheng. The MONK's problems. a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, December 1991.

[73] UCI. Repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html, 2000.

[74] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[75] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[76] J.-P Vert, H. Saigo, and T. Akutsu. Local alignment kernels for biological sequences. In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, *Kernel methods in Computational Biology*, pages 131—154. MIT Press, Boston, 2004. 1.1.1

[77] Larry Wasserman. *All of Statistics : A Concise Course in Statistical Inference (Springer Texts in Statistics)*. Springer, 2004. ISBN 0387402721. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0387402721`.

[78] Larry Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer, 2007. ISBN 0387251456. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&amp;path=ASIN/0387251456`.

[79] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15:1101—1113, 1993.

[80] Tong Zhang and Frank J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proc. 17th International Conf. on Machine Learning*, pages

1191–1198, 2000.

[81] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[82] Z.-H. Zhou and M. Li. Semi-supervised regression with co-training. *International Join Conference on Artificial Intelligence(IJCAI)*, 2005.

[83] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.

[84] X. Zhu. Semi-supervised learning with graphs. 2005. Doctoral Dissertation. 1.1.2

[85] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912—919, 2003. 1.1.2