Learning a Two-Stage SVM/CRF Sequence Classifier

Guilherme Hoefel
Computer Science and Engineering
University of California, San Diego
La Jolla, California 92093-0404
ghoefel@cs.ucsd.edu

Charles Elkan
Computer Science and Engineering
University of California, San Diego
La Jolla, California 92093-0404
elkan@cs.ucsd.edu

ABSTRACT

Learning a sequence classifier means learning to predict a sequence of output tags based on a set of input data items. For example, recognizing that a handwritten word is "cat", based on three images of handwritten letters and on general knowledge of English letter combinations, is a sequence classification task. This paper describes a new two-stage approach to learning a sequence classifier that is (i) highly accurate, (ii) scalable, and (iii) easy to use in data mining applications. The two-stage approach combines support vector machines (SVMs) and conditional random fields (CRFs). It is (i) highly accurate because it benefits from the maximum-margin nature of SVMs and also from the ability of CRFs to model correlations between neighboring output tags. It is (ii) scalable because the input to each SVM is a small training set, and the input to the CRF has a small number of features, namely the SVM outputs. It is (iii) easy to use because it combines existing published software in a straightforward way. In detailed experiments on the task of recognizing handwritten words, we show that the two-stage approach is more accurate, or faster and more scalable, or both, than leading other methods for learning sequence classifiers, including max-margin Markov networks (M3Ns) and standard CRFs.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications—*data mining.*

General Terms

Algorithms.

Keywords

Conditional random fields, support vector machines, sequence learning.

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.
Copyright 2008 ACM 978-1-59593-991-3/08/10 ...$5.00.

1. INTRODUCTION

A highly active and successful direction of research in machine learning in the last seven years concerns methods for what is called structured learning [11, 16, 18, 17, 15, 5]. Structured learning means learning to predict outputs that have internal structure. This structure can be modeled, and, to achieve high predictive accuracy, it must be modeled. Learning to predict a sequence of output tags, given a sequence of input data items, is an example of a structured learning problem. Specifically, suppose the input is a sequence of images where each image is a bitmap of a handwritten letter. A traditional supervised learning approach is to train a function that can recognize the letter encoded by each image separately. In this traditional approach, the trained classifier recognizes each letter in isolation, based only on the information available in the corresponding image. In a structured learning approach, given the sequence of images representing the letters in a word, a single trained model recognizes all the letters of the word, using all the input images *and* using knowledge learned about which letters tend to be adjacent in English. For example, suppose the word to be recognized is "fern." The handwritten third and fourth letters may well be almost identical, so a traditional classifier might recognize this word as "fenn" or "ferr" or "fenr". A sequence classifier would use probabilistic constraints between neighboring output letters to know that "fern" is more likely than the alternatives, even though the alternatives are an equally good fit to the input data at the level of individual letters.

Research on structured learning has seen great progress, with sequence classification as its most important and successful subfield. Indeed, the original paper on conditional random fields (CRFs) has been cited over 1100 times since it was published in 2001 [11]. However, technology transfer from basic research to applications has been limited so far. Accelerating this technology transfer is the goal of this paper. We show that existing software that is high in quality and easy to use, specifically the well-known SVM package named LIBSVM [3] and a new CRF package named CRF-SGD [1], can be used together to achieve high accuracy and high speed on a sequence classification task that so far has been addressed only using complex custom methods that are effectively out of reach for practitioners.

In other words, the goal of the work described here is to show how to benefit from state-of-the-art methods in machine learning by combining them in an uncomplicated way. Frank Lloyd Wright once wrote "'think simple' as my old master used to say–meaning reduce the whole of its parts into the simplest terms, getting back to first principles." Our goal in this paper is to combine multiple theoretical ideas in order to obtain one easy-to-use high-performance method. Following the principle of reducing the whole of its parts into the simplest terms, we reduce the problem of learning a sequence classifier into two subproblems.

The new learning framework is called a two-stage SVM/CRF method. It simplifies ideas introduced previously under the name max-margin Markov networks (M3Ns) [16]. Essentially, we first use SVMs to learn to predict the labels of individual input sequence data items. Then, we use a CRF to predict the sequence of all output labels, where the input to the CRF is the outputs of the SVMs applied to the inputs.[1] The two-stage method gains high accuracy from two complementary strengths: margin-maximization approaches can be more accurate than likelihood-maximization approaches as discriminative classifiers, and learning correlations between neighboring output labels helps resolve ambiguities.

Because our goal is to present a method that practitioners can use easily in multiple other applications, our experiments use off-the-shelf software. As an implementation of SVMs, we use the LIBSVM package [3]. As an implementation of CRFs, we use the very recent CRFSGD package [1]. The latter software is especially interesting, and fast, because it solves the numerical optimization problem at the core of CRFs by stochastic gradient descent, following but simplifying much recent research [19].

In experiments we compare the two-stage method against three baseline methods. The first two baselines treat the problem as unstructured; they are standard logistic regression (LR) and SVMs [3]. The third baseline does not use the margin-maximization idea; it is a standard CRF classifier. In addition to the two-stage SVM/CRF approach, we also investigate a similar two-stage LR/CRF method. Previous studies have shown that different sets of feature-functions lead to widely varying accuracy for CRFs [10]. Hence we investigate a range of alternative sets of feature-functions.

2. THE TWO-STAGE SVM/CRF METHOD

The M3N method combines maximum-margin and output-correlation constraints into a single quadratic programming optimization problem [16]. In addition to the mathematical challenges of combining these two types of constraints, this approach is computationally intensive [13], although algorithms that are faster than the original M3N method have been proposed [17]. The two-stage approach that we introduce has an intuitive rationale that is similar to that of previous max-margin sequence prediction methods, but the new approach is notably simpler mathematically and computationally.

In our approach, first SVMs are trained to predict the label of each input sequence element; this is a standard multiclass supervised learning task. Second, one CRF is trained to predict the output sequence of labels using as its input the outputs from the previously trained SVMs. The intuition is that both learning approaches are somewhat orthogonal in their advantages, so a combination of them can yield superior results.

During SVM training, the goal is to learn each class based on each sequence element (i.e. data item or data point) and its label in the training set, by maximizing the separation between data points with labels in the same class and other data points. Many studies have shown that SVMs tend to obtain superior results, compared to other classifiers, for predicting individual labels. This advantage of SVMs stems from their ability to use high-dimensional feature spaces via kernels, and from theoretical guarantees on generalization ability [16]. However, an important drawback is that it is typically hard to choose the settings for an SVM (in particular, the best value for the soft-margin penalty C) that will yield obtain optimum results. The most common way to choose settings is to use a validation set that is independent from the training and testing sets.

Given a data point in the test set, the output of the trained SVMs is a vector of scores. In the second stage of our approach, this vector is used as the input attributes for a CRF classifier. Traditionally, a feature-function for a CRF is based on one or more data points, and one label or two adjacent labels. Our proposed new type of feature-function is based on a prediction vector of scores for a data point, instead of directly on the attributes of the data point. Essentially, the two-stage approach uses SVMs as a feature induction method, in order to allow a CRF to learn a better overall classifier.

Let X be a set of input sequences and let Y be the corresponding set of sequences of labels. The data (X, Y) consist of samples $(\bar{x}_i, \bar{y}_i)$ for $i = 1, ..., n$. Each sample $(\bar{x}_i, \bar{y}_i)$ consists of $L(i)$ data points and their labels. That is

$$(\bar{x}_i, \bar{y}_i) = \langle(\mathbf{x}_{i1}, y_{i1}), (\mathbf{x}_{i2}, y_{i2}), ..., (\mathbf{x}_{iL(i)}, y_{iL(i)})\rangle.$$

A label y_{ij} can belong to one of c different classes, and each input data point $\mathbf{x}_{ij}$ can have p dimensions, where p is the number of pixels in the image of one character for example. We assume that each dimension can have one of v values.

Our experiments use an optical character recognition (OCR) dataset compiled by Kassel [9] and standardized by Taskar [16], who performed image segmentation to separate the characters in each word, rasterization, and normalization of each character. Previous papers do not mention any further data manipulation such as dimensionality reduction. It is well known that dimensionality reduction can be very important in image processing, but we do not investigate it here.

3. MULTICLASS CLASSIFIERS

For multiclass classification SVMs can be used in either one-against-all or one-against-one fashion. With the one-against-all technique, each class is trained separately against the union of all other classes. Applying the trained SVMs on a test data point $(\mathbf{x}_{ij}, y_{ij})$ yields a vector of prediction scores $(g_1, g_2, ..., g_c)_{ij}$, where c is the number of classes. With the one-against-one technique, each class is trained separately against each other class. Applying the trained SVMs to the test data point yields a vector of prediction scores $(g_1, g_2, ..., g_b)_{ij}$ where $b = c(c-1)/2$.

Previous work [16] has indicated that the one-against-one approach yields slightly more accurate results for the OCR data. There are two additional advantages of using this approach as part of the two-stage SVM/CRF method: it yields faster SVM training, and it increases the bandwidth of information passed to the CRF. Although one-against-one

[1] In previous work, the outputs of other learning methods have been used as the input to an SVM [6, 12], but our approach is the opposite: the output of multiple SVMs is used as the input to another learning method.

training is conducted $c(c-1)/2$ times, each time only the data points in two classes are involved. SVM training time is typically superlinear in the number of training examples, so learning more classifiers each with a smaller training set is a net win. This improvement in running time is proportional to the number of alternative labels ($c = 26$ if labels are letters in the alphabet), so it is considerable. The increase in communication bandwidth between the SVMs and the CRF can potentially improve the accuracy achievable by the CRF. However, the larger number of inputs for the CRF tends to increase its training time.

When used for multiclass classification, logistic regression classifiers produce similar vectors of scores, which can also be used as inputs to a CRF in a second stage. For LR training we use another off-the-shelf tool, the MATLABArsenal package [20]. With logistic regression, each vector of scores is a non-normalized vector of probabilities. With support vector machines, each vector is a collection of scores with numerical values between -12.0 and 5.0.

4. CONDITIONAL RANDOM FIELDS

Given a dataset of input and output sequences (X, Y), the training objective for a CRF model is to choose parameters W (also called weights) that maximize the conditional log likelihood $\log P(Y|X;W)$, which is

$$\sum_{(\bar{x}_i,\bar{y}_i)\in(X,Y)} \log \frac{\exp \sum_{z=1}^{d} w_z F_z(\bar{x}_i,\bar{y}_i)}{\sum_{\bar{y}'} \exp \sum_{z=1}^{d} w_z F_z(\bar{x}_i,\bar{y}')}.$$

Here there are d different fixed feature-functions denoted F_z for $z = 1,\ldots,d$. There is one trainable parameter w_z for each F_z. Each feature-function F_z is actually a sum over output sequence positions of a lower-level feature-function f_z. That is, each high-level feature-function F_z has the form

$$F_z(\bar{x}_i,\bar{y}_i) = \sum_j f_z(\mathbf{x}_{ij}, y_{ij-1}, y_{ij})$$

where j ranges over the elements of $\bar{y}_i$ and y_{i0} is a special token to represent the beginning of a sequence.

Although the lower-level functions f_z can in general be real-valued, all the f_z functions we use are binary, i.e. they have value 0 or 1. Each f_z function can depend on any or all of the input sequence, and/or on up to two adjacent labels in the output sequence $\bar{y}_i$. The reason why only at most two adjacent output labels can be used is that making predictions efficiently with a trained CRF model depends on the Viterbi algorithm to compute

$$\text{argmax} \sum_{z=1}^{d} w_z F_z(\bar{x}_i,\bar{y}_i)$$

and this algorithm cannot handle lower-level feature-functions that involve more than two adjacent elements of $\bar{y}_i$.

We investigate multiple alternative CRF designs that differ in which feature-functions they use. The alternative CRFs that we consider use various combinations of the following six types of feature-function, which are all special cases of the general form above.

Feature-functions of the first type have the form

$$F_z^{(1)}(\bar{x}_i,\bar{y}_i) = \sum_j f_z^{(1)}(\mathbf{x}_{ij}, y_{ij}).$$

There are $c \cdot v \cdot p$ functions of this type, because there are c possible values for y_{ij}, v attributes of $\mathbf{x}_{ij}$, and p possible values for each attribute.

Feature-functions of the second type have the form

$$F_z^{(2)}(\bar{x}_i,\bar{y}_i) = \sum_j f_z^{(2)}(\mathbf{x}_{ij}, y_{ij-1}, y_{ij}).$$

The number of functions of this type is c^2vp.

When dealing with the OCR dataset, previous work suggests that using features that depend only on output labels is beneficial. In particular, the best results of [10, Section 3, Table 2] are obtained using $F_z^{(1)}$ features in addition to features that use just a single label, and just two adjacent labels. We represent these feature types as follows:

$$F_z^{(3)}(\bar{x}_i,\bar{y}_i) = \sum_j f_z^{(3)}(y_{ij})$$

and

$$F_z^{(4)}(\bar{x}_i,\bar{y}_i) = \sum_j f_z^{(4)}(y_{ij-1}, y_{ij}).$$

There are c features of the former type, and c^2 of the latter type.

Our contribution is to introduce features for the two-stage approach that depend on the data point $\mathbf{x}_{ij}$ only indirectly, through prediction scores $g_z(\mathbf{x}_{ij})$ assigned by SVM classifiers. We formalize this idea as follows:

$$F_z^{(5)}(\bar{x}_i,\bar{y}_i) = \sum_j f_z^{(5)}(g_z(\mathbf{x}_{ij}), y_{ij})$$

and

$$F_z^{(6)}(\bar{x}_i,\bar{y}_i) = \sum_j f_z^{(6)}(g_z(\mathbf{x}_{ij}), y_{ij-1}, y_{ij})$$

where $g_z(\mathbf{x}_{ij})$ is one element of the score vector produced by the multiclass SVM classifier applied to $\mathbf{x}_{ij}$.

Real-valued SVM scores are discretized, in order to allow the $f_z^{(5)}$ and $f_z^{(6)}$ feature-functions to be binary. Specifically, only the most significant digit is taken into account. Given a real-valued score $g_z(\mathbf{x}_{ij})$, the integer value that is used as input to the feature-function is

$$g'_z(\mathbf{x}_{ij}) = \lceil g_z(\mathbf{x}_{ij}) \rceil.$$

Each different integer value, for each of the binary SVM classifiers, then gives rise to a different binary feature-function. When logistic regression is used instead of SVMs, scores are probabilities between 0 and 1, so we use

$$g'_z(\mathbf{x}_{ij}) = \lceil 10 \cdot g_z(\mathbf{x}_{ij}) \rceil$$

instead.

As is customary with CRFs, we in fact maximize a regularized version of the conditional log likelihood, that is

$$J(X,Y) = \log P(Y|X;W) + \log P(W)$$

where $\log P(W) = -\frac{\|W\|_2}{2\sigma^2}$. Often the regularization parameter σ is set using a validation dataset, but in our experiments it is fixed at $\sigma = 1$.

The objective function is maximized by gradient descent. The gradient $(\partial/\partial w_z^{(l)})J(X,Y)$ is

$$\sum_{(\bar{x},\bar{y})\in(X,Y)} F_z^{(l)}(\bar{x},\bar{y}) - \sum_{\bar{y}'} p(\bar{y}'|\bar{x}; w_z^{(l)}) F_z^{(l)}(\bar{x},\bar{y}') - \frac{2w_z^{(l)}}{\sigma}$$

for $l \in \{1,2,3,4,5,6\}$. The gradient, for each weight and for each training example $(\bar{x}, \bar{y})$, is essentially the difference between the feature-function value for $(\bar{x}, \bar{y})$ and the average value of the feature-function averaging over each $\bar{y}'$ with probability given by the current model $p(\bar{y}'|\bar{x}; w)$. The CRF software we use, called CRFSGD, does stochastic gradient descent [1]. Our experiments confirm that this approach achieves the same accuracy as a sophisticated quasi-Newton method (L-BFGS, [14]) but is about 10 times faster.

5. PERFORMANCE CRITERIA

Our hypothesis is that the two-stage combined SVM/CRF method just described performs as well as more mathematically and computationally complex methods, in particular the M3N method. In previous papers, Taskar *et al.* and Perez-Cruz *et al.* measure accuracy as the average error per character, but Nguyen *et al.* and Keerthi *et al.* measure accuracy as the average over words of the average error per character in each word. In this paper, we report both measurements, since this is the only way to establish a direct correspondence with previous results. As expected, both definitions of accuracy yield very similar results.

The first definition is

$$AccPerChar = \frac{1}{N} \sum_{(i,j)} I(\hat{y}_{ij} = y_{ij})$$

where N is the total number of characters in the test set, y_{ij} is the true value of the jth character of the ith word in the test set, and $\hat{y}_{ij}$ is the predicted value of this character. The second definition is

$$AccPerWord = \frac{1}{M} \sum_{i=1}^{M} [\frac{1}{L(i)} \sum_{j=1}^{L(i)} I(\hat{y}_{ij} = y_{ij})]$$

where M is the total number of words and $L(i)$ is the total number of characters in the ith word.

6. EXPERIMENTS

The specific dataset used for experiments here is a subset containing 6876 words from the OCR dataset of [9]. This subset was compiled by Ben Taskar, and is precisely the same dataset used previously [16, 13, 10, 15]. Each character image in the dataset is of size of $8 \cdot 16 = 128$ pixels and is labeled with one of 26 letters. Each pixel has value 0 or 1. To the best of our knowledge, this is the preferred dataset for comparing the performance of classifiers where margin and sequential based approaches are combined, given previously published results that study the matter.

In previous work, Taskar *et al.* used an unusual 10-fold cross-validation technique where they divided the data into training sets of about 610 words and test sets of about 5500 words. This approach is unusual because in each fold, a small set is used for training versus a large set for testing. In standard cross-validation, in each fold a large set is used for training and a small set for testing. Nguyen *et al.* applied a similar nonstandard technique, but they used about 600 words for training, about 5400 words for testing, about 100 words for validation. The precise cardinalities of the subsets used in this previous work is not known.

It seems that the reason previous authors used small training sets is time limitations for training. It has been reported [13, Section 4] that the M3N method needed to be halted after 10 iterations of the optimization algorithm for a single fold. The two-stage approach proposed here is much faster. Therefore traditional 10-fold cross-validation can be used, as done also by Perez-Cruz *et al.* This is desirable because standard cross-validation gives a better idea of the ultimate accuracy that can be achieved by different methods, since it is based on larger training sets.

7. METHODS COMPARED

For the standard unstructured classifiers, logistic regression and SVMs, each input data point is separate and is one array of pixels. Both methods are trained in a one-against-one fashion for solving the multi-class problem, which is the same as done previously by Taskar [16, Section 3]. For logistic regression the regularization constant is set to 1. For soft-margin SVMs, three different kernels are tried: linear, quadratic and cubic.

Changing the soft-margin penalty parameter C typically yields significantly different results for different kernels [2]. In our experiments C is set to be 150, 250, and 450, for the linear, quadratic, and cubic kernels respectively. Other training parameters are set to the defaults from LIBSVM. Notice that the CGM experiments also use LIBSVM [15]. Perez-Cruz *et al.* pick C to be 5, and use a radial basis function kernel.

Standard CRF classifiers are trained using two different sets of feature-functions. The first set consists of the $F_z^{(1)}$ and $F_z^{(2)}$ features. Following Keerthi *et al.*, the second set consists of the $F_z^{(1)}$, $F_z^{(3)}$ and $F_z^{(4)}$ feature-functions. In the first set there are $128 \cdot 2 \cdot 26 = 6656$ $F_z^{(1)}$ functions and $128 \cdot 2 \cdot 26 \cdot 26 = 173056$ $F_z^{(2)}$ functions. In the second set there are 26 $F_z^{(3)}$ functions and $26 \cdot 26 = 676$ $F_z^{(4)}$ functions in addition to the $F_z^{(1)}$ functions.

Two-stage SVM/CRF classifiers are trained using three different sets of feature-functions. The first set includes $F_z^{(5)}$ and $F_z^{(6)}$ feature-functions, and thus corresponds to the M3N approach. The second set contains $F_z^{(3)}$, $F_z^{(4)}$ and $F_z^{(5)}$ feature-functions, so it is analogous to the set of CRF feature-functions that performs best in recent experiments [10]. Finally, the third set combines the original CRF feature-functions $F_z^{(1)}$ and $F_z^{(2)}$ with the novel $F_z^{(5)}$ and $F_z^{(6)}$ feature-functions. After discretization, each SVM score is one of at most 17 unique values. Given the one-against-one approach, each score vector has length $(26 \cdot 25)/2 = 325$. Thus, there are at most $325 \cdot 17 \cdot 26 = 143,650$ $F_z^{(5)}$ functions, and at most $325 \cdot 17 \cdot 26 \cdot 26 = 3,734,900$ $F_z^{(6)}$ functions. The CRFSGD software only keeps features that occur more than three times in the training set, so these feature set cardinalities are upper bounds on the number of features actually used.

8. ACCURACY RESULTS

Tables 1 and 2 show accuracy results using nonstandard cross-validation, that is with a small 10% training set in each fold, while Tables 3 and 4 show results using standard cross-validation, with a large 90% training set in each fold. Results are presented as mean accuracy plus/minus standard deviation over ten folds. Rows in italics are results taken from previous papers. If a method from a previous paper does not appear in a table, it is because the previous paper did not report the corresponding performance metric, or did not use the corresponding type of cross-validation. Standard deviations are given where available. Results from Taskar *et al.* appear with two places of accuracy only since they are obtained from a figure in that paper. Finally, Table 5 presents the number of seconds needed to run one fold of cross validation for each method.

The first unstructured baseline, the logistic regression classifier, performs better than previously reported. The improvement may be due to the fact that we use the one-against-one approach. In results not shown, when running logistic regression in one-against-all fashion, our results are the same as previously found by Taskar.

The SVM classifiers based on LIBSVM produce interesting results compared to previous experiments. They yield slightly better accuracy than has been reported by Taskar *et al.*, Nguyen *et al.*, and Perez-Cruz *et al.* The differences may be due to the challenge of setting the soft-margin penalty parameter adequately. In Taskar's work, a multiclass kernel-vector machine [4] is used for the linear, quadratic and polynomial kernels. The results from that method closely match the performance obtained here using LIBSVM.

Nguyen *et al.* use two types of SVM, called SVM^{struct} [8] and $SVM^{multiclass}$, which are both based on the SVM^{light} quadratic optimizer [7]. Notice that Nguyen *et al.* only show results for SVMs with linear kernels, which perform worse than SVMs with polynomial kernels in this domain. SVM^{struct} performs better than $SVM^{multiclass}$ in their experiments; its accuracy is close to the accuracy we can obtain using polynomial kernels. Perez-Cruz *et al.* use the same LIBSVM package that we do; their results using a radial basis function kernel are similar to ours using a linear kernel. Clearly, so far polynomial kernels are the best known for this domain.

Our first baseline method for structured learning, a CRF classifier with feature types $F_z^{(1,2)}$, performs better than the CRF of Taskar *et al.* by around 3 percentage points, and much better than the CRF of Nguyen *et al.*, beating it by 10 percentage points. This big difference in accuracy is likely due to differences choosing features for the CRF. The CRFSGD software lets us efficiently use a large number of feature-functions, which is known to be beneficial for the success of this type of classifier.

Our second CRF baseline uses the feature-functions suggested by Keerthi *et al.*, namely the types $F_z^{(1,3,4)}$. These are token-dependent first-order and token-independent first-order and second-order according to their nomenclature. The results in this case are similar to previous findings.

Last but not least, the results for the novel two-stage approach are very promising. Overall this approach does better than logistic regression, SVM, and CRF methods separately, and offers accuracy similar to that of the more complex M3N and CGM methods. Using feature-functions that are token-dependent ($F_z^{(5,6)}$ or $F_z^{(1,2,5,6)}$) seems to be important in obtaining a good two-stage classifier.

Results with the two-stage logistic regression/CRF method are better than results with either method alone, and almost as good as the best results obtained with the M3N method. Although both logistic regression and CRFs are based on maximizing the conditional log-likelihood of a linear model, supplying the logistic regression vector of probability estimates to the CRF appears to enhance its ability to solve the problem. Presumably the vector of scores makes explicit information that is only implicit in the original data.

Table 1: Small training sets: average accuracy per character.

Method	Accuracy
Taskar's LR	*.71*
LR	.7589 ± .0028
Taskar's SVM (linear)	*.71*
Taskar's SVM (quadr.)	*.80*
Taskar's SVM (cubic)	*.81*
CGM (Graph1)	*.7290 ± .0009*
SVM (linear)	.7334 ± .0049
SVM (quadr.)	.8257 ± .0034
SVM (cubic)	.8204 ± .0029
Taskar's CRF	*.76*
CRF $F_z^{(1,2)}$	.7926 ± .0042
CRF $F_z^{(1,3,4)}$	.7945 ± .0080
LR/CRF $F_z^{(3,4,5)}$	.8136 ± .0022
LR/CRF $F_z^{(5,6)}$	.8512 ± .0032
LR/CRF $F_z^{(1,2,5,6)}$	.8559 ± .0026
Taskar's M3N (linear)	*.80*
SVM/CRF (linear) $F_z^{(3,4,5)}$	.8116 ± .0022
SVM/CRF (linear) $F_z^{(5,6)}$	.8592 ± .0037
SVM/CRF (linear) $F_z^{(1,2,5,6)}$	.8659 ± .0039
Taskar's M3N (quadr.)	*.87*
CGM (Graph2)	*.8750 ± .0011*
SVM/CRF (quadr.) $F_z^{(3,4,5)}$	.8214 ± .0032
SVM/CRF (quadr.) $F_z^{(5,6)}$	.8825 ± .0025
SVM/CRF (quadr.) $F_z^{(1,2,5,6)}$	.8819 ± .0061
Taskar's M3N (cubic)	*.87*
SVM/CRF (cubic) $F_z^{(3,4,5)}$	.8088 ± .0022
SVM/CRF (cubic) $F_z^{(5,6)}$	.8685 ± .0025
SVM/CRF (cubic) $F_z^{(1,2,5,6)}$	.8757 ± .0024
CGM (Graph3)	*.9420 ± .0005*

Table 2: Small training sets: average accuracy per character per word.

Method	Accuracy
LR	.7594 ± .0032
Nguyen's SVM (linear)	*.7146*
Nguyen's SVM^{struct} (linear)	*.7884*
Keerthi's SVM^{struct} (linear)	*.8076*
SVM (linear)	.7341 ± .0050
SVM (quadr.)	.8263 ± .0039
SVM (cubic)	.8210 ± .0033
Nguyen's CRF	*.6770*
Keerthi's CRF	*.8003*
CRF $F_z^{(1,2)}$	.7924 ± .0062
CRF $F_z^{(1,3,4)}$	.7930 ± .0093
LR/CRF $F_z^{(3,4,5)}$	.8139 ± .0031
LR/CRF $F_z^{(5,6)}$	.8519 ± .0042
LR/CRF $F_z^{(1,2,5,6)}$	.8557 ± .0035
Nguyen's M3N	*.7492*
SVM/CRF (linear) $F_z^{(3,4,5)}$	.8107 ± .0030
SVM/CRF (linear) $F_z^{(5,6)}$	.8589 ± .0044
SVM/CRF (linear) $F_z^{(1,2,5,6)}$	.8660 ± .0046
SVM/CRF (quadr.) $F_z^{(3,4,5)}$	.8205 ± .0035
SVM/CRF (quadr.) $F_z^{(5,6)}$	.8810 ± .0019
SVM/CRF (quadr.) $F_z^{(1,2,5,6)}$	.8808 ± .0051
SVM/CRF (cubic) $F_z^{(3,4,5)}$	.8073 ± .0046
SVM/CRF (cubic) $F_z^{(5,6)}$	.8677 ± .0027
SVM/CRF (cubic) $F_z^{(1,2,5,6)}$	.8737 ± .0024

Table 3: Large training sets: average accuracy per character.

Method	Accuracy
LR (linear)	.8182 ± .0041
CGM (Graph1)	*.8740 ± .0009*
SVM (linear)	.8135 ± .0014
SVM (quadr.)	.9003 ± .0040
SVM (cubic)	.9051 ± .0039
CRF $F_z^{(1,2)}$	.8379 ± .0051
CRF $F_z^{(1,3,4)}$	.8562 ± .0089
LR/CRF $F_z^{(3,4,5)}$	.9037 ± .0037
LR/CRF $F_z^{(5,6)}$	.9264 ± .0066
LR/CRF $F_z^{(1,2,5,6)}$	.9214 ± .0062
SVM/CRF (linear) $F_z^{(3,4,5)}$	.8962 ± .0042
SVM/CRF (linear) $F_z^{(5,6)}$	.9114 ± .0038
SVM/CRF (linear) $F_z^{(1,2,5,6)}$	.9082 ± .0056
CGM (Graph2)	*.9690 ± .0003*
SVM/CRF (quadr.) $F_z^{(3,4,5)}$	.9270 ± .0048
SVM/CRF (quadr.) $F_z^{(5,6)}$	.9500 ± .0038
SVM/CRF (quadr.) $F_z^{(1,2,5,6)}$	.9450 ± .0032
SVM/CRF (cubic) $F_z^{(3,4,5)}$	.9237 ± .0063
SVM/CRF (cubic) $F_z^{(5,6)}$	.9468 ± .0042
SVM/CRF (cubic) $F_z^{(1,2,5,6)}$	.9424 ± .0051
CGM (Graph3)	*.9730 ± .0004*

Table 4: Large training sets: average accuracy per character per word.

Method	Accuracy
LR	.8194 ± .0042
SVM (linear)	.8118 ± .0016
SVM (quadr.)	.9018 ± .0038
SVM (cubic)	.9066 ± .0044
CRF $F_z^{(1,2)}$	.8372 ± .0054
CRF $F_z^{(1,3,4)}$	.8586 ± .0086
LR/CRF $F_z^{(3,4,5)}$	.9019 ± .0029
LR/CRF $F_z^{(5,6)}$	.9209 ± .0069
LR/CRF $F_z^{(1,2,5,6)}$	.9190 ± .0087
SVM/CRF (linear) $F_z^{(3,4,5)}$	.8924 ± .0057
SVM/CRF (linear) $F_z^{(5,6)}$	.9066 ± .0042
SVM/CRF (linear) $F_z^{(1,2,5,6)}$	.9034 ± .0073
SVM/CRF (quadr.) $F_z^{(3,4,5)}$	.9205 ± ,0037
SVM/CRF (quadr.) $F_z^{(5,6)}$	.9485 ± .0037
SVM/CRF (quadr.) $F_z^{(1,2,5,6)}$	.9435 ± .0069
SVM/CRF (cubic) $F_z^{(3,4,5)}$	.9229 ± .0050
SVM/CRF (cubic) $F_z^{(5,6)}$	.9463 ± .0015
SVM/CRF (cubic) $F_z^{(1,2,5,6)}$	.9416 ± .0060

The performance of the two-stage SVM/CRF method is good. Its accuracy is comparable to that of the M3N method when using features based on the vector of scores and on adjacent labels ($F_z^{(5,6)}$). The two-stage SVM/CRF also performs as well as the CGM method with cliques of size 2, which is the fair comparison. The CGM method with cliques of size 3 obtains the best overall results. This make sense because there is definitely useful information in triples of letters over and above the information in pairs of letters. For example, while "st" and "th" are both common letter pairs in English, the triplet "sth" is rare.

Tables 3 and 4 show that using traditional cross-validation, with a large training set in each fold, leads to significantly improved accuracy. With this setup, all methods do 5 to 10 percentage points better than with a smaller training set. In summary, Tables 1 to 4 together show that the two-stage approach, with information from either logistic regression or SVMs provided as input to a CRF, yields the same accuracy as mathematically more complex methods.

9. TIMING RESULTS

Previous studies do not mention the time required to conduct experiments. Table 5 shows the number of seconds needed to run one fold of cross-validation for each of our methods, with small and with big training sets. The entries in the table for LR/CRF and SVM/CRF are the time needed by the CRF stage for these approaches. Thus, the total time for the SVM/CRF two-stage approach is the sum of the SVM and SVM/CRF entries. The computers used for Table 5 are quite standard and inexpensive (Redhat Linux EL4, dual P4 3.2GHz, single CPU used, 2GB memory).

As expected, logistic regression training is fastest, while SVM training is slowest. Given that the larger training set is 9 times bigger, a ratio of running times of 9 or less can be considered reasonable. The observed ratio is reasonable for all methods, except for SVM training with a linear kernel.

Table 5: Time in seconds for one fold of training and testing.

Method	Small	Large
LR (linear)	195	403
SVM (linear)	1540	62524
SVM (quadr.)	3184	9520
SVM (cubic)	2780	13770
CRF $F_z^{(1,2)}$	447	2308
CRF $F_z^{(1,3,4)}$	123	2352
LR/CRF $F_z^{(3,4,5)}$	272	623
LR/CRF $F_z^{(5,6)}$	1296	6591
LR/CRF $F_z^{(1,2,5,6)}$	1802	9318
SVM/CRF (linear) $F_z^{(3,4,5)}$	267	692
SVM/CRF (linear) $F_z^{(5,6)}$	1352	7550
SVM/CRF (linear) $F_z^{(1,2,5,6)}$	1313	10319
SVM/CRF (quadr.) $F_z^{(3,4,5)}$	335	592
SVM/CRF (quadr.) $F_z^{(5,6)}$	1267	6375
SVM/CRF (quadr.) $F_z^{(1,2,5,6)}$	2155	9064
SVM/CRF (cubic) $F_z^{(3,4,5)}$	259	651
SVM/CRF (cubic) $F_z^{(5,6)}$	1228	6279
SVM/CRF (cubic) $F_z^{(1,2,5,6)}$	1718	8930

It is an unfortunate drawback of SVMs that training time often increases more than linearly as the number of training examples increases. This phenomenon is observed here for SVM training with the linear kernel. In future work, we plan to use one of the more recent SVM implementations that tend to be much faster because they use stochastic gradient descent.

10. CONCLUSION

Structured learning is a new research area in machine learning that has not yet seen wide usage in data mining or knowledge discovery. Within the field of structured learning, the most studied task has been how to learn a classifier that maps a sequence of inputs into a sequence of output labels. Above, we have described a practical new approach to training a sequence classifier. Our experiments show that the proposed method achieves high accuracy, and is faster and more scalable than competitors.

The proposed method combines support vector machines and conditional random fields in a two-stage approach. It achieves high accuracy because of the maximum-margin nature of SVMs, and because CRFs can model correlations between neighboring output labels. The SVM stage of the new method is scalable because the input for training each SVM is only a small subset of the entire training data. The CRF stage of the new method is scalable because the CRF uses only a limited number of features, namely the outputs of the SVMs trained in the first stage.

We report the results of detailed experiments on the task of recognizing handwritten words. Our results provide a lot of detail concerning just one dataset, rather than being less detailed but involving multiple datasets. The reason for this choice is partly that a previous comparison paper in this area [13] has been controversial. The results of this particular previous paper show CRFs and the M3N method performing much worse than in the experience of other researchers. The reason for some of the poor results in [13] was uncovered by [10]. Now, we have performed careful and systematic experiments whose results, reported here, supersede those of [13], and will resolve the controversy, we hope.

We feel confident that the good performance obtained on the handwritten word recognition problem by the two-stage method will carry over to other sequential prediction problems. The reason is the orthogonal strengths of the two phases of the two-stage method. In general, a margin-based approach can extract the most important information about individual data points, while a sequential approach can augment the learning process by exposing the sequentially structured nature of the problem. The results above show that the two-stage SVM/CRF method yields greater accuracy than its component individual methods, which are the current practical state of the art. The two-stage method matches closely the accuracy achievable with the M3N and CGM methods, which are more complex mathematically and computationally. For practical purposes, what is most important is that the good SVM/CRF results are obtained using robust off-the-shelf software. This fact means that the proposed SVM/CRF combination is usable immediately by other researchers and practitioners in their application areas.

Acknowledgments: This work was performed while the first author was a student at UCSD and employed by Qualcomm, Inc.

11. REFERENCES

[1] L. Bottou. *CRFSGD software*, 2008. Available at `http://leon.bottou.org/projects/sgd`.

[2] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2007. Available at `http://www.csie.ntu.edu.tw/∼cjlin/libsvm`.

[4] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[5] V. Franc and B. Savchynskyy. Discriminative learning of max-sum classifiers. *Journal of Machine Learning Research*, 9:67–104, 2008.

[6] T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In T. Lengauer, R. Schneider, P. Bork, D. L. Brutlag, J. I. Glasgow, H.-W. Mewes, and R. Zimmer, editors, *ISMB*, pages 149–158. AAAI, 1999.

[7] T. Joachims. *SVM^{light} Support Vector Machine*, 2004. Available at `http://svmlight.joachims.org`.

[8] T. Joachims. *SVM-hmm sequence tagging with structural support vector machines*, 2008. Version 3.03 available at `http://www.cs.cornell.edu/People-/tj/svm_light/svm_hmm.html`.

[9] R. H. Kassel. *A comparison of approaches to on-line handwritten character recognition*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

[10] S. S. Keerthi and S. Sundararajan. CRF versus SVM-Struct for sequence labeling. Technical report, Yahoo Research, 2007.

[11] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

[12] L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Journal of Computational Biology*, 10(6):857–868, 2003.

[13] N. Nguyen and Y. Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 681–688, 2007.

[14] J. Nocedal and S. J. Wright. Limited memory BFGS. In *Numerical Optimization*, pages 222–247. Springer, 1999.

[15] F. Perez-Cruz, Z. Ghahramani, and M. Pontil. Conditional graphical models. In *Predicting Structured Data*, pages 265–282. MIT Press, Cambridge, MA, USA, 2006.

[16] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *NIPS*. MIT Press, 2003.

[17] B. Taskar, S. Lacoste-Julien, and M. I. Jordan. Structured prediction, dual extragradient and Bregman projections. *Journal of Machine Learning Research*, 7:1627–1653, 2006.

[18] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[19] S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML)*, pages 969–976, 2006.

[20] R. Yan. *MATLABArsenal: A Matlab package for classification algorithms*, 2006. Carnegie Mellon University, School of Computer Science.