Kernel Conditional Random Fields: Representation and Clique Selection

John Lafferty LAFFERTY@CS.CMU.EDU
Xiaojin Zhu ZHUXJ@CS.CMU.EDU
Yan Liu YANLIU@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University, Pittsburgh PA, USA

Abstract

Kernel conditional random fields (KCRFs) are introduced as a framework for discriminative modeling of graph-structured data. A representer theorem for conditional graphical models is given which shows how kernel conditional random fields arise from risk minimization procedures defined using Mercer kernels on labeled graphs. A procedure for greedily selecting cliques in the dual representation is then proposed, which allows sparse representations. By incorporating kernels and implicit feature spaces into conditional graphical models, the framework enables semi-supervised learning algorithms for structured data through the use of graph kernels. The framework and clique selection methods are demonstrated in synthetic data experiments, and are also applied to the problem of protein secondary structure prediction.

1. Introduction

Many classification problems involve the annotation of data items having multiple components, with each component requiring a classification label. Such problems are challenging because the interaction between the components can be rich and complex. In text, speech, and image processing, for example, it is often useful to label individual words, sounds, or image patches with categories to enable higher level processing; but these labels can depend on one another in a highly complex manner. For biological sequence annotation, it is desirable to annotate each amino acid in a protein with a label, with the collection of labels representing the global geometric structure of the molecule. Here the labels in principle depend on the physical characteristics of the molecule and its ambient chemical environment. In each case, classification tasks naturally arise which clearly violate the assumption of independent and identically distributed instances that is made in the majority of classification procedures in statistics and machine learning. It is therefore of central importance to extend recent advances in classification theory and practice to structured, non-independent data classification problems.

Conditional random fields (Lafferty et al., 2001) have been proposed as an approach to modeling the interactions between labels in such problems using the tools of graphical models. A conditional random field (CRF) is a model that assigns a joint probability distribution over labels conditional on the input, where the distribution respects the independence relations encoded in a graph. In general, the labels are not assumed to be independent, nor are the observations conditionally independent given the labels, as is assumed in generative models such as hidden Markov models. The CRF framework has already been used to obtain promising results in a number of domains where there is interaction between labels, including tagging, parsing and information extraction in natural language processing (Collins, 2002; Sha & Pereira, 2003; Pinto et al., 2003) and the modeling of spatial dependencies in image processing (Kumar & Hebert, 2003). In related work, Taskar et al. (2003) have studied random fields (also known as Markov networks) fit using loss functions that incorporate a generalized notion of margin, and have observed how the "kernel trick" applies to this family of models.

We present an extension of conditional random fields that permits the use of implicit features spaces through Mercer kernels, using the framework of regularization theory. Such an extension is motivated by the significant body of recent work that has shown kernel methods to be extremely effective in a wide variety of machine learning techniques; for example, they enable the integration of multiple sources of information in a principled manner. Our introduction of Mercer kernels into conditional graphical models is also motivated by the problem of semi-supervised learning. In many domains, the collection of annotated training data is difficult and costly, as it requires the efforts of expert human annotators, while the collection of unlabeled data may

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004.

be relatively easy and inexpensive. The emerging theme in recent research in semi-supervised learning is that kernel methods, in particular those based on graphical representations of unlabeled data, form a theoretically attractive and empirically promising set of techniques for combining labeled and unlabeled data (Belkin & Niyogi, 2002; Chapelle et al., 2002; Smola & Kondor, 2003; Zhu et al., 2003).

In Section 2 we formalize the learning problem and present a version of the classical representer theorem of Kimeldorf and Wahba (1971). Unlike the classical result, for kernel conditional random fields the dual parameters depend on all potential assignments of labels to cliques in the graph, not only the observed labels. This motivates the need for algorithms to derive sparse representations, since the full representation has parameters for each labeled clique in the graphs appearing in the training data. In Section 3 we present a greedy algorithm for selecting a small number of representative cliques. This "clique selection" algorithm parallels the "import vector selection" algorithms of kernel logistic regression (Zhu & Hastie, 2001), and the feature selection methods that have been previously proposed for random fields and conditional random fields using explicit features (McCallum, 2003).

In Section 4 the ideas and methods are demonstrated on two synthetic data sets, where the effects of the underlying graph kernels, clique selection, and sequential modeling can be clearly seen. In Section 5 we report the results of experiments using kernel CRFs for protein secondary structure prediction. This is the task of mapping primary sequences of amino acids onto a string of secondary structure assignments, such as helix, sheet, or coil. It is widely believed that secondary structure can contribute valuable information to discerning how proteins fold in three dimensions. We compare kernel conditional random fields, estimated using clique selection, against support vector machine classifiers, with both methods using kernels derived from position-specific scoring matrices (PSI-BLAST profiles) as input features. In addition, we give results for the use of graph kernels derived from the PSI-BLAST profiles in a transductive, semi-supervised framework for estimating the kernel CRFs. The paper concludes with a brief discussion in Section 6.

2. Representation

Before proceeding with formalism, we give some intuition for what our framework is intended to capture. Our goal is to annotate structured data, where the structure is represented by a graph. Labels are to be assigned to the nodes in the graph in order to minimize some loss function, such as 0-1 error; the labels come from a small set $\mathcal{Y}$, for example, $\mathcal{Y} = \{\texttt{red}, \texttt{blue}, \texttt{green}\}$. Each vertex in the graph is associated with a feature vector $\boldsymbol{x}_v \in \mathcal{X}$. In image processing, the feature vector at a node might include a pixel intensity, as well as average pixel intensities smoothed over neighboring regions using wavelets. In protein secondary structure prediction, each node might correspond to an amino acid in the protein, and the feature vector at a node may include an amino acid histogram of all protein fragments in a database which closely match the given protein at that node. In the following section we present our notation and formal framework for such problems.

2.1. Cliques and labeled graphs

Let $\mathfrak{G}$ denote a collection of finite graphs. For example, $\mathfrak{G}$ might be the set of finite chains, appropriate for sequence modeling, or the rectangular 2-dimensional grids, appropriate for some image processing tasks. The set of vertices of a graph $\mathfrak{g} \in \mathfrak{G}$ is denoted by $V(\mathfrak{g})$, and size of the graph is the number of vertices, denoted $|\mathfrak{g}| = |V(\mathfrak{g})|$. A *clique* is a subset of the vertices which is fully connected, with any pair of vertices joined by an edge; we denote the set of cliques in the graph by $\mathcal{C}(\mathfrak{g})$. The number of vertices in a clique is denoted by $|c|$. Similarly, we denote by $\mathcal{C}(\mathfrak{G}) = \{(\mathfrak{g}, c) \mid \mathfrak{g} \in \mathfrak{G},\ c \in \mathcal{C}(\mathfrak{g})\}$ the collection of cliques across varying graphs. In other words, a member of $\mathcal{C}(\mathfrak{G})$ consists of a graph and a distinguished clique of that graph. We will work with kernels that compare components of different graphs. For example, we could consider a kernel $K : \mathcal{C}(\mathfrak{G}) \times \mathcal{C}(\mathfrak{G}) \to \{0, 1\}$ given by $K((\mathfrak{g}, c), (\mathfrak{g}', c')) = \delta(|c|, |c'|)$.

We next consider labelings of a graph. Let $\mathcal{Y}$ be a finite set of labels; infinite $\mathcal{Y}$ is also possible in a regression framework, but we restrict to finite $\mathcal{Y}$ for simplicity. The set of $\mathcal{Y}$-labelings of a graph $\mathfrak{g}$ is denoted $\mathcal{Y}(\mathfrak{g}) = \{\boldsymbol{y} \mid \boldsymbol{y} \in \mathcal{Y}^{|\mathfrak{g}|}\}$, and the collection of all $\mathcal{Y}$-labeled graphs is $\mathcal{Y}(\mathfrak{G}) = \{(\mathfrak{g}, \boldsymbol{x}) \mid \mathfrak{g} \in \mathfrak{G},\ \boldsymbol{y} \in \mathcal{Y}(\mathfrak{g})\}$. Similarly, let $\mathcal{X}$ be an input feature space; for example, $\mathcal{X} = \mathbb{R}^n$. The set $\mathcal{X}(\mathfrak{g}) = \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathcal{X}^{|\mathfrak{g}|}\}$ denotes the set of assignments of a feature vector to each vertex of the graph $\mathfrak{g}$; $\mathcal{X}(\mathfrak{G}) = \{(\mathfrak{g}, \boldsymbol{x}) \mid \mathfrak{g} \in \mathfrak{G},\ \boldsymbol{x} \in \mathcal{X}(\mathfrak{g})\}$ is the collection of all such annotated graphs. Finally, let $\mathcal{Y}_{\mathcal{C}}(\mathfrak{g}) = \{(c, \boldsymbol{y}_c) \mid c \in \mathcal{C}(\mathfrak{g}),\ \boldsymbol{y}_c \in \mathcal{Y}^{|c|}\}$ be the set of $\mathcal{Y}$-labeled cliques in a graph. As above, we similarly define $\mathcal{X}\mathcal{Y}_{\mathcal{C}}(\mathfrak{g}) = \{(\boldsymbol{x}, c, \boldsymbol{y}_c) \mid \boldsymbol{x} \in \mathcal{X}(\mathfrak{g}),\ (c, \boldsymbol{y}_c) \in \mathcal{Y}_{\mathcal{C}}(\mathfrak{g})\}$ and $\mathcal{X}\mathcal{Y}_{\mathcal{C}}(\mathfrak{G}) = \{(\mathfrak{g}, \boldsymbol{x}, c, \boldsymbol{y}_c) \mid (\boldsymbol{x}, c, \boldsymbol{y}_c) \in \mathcal{X}\mathcal{Y}_{\mathcal{C}}(\mathfrak{g})\}$.

2.2. Representer Theorem

The prediction task for conditional graphical models is to learn a function $h : \mathcal{X}(\mathfrak{G}) \to \mathcal{Y}(\mathfrak{G})$ where $h(\mathfrak{g}, \boldsymbol{x}) \in \mathcal{Y}(\mathfrak{g})$ is a labeling of $\mathfrak{g}$, with the goal of minimizing a suitably defined loss function. The classifier $h = h_n$ is chosen based on a labeled sample $\{(\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{n}$, with each $(\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ being a labeled graph, the graph possibly changing from example to example.

To limit the complexity of the hypothesis, we will assume that it is determined completely by a function $f : \mathcal{XY}_{\mathcal{C}}(\mathfrak{G}) \to \mathbb{R}$. Let $f(\mathfrak{g}, \boldsymbol{x})$ denote the collection of values $\{f(\mathfrak{g}, \boldsymbol{x}, c, \boldsymbol{y}_c)\}$, with $c \in \mathcal{C}(\mathfrak{g})$ varying over the cliques of $\mathfrak{g}$ and $\boldsymbol{y}_c \in \mathcal{Y}^{|c|}$ varying over all possible labelings of that clique. We assume that a loss function $\phi(\boldsymbol{y}, f(\mathfrak{g}, \boldsymbol{x}))$ is given. As an important example, and the loss function used in this paper, consider the *negative log loss*

$$\phi(\boldsymbol{y}, f(\mathfrak{g}, \boldsymbol{x})) = \tag{1}$$

$$-\sum_{c \in \mathcal{C}(\mathfrak{g})} f_c(\boldsymbol{x}, \boldsymbol{y}_c) + \log \sum_{\boldsymbol{y}' \in \mathcal{Y}(\mathfrak{g})} \exp\left(\sum_{c \in \mathcal{C}(\mathfrak{g})} f_c(\boldsymbol{x}, \boldsymbol{y}'_c)\right)$$

where $f_c(\boldsymbol{x}, \boldsymbol{y}_c)$ is shorthand for $f(\mathfrak{g}, \boldsymbol{x}, c, \boldsymbol{y}_c)$. The negative log *marginal* loss could also be considered for minimizing the per-node error. The negative log loss function corresponds to a conditional random field given by

$$p(\boldsymbol{y} \mid \mathfrak{g}, \boldsymbol{x}) = Z^{-1}(\mathfrak{g}, \boldsymbol{x}, f) \exp\left(\sum_c f_c(\boldsymbol{x}, \boldsymbol{y}_c)\right) \tag{2}$$

We now discuss how the "representer theorem" of kernel machines (Kimeldorf & Wahba, 1971) applies to conditional graphical models. While this is a simple extension, we're not aware of an analogous formulation in the statistics or machine learning literature.

Let K be a Mercer kernel on $\mathcal{XY}_{\mathcal{C}}(\mathfrak{G})$; thus

$$K((\mathfrak{g}, \boldsymbol{x}, c, \boldsymbol{y}_c), (\mathfrak{g}', \boldsymbol{x}', c', \boldsymbol{y}'_{c'})) \in \mathbb{R}$$

for each $(\boldsymbol{x}, c, \boldsymbol{y}_c) \in \mathcal{XY}_{\mathcal{C}}(\mathfrak{g})$ and $(\boldsymbol{x}', c', \boldsymbol{y}'_{c'}) \in \mathcal{XY}_{\mathcal{C}}(\mathfrak{g}')$. Intuitively, this assigns a measure of similarity between a labeled clique in one graph and a labeled clique in a (possibly) different graph. We denote by $\mathcal{H}_K$ the associated reproducing kernel Hilbert space, and by $\|\cdot\|_K$ the associated norm on $L^2(\mathcal{XY}_{\mathcal{C}}(\mathfrak{G}))$.

Consider a regularized loss function of the form

$$R_\phi f = \sum_{i=1}^n \phi\left(\boldsymbol{y}^{(i)}, f(\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)})\right) + \Omega\left(\|f\|_K\right)$$

It is important to note that the loss depends on all possible assignments $\boldsymbol{y}_c$ of labels to each clique, not just those observed in the labeled data $\boldsymbol{y}^{(i)}$. Suppressing the dependence on the graph $\mathfrak{g}$ in the notation, let $K_c(\boldsymbol{x}, \boldsymbol{y}_c;, \cdot) = K((\mathfrak{g}, \boldsymbol{x}, c, \boldsymbol{y}_c), \cdot)$. Following the argument for the standard representer theorem, it can easily be shown that the minimizer of a regularized loss function of the above form can be expressed in terms of the basis functions $f(\cdot) = K(\cdot, (\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)}, c, \boldsymbol{y}_c))$.

Proposition (Representer theorem for CRFs). *Let K be a Mercer kernel on $\mathcal{XY}_{\mathcal{C}}(\mathfrak{G})$ with associated RKHS norm $\|\cdot\|_K$, and let $\Omega : \mathbb{R}_+ \to \mathbb{R}_+$ be strictly increasing. Then the minimizer $f^\star$ of*

$$R_\phi f = \sum_{i=1}^n \phi\left(\boldsymbol{y}^{(i)}, f(\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)})\right) + \Omega\left(\|f\|_K\right)$$

if it exists, has the form

$$f^\star(\cdot) = \sum_{i=1}^n \sum_{c \in \mathcal{C}(\mathfrak{g}^{(i)})} \sum_{\boldsymbol{y}_c \in \mathcal{Y}^{|c|}} \alpha_c^{(i)}(\boldsymbol{y}_c) K_c(\boldsymbol{x}^{(i)}, \boldsymbol{y}_c; \cdot)$$

The key property distinguishing this result from the standard representer theorem is that the "dual parameters" $\alpha_c^{(i)}(\boldsymbol{y}_c)$ now depend on *all* assignments of labels.

2.3. Two special cases

Let $\overline{K}$ be a Mercer kernel on $\mathcal{Z} = \mathcal{X} \times \mathcal{Y} \times \mathcal{Y}$. Thus, the kernel is defined in terms of the matrix entries $\overline{K}(\boldsymbol{z}, \boldsymbol{z}')$ where $\boldsymbol{z} = (x, y_1, y_2)$. Using $\overline{K}$ we can define a kernel on edges in $\mathcal{XY}_{\mathcal{C}}(\mathfrak{G})$ by $K((\mathfrak{g}, \boldsymbol{x}, (v_1, v_2), (y_1, y_2)), (\mathfrak{g}', \boldsymbol{x}', (v'_1, v'_2), (y'_1, y'_2))) = \overline{K}((\boldsymbol{x}_{v_1}, y_1, y_2), (\boldsymbol{x}'_{v'_1}, y'_1, y'_2))$. For the regularized risk minimization problem

$$\min_{f \in \mathcal{H}_K} R_\phi(\boldsymbol{x}, f, \lambda) = \min_{f \in \mathcal{H}_K} \sum_{i=1}^n \phi(\boldsymbol{y}^{(i)}, f(\boldsymbol{x}^{(i)})) + \lambda \|f\|_K$$

where $f \in \mathcal{H}_K$, the CRF representer theorem implies that the solution $f^\star$ has the form

$$f^\star_{(v_1,v_2)}(\boldsymbol{x}, y_1, y_2) = \sum_{i=1}^n \sum_{y,y'} \sum_{(v,v')} \alpha^{(i)}_{(v,v')}(y, y') \overline{K}((\boldsymbol{x}_{v_1}, y_1, y_2), (\boldsymbol{x}_v^{(i)}, y, y'))$$

In the special case of kernel $\overline{K}(\boldsymbol{z}, \boldsymbol{z}') = \overline{K}(x, x')\,\delta(y_1, y'_1)$ it follows that

$$f^\star_{(v_1,v_2)}(\boldsymbol{x}, y_1, y_2) = \sum_{i=1}^n \sum_{v \in V(\mathfrak{g}^{(i)})} \alpha_v^{(i)}(y_1)\,\overline{K}(\boldsymbol{x}_{v_1}, \boldsymbol{x}_v^{(i)})$$

Under the probabilistic model (2), this is simply kernel logistic regression. In the special case of $\overline{K}(\boldsymbol{z}, \boldsymbol{z}') = \overline{K}(x, x')\,\delta(y_1, y'_1) + \delta(y_1, y'_1)\,\delta(y_2, y'_2)$ we get that

$$f^\star_{(v_1,v_2)}(\boldsymbol{x}, y_1, y_2) = \sum_{i,v} \alpha_v^{(i)}(y_1)\,\overline{K}(\boldsymbol{x}_v^{(i)}, \boldsymbol{x}_{v_1}) + \alpha(y_1, y_2)$$

and we recover a simple type of semiparametric CRF.

3. Clique Selection

The representer theorem shows that the minimizing function f is supported by labeled cliques over the training examples; however, this may result in an extremely large number of parameters. We therefore pursue a strategy of incrementally selecting cliques in order to greedily reduce the regularized risk. The resulting procedure is parallel to forward stepwise logistic regression, and to related methods for kernel logistic regression (Zhu & Hastie, 2001), as well as to the greedy selection procedure presented in (Della Pietra et al., 1997).

Our algorithm will maintain an *active set* $\mathcal{A} = \{(\mathfrak{g}^{(i)}, c, \boldsymbol{y}_c)\} \subset \mathcal{Y}_{\mathcal{C}}(\mathfrak{G})$ of labeled cliques, where the labelings are not restricted to those appearing in the training data. Each such candidate clique can be represented by a basis function $h(\cdot) = K((\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)}, c, \boldsymbol{y}_c), \cdot) \in \mathcal{H}_K$, and is assigned a parameter $\alpha_h = \alpha_c^{(i)}(\boldsymbol{y}_c)$. We work with the regularized risk

$$R_\phi(f) = \sum_i \phi\left(\boldsymbol{y}^{(i)}, f(\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)})\right) + \frac{\lambda}{2}\|f\|_K^2 \qquad (3)$$

where ϕ is the log-loss of equation (1). To evaluate a candidate h, one strategy is to compute the *gain* $\sup_\alpha R_\phi(f) - R_\phi(f + \alpha h)$, and to choose the candidate h having the largest gain. This presents an apparent difficulty, since the optimal parameter α cannot be computed in closed form, and must be evaluated numerically. For sequence models this involves forward-backward calculations for each candidate h, the cost of which is prohibitive.

As an alternative, we adopt the functional gradient descent approach, which evaluates a small change to the current function. For a given candidate h, consider adding h to the current model with small weight ε; thus $f \mapsto f + \varepsilon h$. Then $R_\phi(f + \varepsilon h) = R_\phi(f) + \varepsilon dR_\phi(f, h) + O(\varepsilon^2)$, where the functional derivative of R_ϕ at f in the direction h is computed as

$$dR_\phi(f, h) \quad = \quad E_f[h] - \widetilde{E}[h] + \lambda \langle f, h\rangle_K$$

where $\widetilde{E}[h] = \sum_i h(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ is the empirical expectation and $E_f[h] = \sum_i \sum_{\boldsymbol{y}} p(\boldsymbol{y} \,|\, \boldsymbol{x}^{(i)}, f)\, h(\boldsymbol{x}^{(i)}, \boldsymbol{y})$ is the model expectation conditioned on $\boldsymbol{x}$, combined with the empirical distribution on $\boldsymbol{x}$. The idea is that in directions h where the functional gradient $dR_\phi(f, h)$ is large, the model is mismatched with the labeled data; this direction should be added to the model to make a correction. This results in the greedy clique selection algorithm summarized in Figure 1.

Following our earlier notation,

$$h(\boldsymbol{x}^{(i)}, \boldsymbol{y}) = \sum_{c \in \mathcal{C}(\mathfrak{g}^{(i)})} h(\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)}, c, \boldsymbol{y}_c)$$

is the sum over all cliques. The candidate functions h might include functions of the form

$$h(\cdot) = K((\mathfrak{g}^{(i)}, \boldsymbol{x}^{(i)}, c, \boldsymbol{y}_c), \cdot)$$

Initialize with $f = 0$, and iterate:

1. For each candidate $h \in \mathcal{H}_K$, supported by a single labeled clique, calculate the functional derivative $dR_\phi(f, h)$.
2. Select the candidate $h = \arg\max_h |dR_\phi(f, h)|$ having the largest gradient direction. Set $f \mapsto f + \alpha_h h$.
3. Estimate parameters α_f for each active f by minimizing $R_\phi(f)$.

Figure 1. Greedy Clique Selection. Labeled cliques encode basis functions h which are greedily added to the model, using a form of functional gradient descent.

where i is a specific instance, c is a particular clique of $\mathfrak{g}^{(i)}$, and y_c is a labeling of that clique. Alternatively, in a slightly less greedy manner, at each step in the selection procedure a specific instance and clique may be selected, and functions for *each* clique labeling may be added.

In the experiments reported below for sequences, the marginal probabilities $p(\boldsymbol{y}_t = y \,|\, \boldsymbol{x})$ and expected counts for the state transitions are required; these are computed using the forward-backward algorithm, with log domain arithmetic to avoid underflow. A quasi-Newton method (BFGS, cubic-polynomial line search) is used to estimate the parameters in step 3. Prediction is carried out using the forward-backward algorithm to compute marginals rather than using the Viterbi algorithm.

3.1. Combining multiple kernels

The above use of kernels enables semi-supervised learning for structured prediction problems. One of the emerging themes in semi-supervised learning is that graph kernels can provide a useful framework for combining labeled and unlabeled data. Here an undirected graph is defined over labeled and unlabeled data instances, and generally the assumption is that labels vary smoothly over the graph. The graph is represented by the weight matrix W, and one can construct a kernel from the graph Laplacian, substituting eigenvalues λ by $r(\lambda)$, where r is a non-negative and (typically) decreasing function. This regularizes high frequency components and encourages smooth functions on the graph; see (Smola & Kondor, 2003) for a description of this unifying view of graph kernels.

It is important to note that such a use of a graph kernel for semi-supervised learning introduces an *additional* graphical structure, which should not be confused with the graph representing the explicit dependencies between labels in a CRF. For example, when modeling sequences, the natural CRF graph structure is a chain. By incorporating unlabeled data through the use of a graph kernel, an additional graph that will generally have many cycles is implicitly introduced. However, the graph kernel and a more standard kernel may be naturally combined as a linear combination; see, for example, (Lanckriet et al., 2004).

4. Synthetic Data Experiments

To demonstrate the properties and advantages of KCRFs, we prepared two synthetic datasets: a "galaxy" dataset to investigate the relation to semi-supervised and sequential learning, and an HMM with Gaussian mixture emission probabilities to demonstrate the properties of clique selection and the advantages of incorporating kernels.

Galaxy. The "galaxy" dataset is a variant of two spirals; see Figure 2 (left). Note the dense core of points from both classes. The sequences are generated from a 2-state hidden Markov model (HMM), where each state emits instances uniformly from one of the classes. There is a 90% chance of staying in the same state. The idea is that under a sequence model, an example from the core will have a better than random chance to be labeled correctly based on the context. This is not true under a non-sequence model, and the dataset as a whole will thus have about a 20% Bayes error rate under the iid assumption. We sample 100 sequences of length 20. Note the choice of semi-supervised vs. standard kernels and sequence vs. non-sequence models are orthogonal; the four combinations are all tested on. We construct a semi-supervised graph kernel by first creating an unweighted 10-nearest neighbor graph. We then compute the graph Laplacian L, and form the kernel $K = 10\left(L + 10^{-6}I\right)^{-1}$. This corresponds to a function $r(\lambda) = 1/(\lambda + 10^{-6})$ on L's eigenvalues. The standard kernel is the radial basis function (RBF) kernel with bandwidth $\sigma = 0.35$. All parameters here and below are tuned by cross validation.

Figure 2 (center) shows the results of using kernel logistic regression with the semi-supervised kernel and with the RBF kernel; here the sequence structure is ignored. For each training set size, which ranges from 20 to 400 points, 10 random trials were performed. The error intervals shown are one standard error. When the labeled set size is small, the graph kernel is much better than the RBF kernel. However both kernels saturate at the 20% Bayes error rate.

Next we apply both kernels to the semiparametric KCRF model in section 2.3; see Figure 2 (right). Note the x-axis is the number of training sequences–since each sequence has 20 instances, the range is the same as Figure 2 (center). The kernel CRF is capable of getting under the 20% Bayes error floor of the non-sequence model, with both kernels and sufficient labeled data. However, the graph kernel is able to learn the structure much faster than the RBF kernel. Evidently, the high error rate for low label data sizes prevents the RBF model from effectively using the context.

HMM with Gaussian mixtures. This more difficult dataset is generated from a 3-state HMM. Each state is a mixture of 2 Gaussians with random mean and covariance. The Gaussians strongly overlap; see Figure 3 (left). The transition probabilities favor remaining in the state, with a probability of 0.8, and to transition to each of the other two states with equal probability 0.1; we generate 100 sequences of length 30. We use an RBF kernel with $\sigma = 0.5$. (A graph kernel is slightly worse than the RBF kernel on this dataset, and is not shown.) We perform 20 trials for each training set size, and in each trial we perform clique selection to select the top 20 vertices. The center and right plots in Figure 3 show that the semiparametric KCRF again outperforms kernel logistic regression with the same RBF kernel.

Figure 4 shows clique selection, with a training size 20 sequences, averaged over 20 random trials. The regularized risk (left), which is training set likelihood plus regularizor, always decreases as we select more vertices into the KCRF. On the other hand, the test set likelihood (center) and accuracy (right) saturate and even worsen slightly, showing signs of overfitting. All curves change dramatically at first, demonstrating the effectiveness of the clique selection algorithm. In fact, fewer than 10 vertex cliques are sufficient for this problem.

5. Protein Secondary Structure Prediction

For the protein secondary structure prediction task, we used the RS126 dataset, on which many current methods have been developed and tested (Cuff & Barton, 1999). It is a non-homologous dataset, since among the 126 protein chains, no two proteins share more than 25% sequence identity over a length of more than 80 residues (Cuff & Barton, 1999). The dataset can be downloaded from `http://barton.ebi.ac.uk/`.

We adopt the DSSP definition of protein secondary structure (Kabsch & Sander, 1983), which is based on hydrogen bonding patterns and geometric constraints. Following the discussion in (Cuff & Barton, 1999), the 8 DSSP labels are reduced to a 3 state model as follows: H & G map to helix (H), E & B to sheets (E), and all other states to coil (C).

The state-of-the-art performance for secondary structure prediction is achieved by window-base methods, using the position-specific scoring matrices (PSSM) as input features, i.e., PSI-BLAST profiles, together with Support Vector Machines (SVMs) as the underlying learning algorithm (Jones, 1999; Kim & Park, 2003). Finally, the raw predictions are fed into a second layer SVM to filter out physically unrealistic predictions, such as one sheet residue surrounded by helix residues (Jones, 1999).

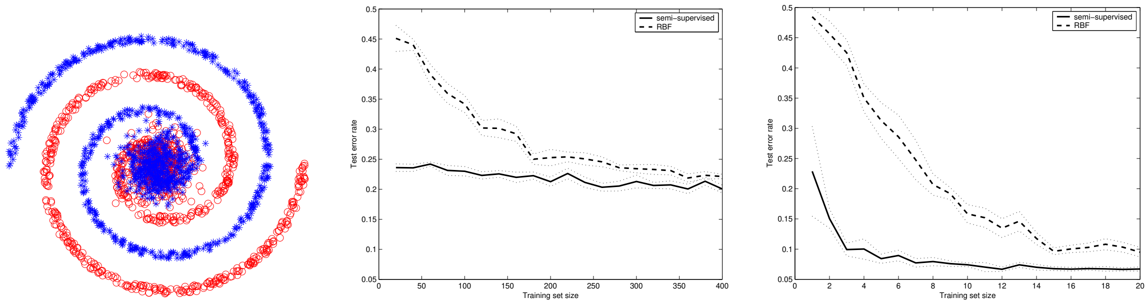

Figure 2. Left: The galaxy data. Center: *Kernel logistic regression*, comparing two kernels: RBF and a graph kernel using the unlabeled data. Right: *Kernel conditional random fields*, which take into account the sequential structure of the data.

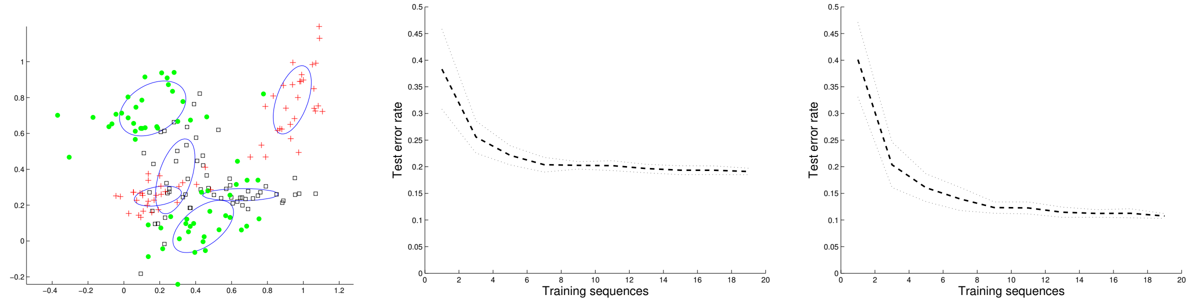

Figure 3. Left: The Gaussian mixture data (only a few data points are shown). Center: Kernel logistic regression with an RBF kernel. Right: Kernel CRF with the same kernel.

In our experiments, we apply a linear transformation L to the PSSM matrix elements according to $L(x) = 0$ for $x \leq -5$, $L(x) = \frac{1}{2} + \frac{x}{10}$ for $-5 \leq x \leq 5$, and $L(x) = 1$ for $x \geq 5$. This is the same transform used by Kim and Park (2003), which achieved one of the best results in the recent CASP (Critical Assessment of Structure Predictions) competition. The window size is set to 13 by cross-validation. Therefore the number of features per position is 13×21 (the number of amino acids plus gap).

Clique selection. We use an RBF kernel with bandwidth $\sigma = 0.1$ chosen by cross-validation. Figure 5 (left) shows the kernel CRF risk reduction as clique selection proceeds, when only vertex clique candidates are allowed (note there are always position independent edge parameters in the KCRF models, to prevent the models from degrading into kernel logistic regression), and when both vertex and edge cliques are allowed. (The kernel between vertex cliques is $\overline{K}(\boldsymbol{z}, \boldsymbol{z}') = \overline{K}(x, x')\,\delta(y_1, y_1')$, and between edge cliques it is $\overline{K}(\boldsymbol{z}, \boldsymbol{z}') = \overline{K}(x, x')\,\delta(y_1, y_1')\,\delta(y_2, y_2')$.) The total number of clique candidates is about 4800 (vertex only) and 20000 (vertex and edge). The rapid reduction in risk indicates sparse training of kernel CRFs is successful. Also when more flexibility is allowed by including edge cliques, the risk reduction is much faster. The more flexible model also has higher test set log likelihood (center) although this does not improve the test set accuracy too much (right). These observations are generally true for other trials too.

Per-residue accuracy. To evaluate prediction performance, we use the overall per-residue accuracy (also known as Q_3). We experiment with training set size of 5 and 10 sequences respectively. For each size we perform 10 trials where the training sequences are randomly sampled, and the remaining proteins are used as the test set. For kernel CRF we select 300 cliques, again from either vertex candidates alone or vertex and edge candidates. We compare them with the SVM-light package (Joachims, 1998) for SVM classifier. All methods use the same RBF kernel. See Table 1. KCRFs and SVMs have comparable performance.

Transition accuracy. Further information can be obtained by studying transition boundaries, for example, the transition from "coil" to "sheet." From the point of view of structural biology, these transition boundaries may provide important information about how proteins fold in three dimension. On the other hand, those are the positions where most secondary structure prediction systems will fail. The transition boundary is defined as a pair of adjacent positions $(i, i+1)$ whose true labels differ. It is classified correctly only if both labels are correct. This is a very hard problem, as can be seen in Table 2 and KCRFs are able to achieve a considerable improvement over SVM.

Semi-supervised learning. We start with an unweighted 10 nearest neighbor graph over positions in both training and

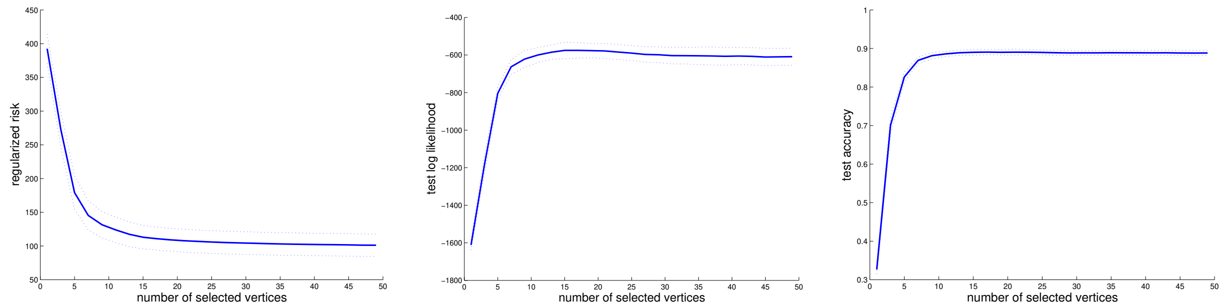

Figure 4. Clique selection on the Gaussian mixture data. Left: regularized risk; Center: test set log likelihood; Right: test set accuracy.

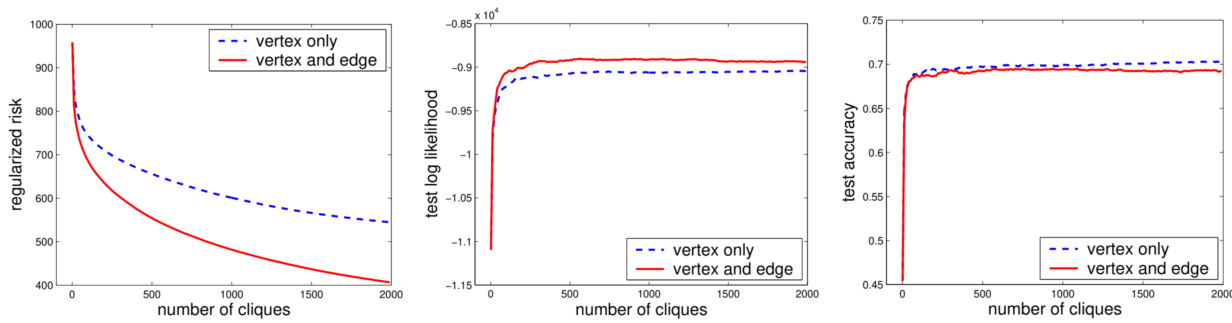

Figure 5. Clique selection for KCRFs on the protein data. Left: regularized risk; Center: test set log likelihood; Right: test set accuracy. The two curves represent the cases where only vertex cliques are selected (dashed) vs. both vertex and edge cliques are selected (solid).

test sequences, with the metric being Euclidean distance in the feature space. Then the eigensystem of the normalized Laplacian is computed. The semi-supervised graph kernel is obtained with the function $r(\lambda_i) = \frac{1}{\lambda_i + 0.01}$ on the first $i \leq 200$ eigenvalues. The rest eigenvalues are set to zero. We use the graph kernel together with the RBF kernel in KCRF. As a clique candidate is associated with a kernel, we now select two best candidates per iteration, one with the graph kernel and the other with the RBF kernel. We still run for 300 iterations for all trials. We also report the results using Transductive SVMs (TSVMs) (Joachims, 1999) with the RBF kernel. From the results in Table 3, we can see that the semi-supervised graph kernel is significantly better than TSVMs on the 5-protein dataset while achieves no improvement on the other one. To diagnose the cause, we look at the graph together with all the test labels. We find that the labels are not smooth w.r.t. the graph: on average only 54.5% of a node's neighbors have the same label as that node. Detecting faulty graphs without using large amount of labels, and constructing better graphs remain future research.

The approximate average running time of each trial, including both training and testing, is 30 minutes for KCRFs, 7 minutes for SVMs, and 16 hours for TSVMs. For KCRFs the majority of the time is spent on clique selection.

6. Conclusion

Kernel conditional random fields have been introduced as a framework for approaching graph-structured classification problems. A representer theorem was derived which shows how KCRFs can be motivated by regularization theory. The resulting techniques combine the strengths of hidden Markov models, or more general Bayesian networks, kernel machines, and standard discriminative linear classifiers including logistic regression and SVMs. The formalism presented is quite general, and should apply naturally to a wide range of problems.

Our experimental results on synthetic data, while carefully controlled to be simple, clearly indicate how sequence modeling, graph kernels for semi-supervised learning, and clique selection for sparse representations work together within this framework. The success of these methods in real problems will depend on the choice of suitable kernels that capture the structure of the data.

For protein secondary structure prediction, our results are only suggestive. Secondary structure prediction is a problem that has been extensively studied for more than 20 years; yet the task remains difficult, with prediction accuracies remaining low. The major bottleneck lies in beta-sheet prediction, where there are long range interactions between regions of the protein chain that are not necessarily consecutive in the primary sequence. Our experimental results indicate that KCRFs and semi-supervised kernels have the

	5 protein set		10 protein set	
Method	Accuracy	std	Accuracy	std
KCRF (v)	0.6625	0.0224	0.6933	0.0276
KCRF (v+e)	0.6562	0.0202	0.6933	0.0272
SVM	0.6509	0.0307	0.6875	0.0235

Table 1. Per-residue accuracy of different methods for secondary structure prediction, with the RBF kernel. KCRF (v) uses vertex cliques only; KCRF (v+e) uses vertex and edge cliques.

	5 protein set		10 protein set	
Method	Accuracy	std	Accuracy	std
KCRF (v)	0.1097	0.0271	0.1462	0.0235
KCRF (v+e)	0.1114	0.0250	0.1522	0.0214
SVM	0.0667	0.0313	0.1066	0.0311

Table 2. Transition accuracy with different methods.

	5 protein set		10 protein set	
Method	Accuracy	std	Accuracy	std
KCRF (v)	0.6722	0.0194	0.6854	0.0190
KCRF (v+e)	0.6674	0.0201	0.6819	0.0194
Trans. SVM	0.6480	0.0276	0.6813	0.0210

Table 3. Per-residue accuracy with semi-supervised methods.

potential to lead to progress on this problem, where the state of the art has been based on heuristic "sliding window" methods. However, our results also suggest that the improvement due to semi-supervised learning is hindered by the lack of a good similarity measure with which to construct the graph. The construction of an effective graph is a challenge that may best be tackled by biologists and machine learning researchers working together.

Acknowledgments

This work was supported in part by NSF ITR grants CCR-0122581, IIS-0205456 and IIS-0312814.

References

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden Markov support vector machines. *ICML'03*.

Belkin, M., & Niyogi, P. (2002). *Semi-supervised learning on manifolds* (Technical Report TR-2002-12). University of Chicago.

Chapelle, O., Weston, J., , & Schoelkopf, B. (2002). Cluster kernels for semi-supervised learning. *NIPS'02*.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. *Proceedings of EMNLP'02*.

Cuff, J., & Barton, G. (1999). Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins*, *34*, 508–519.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE PAMI*, *19*, 380–393.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *ECML*.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *ICML'99*.

Jones, D. (1999). Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol.*, *292*, 195–202.

Kabsch, W., & Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, *22*, 2577–2637.

Kim, H., & Park, H. (2003). Protein secondary structure prediction based on an improved support vector machines approach. *Protein Eng.*, *16*, 553–60.

Kimeldorf, G., & Wahba, G. (1971). Some results on Tchebychean spline functions. *J. Math. Anal. Applic.*, *33*, 82–95.

Kumar, S., & Hebert, M. (2003). Discriminative fields for modeling spatial dependencies in natural images. *NIPS'03*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML'01*.

Lanckriet, G., Cristianini, N., an Laurent El Ghaoui, P. B., & Jordan, M. (2004). Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, *5*, 27–72.

McCallum, A. (2003). Efficiently inducing features of conditional random fields. *UAI'03*.

Pinto, D., McCallum, A., Wei, X., & Croft, W. B. (2003). Table extraction using conditional random fields. *SIGIR'03*.

Sha, F., & Pereira, F. (2003). Shallow parsing with conditional random fields. *Proceedings of HLT-NAACL*.

Smola, A., & Kondor, R. (2003). Kernels and regularization on graphs. *COLT'03*.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin Markov networks. *NIPS'03*.

Zhu, J., & Hastie, T. (2001). Kernel logistic regression and the import vector machine. *NIPS'01*.

Zhu, X., Gharahmani, Z., & Lafferty, J. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. *ICML'03*.