Constrained Optimization for Validation-Guided Conditional Random Field Learning

Minmin Chen Yixin Chen Michael R. Brent Aaron E. Tenney
Department of Computer Science and Engineering
Washington University in St. Louis
Saint Louis, Missouri, USA
{mc15, chen, brent, aet1}@cse.wustl.edu

ABSTRACT

Conditional random fields(CRFs) are a class of undirected graphical models which have been widely used for classifying and labeling sequence data. The training of CRFs is typically formulated as an unconstrained optimization problem that maximizes the conditional likelihood. However, maximum likelihood training is prone to overfitting. To address this issue, we propose a novel constrained nonlinear optimization formulation in which the prediction accuracy of cross-validation sets are included as constraints. Instead of requiring multiple passes of training, the constrained formulation allows the cross-validation be handled in one pass of constrained optimization.

The new formulation is discontinuous, and classical Lagrangian based constraint handling methods are not applicable. A new constrained optimization algorithm based on the recently proposed extended saddle point theory is developed to learn the constrained CRF model. Experimental results on gene and stock-price prediction tasks show that the constrained formulation is able to significantly improve the generalization ability of CRF training.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning* ; J.3 [**Life and Medical Sciences**]: [Biology and genetics]; J.1 [**Administrative Data Processing**]: [Financial]

General Terms

Algorithm, Experimentation

Keywords

Conditional random fields, Constrained optimization, Cross validation, Extended saddle points

1. INTRODUCTION

We study in the paper new formulations and optimization algorithms for training Conditional Random Fields (CRFs), a class of discriminative probabilistic models for sequence labeling. CRFs have attracted intensive interest and achieved success in various domains, such as as computer vision [12, 17], natural language processing [10, 16, 15, 13], and bioinformatics [6]. In general, CRFs model the conditional distribution $p(\mathbf{y}|\mathbf{x})$ between the labeling sequence and the observation sequence. The learning of CRF parameters is traditionally formulated as an unconstrained optimization problem of maximizing the conditional distribution. We propose a constrained formulation for training CRFs in this paper.

A key advantage of CRFs is that they support the use of complex features. In most cases, it is insufficient to include only simple and independent features. Rather, it is necessary to use long-range features that are overlapping and inter-dependent. It is hard for traditional generative graphical models, such as hidden Markov models (HMMs), to capture these complex features. The Markovian assumption requires that the current state only depends on the previous one and that the observation depends only on the current state. These assumptions severely limit the possible features that can be included in HMMs. CRFs overcome the above limitations of HMMs [10] by relaxing the independence assumptions, allowing overlapping and dependent features to be included and learned in a unified fashion.

The flexibility of CRF models encourages the use of large feature sets with a rich collection of features. During our study, we found that CRF models with a large feature set often suffer from overfitting. A CRF model with the optimal parameter set found on the training data often has a significant performance degradation when applied to unseen data. For instance, in a gene prediction task, using the CRF model trained on the training data, we achieved 100% gene level sensitivity and specificity when testing on the same set, comparing to an average of only $2-4\%$ accuracy when testing on unseen data.

One existing approach to address the overfitting problem of CRFs is to include a regularization term in the objective function [16, 19] to penalize large weight vectors. A major problem with this approach is that, the regularization factor which controls the penalty strength is hard to control. A randomly selected regularization factor will not be effective in addressing the overfitting problem.

In this paper, we propose a new constrained formulations for CRF training to help overcome the overfitting problem, inspired by the idea of cross-validation. Cross-validation [9] is often used to estimate the accuracy of a classifier and to select models. We propose to use cross-validation in a novel way as a measure to address overfitting. Constraints

KDD'09, June 28–July 1, 2009, Paris, France.

prescribing the difference of the score of labels generated by the trained CRF model and that of the real labels on the validation sets are added to prevent the model from fitting freely and tightly to the training data.

The constrained formulation has several advantages. First, our experience shows that the cross-validation constraints can effectively help address the overfitting problem, as this approach can enforce an even distribution of validation errors across the training sets. Second, instead of separating the training and validation phases, we integrate validation into the training process by modeling the validation quality as constraints in the problem formulation. Third, supported by a constrained optimization solver proposed in this paper, we only need one run of constrained optimization, although we have multiple validation sets. Fourth, we create our validation sets from the training data, avoiding waste of training data, which is crucial for problems with limiting data available. Further, the constraints are beneficial in difficult training scenarios. We have found that, for CRF training, gradient-based optimization methods may terminate prematurely at non-optimal points due to a flat terrain near the global optimum [4]. The violated constraints provide additional guidance during search, leading the search to good directions that effectively reduce the objective function.

Since the new constraint formulation is discontinuous and non-differentiable, classic constrained optimization methods, like Lagrangian method, are not applicable. A new constrained optimization algorithm based on the recently proposed extended saddle point (ESP) theory [20, 21] is developed to solve the constrained problem. Unlike the traditional Lagrange multiplier method, in which a set of unique Lagrange multipliers is required, the ESP condition is satisfied for an extended range of penalty multipliers, making the search much easier and more robust. Further, the ESP theory provides optimality conditions for problems with non-differentiable constraints, which is needed in our CRF formulation. Based on the ESP theory, we develop a constrained solver to efficiently train constraint-based CRFs.

The paper is organized as follows. In Section 2, we review the basics of CRFs and the traditional parameter estimation methods. In Section 3, we detail our new constrained formulation. In Section 4, we describe our constrained optimization algorithm. Experimental results are presented in Section 5 and conclusions given in Section 6.

2. BACKGROUND

As shown in Figure 1(a), a CRF is a graphical model based on an undirected graph $G = (V, E)$, such that $Y = (Y_v)_{v \in V}$, the set of hidden variables, when conditioned on X, the set of variables on the observation sequence, obey the Markovian property with respect to the graph: $p(Y_v|X, Y_u, u \neq v) = p(Y_v|X, Y_u, u \sim v)$, where $u \sim v$ indicates that u and v are neighbors in G [10]. The fully connected subgraphs of G define a set of cliques $\mathcal{C} = \{X_c, Y_c\}$. Each clique defines a *feature* for the CRFs.

Each predefined feature typically returns a binary value. For example, a feature can be defined as:

$$f_k(\mathbf{y}_t, \mathbf{x}_t) = 1_{\{y_{t-1}=NC\}} 1_{\{y_t=C\}} 1_{\{\mathbf{x}_{t\cdots t+2}='ATG'\}}$$

It evaluates to 1 when the previous state is "NC", the current state is "C", and the observations at time t to $t+2$ are 'ATG'.

In this paper, we focus on linear-chain CRF, a class of

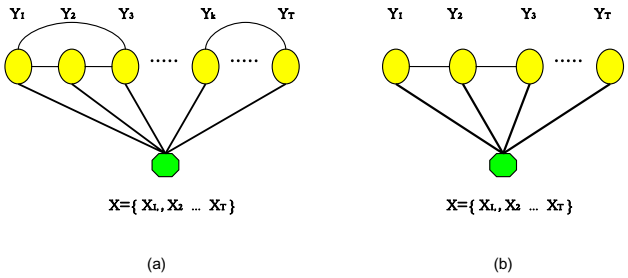

Figure 1: (a) A general graphical CRF model. (b) A linear-chain CRF model.

CRFs that is computationally tractable and widely used. In linear chain CRFs, each each clique (feature) only involves two consecutive hidden states as shown in Figure 1(b).

Let $\mathbf{x}$ and $\mathbf{y}$ be the observation and labeling sequences, respectively. By the fundamental theorem of random fields [7], a linear-chain CRF defines the conditional distribution of a label sequence $\mathbf{y}$, given the observation sequence $\mathbf{x}$ as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{k=1}^{K}\sum_{t=1}^{T} \omega_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\}, \quad (1)$$

where $\{f_k(y_t, y_{t-1}, \mathbf{x}_t)\}_{k=1}^{K}$ is a set of feature functions, and $W = \{\omega_k\} \in \mathcal{R}^K$ is the corresponding weight vector. $Z(\mathbf{x})$ is an instance-specific normalization function

$$Z(\mathbf{x}) = \sum_{\mathbf{y'}} \exp\left\{\sum_{k=1}^{K}\sum_{t=1}^{T} \omega_k f_k(y'_t, y'_{t-1}, \mathbf{x}_t)\right\}. \quad (2)$$

2.1 CRF training and overfitting

Given training data $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\}$, to estimate the weights $\{\omega_k\}$, we typically solve the following unconstrained optimization problem to maximize the conditional log likelihood:

$$\text{maximize}_W \quad \mathcal{O}(W) = \log p(\mathbf{y}|\mathbf{x}). \quad (3)$$

It is a convex optimization problem with only one global optimum. The objective function cannot be maximized in a closed form and numerical optimization is used for training.

The partial derivatives of the objective function with respect to each feature weight $\omega_k, k = 1, 2, \cdots, K$ takes this form,

$$\begin{aligned}\frac{\partial \mathcal{O}}{\partial \omega_k} &= \sum_{t=1}^{T} f_k(y_t, y_{t-1}, \mathbf{x}_t) \\ &- \sum_{t=1}^{T}\sum_{y,y'} f_k(y, y', \mathbf{x}_t) p(y, y'|\mathbf{x}^{(i)}) \end{aligned} \quad (4)$$

In the derivative, the first term is the empirical count of feature f_k in the training data, or in other words, the expected number of appearances of f_k under the empirical distribution. The second term is the expected number of appearances of f_k under the model distribution $p(\mathbf{y}|\mathbf{x}; W)\hat{p}(\mathbf{x})$ [10]. When the model is optimized, the gradient reaches zero and the two expectations will be equal. Hence, CRF training can be viewed as a maximum likelihood training for exponential models which tries to find an optimal parameter set to best fit the training data. Therefore, like other maximum

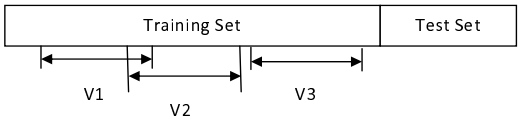

Figure 2: Definitions of the training, test, and validation sets in the STMV framework.

likelihood methods, CRF training is prone to overfitting the training data.

2.2 Related work

A common way to avoid overfitting is via regularization, which penalizes weight vectors whose norm is too large. Instead of maximizing solely the conditional distribution, an Ω_2 regularization term is added:

$$\max_W \quad \mathcal{O}(W) = \log p(\mathbf{y}|\mathbf{x}) - \sum_{k=1}^{K} \frac{\omega_k^2}{2\delta^2}, \tag{5}$$

where δ^2 is a parameter determining the strength of penalty. The regularized formulation can be viewed as maximizing a posteriori estimation of W, which is assigned a Gaussian prior with zero mean and covariance $\delta^2 I$.

To determine the best regularization parameter requires a computationally-intensive parameter sweep. In previous work, part of the training data is held out to find the regularization parameter δ^2 first. However, it has been reported that the model accuracy does not appear to be sensitive to the changes in δ^2, even when δ^2 is varied up to a factor of 10 [18]. Our own experimental results have confirmed it.

A number of other smoothing methods have been used to address the overfitting problems for maximum entropy models [5]. The smoothing method tries to relax the constraint that the model distribution should be exactly the same as the empirical distribution on the training data. The good-Turing smoothing algorithm [14] tries to add a discount to the empirical count of events in the training data. The fuzzy maximum entropy framework encourages the model to fit the training data, while at the meantime adds a prior distribution favoring uniform models. The fat constraints method [11] tries to avoid over-fitting by not exactly satisfying the constraints but instead requiring the difference between the empirical and model distributions to be within certain range. It was reported that the Gaussian prior in (5) performs as well as or better than all the other algorithms [5].

3. CONSTRAINED OPTIMIZATION FORMULATION

We propose a constrained formulation for CRF training based on cross-validation. Our approach integrates cross-validation into the training process in a novel way, aiming at avoiding overfitting and finding better models.

3.1 The single training multiple validation (STMV) framework

A common setting of cross validation is the k-fold cross-validation, in which data $\mathcal{D}$ is randomly split into k mutually exclusive subsets $\mathcal{D}_1, \cdots, \mathcal{D}_k$. The predictor is trained and tested k times, each time the whole data set $\mathcal{D}$ excluding $\mathcal{D}_t, t \in 1, 2, \cdots, k$ is used as the training set and $\mathcal{D}_t$ as the test set. Such a method is often used for model selection. However, the method wastes $1/k$ of the training data and requires k-fold training time.

CRF training for large scale problems is computationally expensive. In a gene prediction task with around 300K bases and 30K features, it takes 5 to 6 hours on our cluster with 20 processors to complete one run of the training process. Since CRF optimization is usually time-consuming, it is very expensive, if not prohibitive, to use the traditional k-fold cross-validation.

We use cross-validation in a single training multiple validation (STMV) framework originally proposed for neural network training [22]. Instead of separating the training and validation phases, we integrate validation into the training process by modeling the validation quality as constraints in the problem formulation. We select multiple subsets from the training sequence as validation sets as shown in Figure 2. The entire training set is used for training. This way, we do not waste any training data as the k-fold cross-validation does. Also, this approach can enforce an even distribution of validation errors across the training set and offer flexibility in choosing the validation sets.

3.2 Constrained Formulation

As illustrated in Figure 2, we use multiple subsets from the training set as the validation sets. Define

$$S_W(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{K} \sum_{t \in I} \omega_k f_k(y_t, y_{t-1}, \mathbf{x}_t)$$

to be the **score** of a specific labeling sequence $\mathbf{y}$ for an observation sequence $\mathbf{x}$, where I is the set of indices in sequences $\mathbf{x}$ and $\mathbf{y}$. We propose the following **constrained CRF (CCRF)** formulation:

$$\begin{aligned} \max_W \ \mathcal{O}(W) &= \log p(\mathbf{y}|\mathbf{x}) && (6) \\ \text{subject to } h_j(W) &\le \gamma, \\ \text{where } h_j(W) &= S_W(\mathbf{v}^{(j)}, \hat{\mathbf{y}}^{(j)}) - S_W(\mathbf{v}^{(j)}, \mathbf{y}^{(j)}). \\ & \quad \forall j = 1, \cdots, V, && (7) \end{aligned}$$

Here, γ is a small positive constant, $\mathcal{V} = \{(\mathbf{v}^{(j)}, \mathbf{y}^{(j)})\}_{j=1,\cdots,V}$ are the validation sets extracted from the training data, V is the total number of validation sets, and $\hat{\mathbf{y}}^{(j)}$ is the most likely labeling sequence for the observation sequence $\mathbf{v}^{(j)}$ found by the Viterbi algorithm under the current model. From the mechanism of Viterbi algorithm, we know that $\hat{\mathbf{y}}^{(j)}$ is the sequence with the best score.

Instead of using the prediction accuracy of the validation sets, we use the score difference between the best scoring labeling sequence $\hat{\mathbf{y}}_i$ and the desired labeling sequence $\mathbf{y}_i$ in our constraints. We observe that a CRF model learned through unconstrained optimization often has substantial differences of scores on the validation sets. Generally, the performance on the validation set is strongly correlated to the performance of the trained model on unseen data. A trained CRF model giving many wrong predictions on the validation sets tend to perform poorly on the testing data. To ensure non-deteriorated performance on unseen data, we want to make sure that the trained model not only fits the training data, but also the validation sets. In other words, we want to make sure that, for the trained model, the score

difference between the most likely sequence $\hat{\mathbf{y}}_i$ and the annotated sequence $\mathbf{y}_i$ is small for each validation set.

In our formulation, the parameter γ controls the score difference allowed for each validation set. We set γ to be a small positive constant. In difficult training scenarios, it may be impossible to reach convergence with $\gamma = 0$. Setting $\gamma > 0$ may allow training to converge faster. Further, suitable relaxation may help avoid overfitting.

4. EXTENDED SADDLE POINT SEARCH ALGORITHM

The CCRF formulation in (6) has discontinuous and non-differentiable constraints. The constraint for validation set $j, j = 1..V$, can be re-written as:

$$\begin{aligned} g_j(W) &= \sum_{k=1}^{K} \sum_{t \in I_j} \omega_k f_k(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)}) \\ &- \sum_{k=1}^{K} \sum_{t \in I_j} \omega_k f_k(y_t^{(j)}, y_{t-1}^{(j)}, \mathbf{v}_t^{(j)}) - \gamma \leq 0, \end{aligned}$$

where I_j is the set of indices of the j^{th} validation set. Note that, for each predefined *feature* $f_k(y', y, \mathbf{v}_t)$, it is satisfied when the observation at time t matches $\mathbf{v}_t$. The state changes of W may lead to changes of the labeling sequence $\hat{\mathbf{y}}^{(j)}$ generated by the Viterbi algorithm for each validation set $\mathcal{V}^{(j)}$. When the labeling sequence changes, the feature functions $f_k(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)})$ may change from 1 to 0 (true to false) or from 0 to 1 (false to true), resulting in a discontinuous $g_j(W)$.

Since the CCRF problem in (6) has non-differentiable constraints, it cannot be readily solved using existing Lagrangian methods that require the differentiability of functions. Sampling methods such as simulated annealing and genetic algorithms are too slow. Penalty methods can handle such constraints but it is in general difficult to choose proper penalties. In this paper, we propose to solve (6) using the recently developed extended saddle point (ESP) theory [20, 21]. The ESP search has successfully solved discontinuous optimization problems from AI planning [21] and engineering [20].

4.1 Extended saddle point (ESP) theory

The ESP theory can handle nonlinear programming (NLP) problems in a continuous, discrete or mixed space. Here we focus on NLPs with continuous variables. A NLP is defined as follows, with functions f, $h = (h_1, \ldots, h_m)^T$, and $g = (g_1, \ldots, g_r)^T$ defined in real space $\mathcal{R}$:

$$(P_c): \min_x f(x) \text{ where } x = (x_1, \ldots, x_v)^T \in \mathcal{R}^v \quad (8)$$
$$\text{subject to } h(x) = 0 \text{ and } g(x) \leq 0.$$

In order to solve CCRF, we assume that $f(x)$ is continuous and differentiable, but $h(x)$ and $g(x)$ may be non-differentiable.

DEFINITION 1. *A point x^* is a CGM, a constrained global minimum of (P_c), if x^* is feasible and $f(x^*) \leq f(x)$ for all feasible x.*

While *CGM* is generally difficult to find for nonlinear problems, a common goal of solving P_c is to find a constrained local minimum x with respect to $\mathcal{N}_{CN}(x) = \{x' : \|x' - x\| \leq \epsilon \text{ and } \epsilon \to 0\}$, the *continuous neighborhood* of x.

DEFINITION 2. *A point x^* is a CLM, a constrained local minimum with respect to the continuous neighborhood of x^* in P_c, if x^* is feasible and $f(x^*) \leq f(x)$ for all feasible $x \in \mathcal{N}_{CN}(x^*)$.*

The ESP theory is based on the ℓ_1^m-penalty function defined below.

DEFINITION 3. *The ℓ_1^m-penalty function of P_c in (8) is*

$$L_c(x, \alpha, \beta) = f(x) + \alpha^T |h(x)| + \beta^T g^+(x),$$

where $g^+(x) = \max(0, g(x))$, $\alpha \in \mathcal{R}^m$ and $\beta \in \mathcal{R}^r$ are the extended penalty values.

Instead of using a single penalty term as the traditional ℓ_1-penalty, the ℓ_1^m-penalty uses multiple penalty parameters α and β, one for each constraint. The ℓ_1^m-penalty can be viewed as a mixture of the Lagrangian function with multiple multipliers and the ℓ_1-penalty with non-negative transformations on constraints.

DEFINITION 4. Constraint-qualification condition. *A point $x \in \mathcal{R}^v$ meets the constraint qualification if there exists no direction $\vec{p} \in \mathcal{R}^v$ along which the subdifferentials of equality and active inequality constraints are all zero. Let*

$$D_x(\phi(x), \vec{p}) = \lim_{\epsilon \to 0} \frac{\phi(x + \epsilon \vec{p}) - \phi(x)}{\epsilon},$$

the condition is:

$$\nexists\, \vec{p} \in \mathcal{R}^v \text{ such that } D_x(h_i(x), \vec{p}) = 0$$
$$\text{and } D_x(g_j(x), \vec{p}) = 0 \quad \forall i \in C_h \text{ and } j \in C_g,$$

where C_h and C_g are, respectively, the sets of indices of equality and active inequality constraints. This qualification is always less restrictive *than the KKT regularity condition.*

The following theorem gives a necessary and sufficient condition for CLMs of NLPs.

THEOREM 1. Necessary and sufficient extended saddle point condition on *CLM* of P_c. *Suppose $x^* \in \mathcal{R}^v$ satisfies the constraint qualification, then x^* is a CLM of P_c if and only if there exist finite $\alpha^* \geq 0$ and $\beta^* \geq 0$ such that, for any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$, the following condition is satisfied:*

$$L_c(x^*, \alpha, \beta) \leq L_c(x^*, \alpha^{**}, \beta^{**}) \leq L_c(x, \alpha^{**}, \beta^{**}) \quad (9)$$

for all $x \in \mathcal{N}_{CN}(x^*)$, $\alpha \in \mathcal{R}^m$, and $\beta \in \mathcal{R}^r$.

Theorem 1 differs from the traditional saddle point (SP) condition [1] in a significant way. The traditional SP condition is based on a Lagrangian function and works only for NLPs with continuous and differentiable constraints. Further, the SP condition is true at unique Lagrange multipliers. In contrast, ESP based on the ℓ_1^m function works for discontinuous, non-differentiable NLPs and it suffices to find any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$. Computationally, it is much easier to find α^{**} and β^{**} in an extended region than finding unique Lagrange multipliers as required by the traditional Lagrangian theory.

Figure 3 shows a general framework that implements the conditions in Theorem 1. The inner loop looks for local minima of $L_c(x, \alpha, \beta)$ in the continuous neighborhoods, whereas

$\alpha \longrightarrow 0$; $\beta \longrightarrow 0$;
repeat
  increase α_i by δ_i if $h_i(x, y) \neq 0$ for all i;
  increase β_j by δ_j if $g_j(x, y) \not\leq 0$ for all j;
  repeat
    perform descent of $L_c(x, \alpha, \beta)$ with respect to x;
  until a local minimum of $L_c(x, \alpha, \beta)$ has been found;
until a *CLM* of P_c has been found or $(\alpha > \bar{\alpha}^*$ and $\beta > \bar{\beta}^*)$;

Figure 3: Iterative implementation of ESP search for locating *CLM* of P_c. $\bar{\alpha}^*$ and $\bar{\beta}^*$ are pre-defined upper bounds.

the outer loop performs ascents on α and β for unsatisfied constraints and stops when a *CLM* has been found.

Significance of the ESP condition. The ESP condition has several salient advantages over previous constraint handling theory. Unlike the Karush-Kuhn-Tucker (KKT) condition in the Lagrangian theory that works only for continuous and differentiable problems, ESP condition offers a uniform treatment to discrete and mixed problems and does *not* require the constraints to be differentiable or in closed form. More importantly, unlike the KKT condition that requires finding a set of ***unique*** Lagrange multipliers which are oftentimes hard to locate exactly for large problems, ESP condition is satisfied over ***an extended region*** of penalty values. In fact, ESP condition is true for any $\alpha^{**} > \alpha^*$ and $\beta^{**} > \beta^*$, where α^* and β^* are ***finite*** thresholds. Experimental results on discontinuous planning and engineering problems [20, 21] suggest that it is generally much easier to locate suitable penalty values in an extended region than to find the exact Lagrange multipliers.

The ESP condition result is also stronger than the KKT condition in the sense that KKT condition is only necessary but not sufficient. That is, any qualified *CLM* satisfies the KKT condition, but a point satisfying the KKT condition may not be a *CLM*. Thus, an algorithm converging to a KKT point may not find a *CLM*.

4.2 The ESP search algorithm

We develop a solver that solves the constrained problem (6) based on the ESP search framework in Figure 3. We have recently developed CRF-OPT [4], a package for general unconstrained CRF training, on top of the Toolkit for Advanced Optimization (TAO) [2]. CRF-OPT uses the Limited Memory Variable Metric (LMVM) method, a quasi-Newton method, as the unconstrained optimization algorithm.

We develop our constrained CRF solver on top of CRF-OPT. The ℓ_1^m-penalty function of (6) is:

$$L_c(W, \beta) = -\mathcal{O}(W) + \sum_{j=1}^{V} \beta_j g_j^+(W) \tag{10}$$

Our ESP algorithm consists of two loops. The **outer loop** updates β and the **inner loop** minimizes $L_c(W, \beta)$ in the W-space using the LMVM method in TAO. The quasi-Newton's method, LMVM, requires the objective and gradient information of $L_c(W, \beta)$ and approximates the inverse Hessian to generate descent directions. Since $g_j(W)$ is not differentiable, we approximate its gradient as follows.

The partial derivative of $L_c(W, \beta)$, with respect to weight

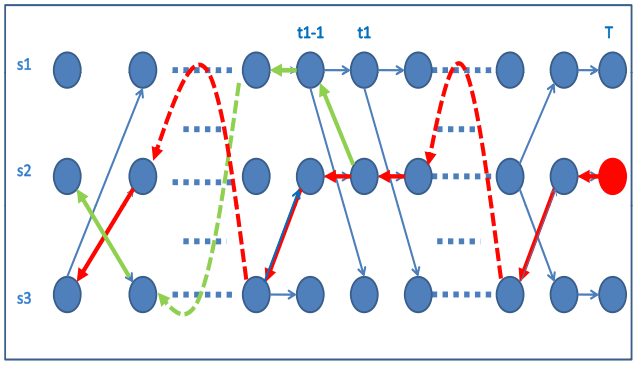

Figure 4: Change of weights influences the most likely sequence selected by the Viterbi algorithm. See text for explanation.

$\omega_k, k = 1..K$, can be expressed as:

$$\frac{\partial L_c(W, \beta)}{\partial \omega_k} = -\frac{\partial \mathcal{O}(W)}{\partial \omega_k} + \sum_{j=1}^{V} \beta_j \frac{\partial g_j^+(W)}{\partial \omega_k} \tag{11}$$

The first term, $\frac{\partial \mathcal{O}(W)}{\partial \omega_k}$, is the gradient of the original objective, and can be calculated efficiently by the forward-backward algorithm [18]. We now focus on computing $\frac{\partial g_j^+(W)}{\partial \omega_k}$.

$$\frac{\partial g_j^+(W)}{\partial \omega_k} = \frac{\partial}{\partial \omega_k}\Big(S_W(\mathbf{v}^{(j)}, \hat{\mathbf{y}}^{(j)}) - S_W(\mathbf{v}^{(j)}, \mathbf{y}^{(j)})\Big)$$
$$= \sum_{t \in I_j} \sum_{k'=1}^{K} \omega_{k'} \cdot \frac{\partial}{\partial \omega_k} f_{k'}(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)}) +$$
$$\sum_{t \in I_j} \Big(f_k(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)}) - f_k(y_t^{(j)}, y_{t-1}^{(j)}, \mathbf{v}_t^{(j)})\Big) \tag{12}$$

The first term of (12)

$$\sum_{k'=1}^{K} \sum_{t \in I_j} \omega_{k'} \frac{\partial}{\partial \omega_k} f_{k'}(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)}) \tag{13}$$

is needed because the most likely sequence $\hat{\mathbf{y}}^{(j)}$ of the validation set $\mathcal{V}^{(j)}$ depends on ω_k.

Figure 4 illustrates the mechanism of the Viterbi algorithm and how the change of ω_k will affect the counts of features. Define feature f_k as,

$$f_k(y_{t-1}, y_t, \mathbf{x}_t) = \mathbf{1}_{\{y'=s_1\}} \mathbf{1}_{\{y=s_2\}} \mathbf{1}_{\{\mathbf{x}_t=\mathbf{v}\}}.$$

Let $\mathcal{S}$ be the set of possible states ($\mathcal{S} = \{s_1, s_2, s_3\}$ in Figure 4) and let the length of the sequence be T. Before backtracking, Viterbi will do a forward run. For each state $y \in \mathcal{S}$ at time t, it finds out the state at time $t-1$ that will most likely lead to this state, as shown in blue arrows in the figure. Also, the highest probability of reaching state y at time t is remembered in $\tau(t, y)$. Then, Viterbi backtracks from time T by starting from the state $y^\star$, $\tau(T, y^\star) \geq \tau(T, y), \forall y \in \mathcal{S}$, and follows the red arrows in the figure. Let $t1$ be the position where the backward process differs. As we change the weight of ω_k, Viterbi finds that state s_1 at time $t_1 - 1$ has a better chance to lead to s_2 , and backtracks in a different path and changes the labels $\hat{y_t}^{(j)}, t \in [1, t_1 - 1]$, as shown

in green arrows. As a result, the satisfiability of all the features, not only feature f_k, will be changed. The direct effect of changing weight ω_k is the change of the satisfiability of feature f_k. That is, the origin of the change is when at one position t where $f_k(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)})$ changes from 0 to 1 or from 1 to 0.

Note that, in the Viterbi algorithm, most of the computational time is spent on the forward run, and the backtracking process requires little time. Hence, we want to avoid repeatedly carriyng out the forward runs when approximating the discontinuous term (13).

We detail our approximation scheme when weight ω_k is increased as follows. The case when ω_k is decreased is symmetric.

For a feature $f_k(y', y, \mathbf{v})$, we find out the positions t where $\mathbf{v}_t^{(j)} = \mathbf{v}$, $\hat{y}_t^{(j)} = y$ and $\hat{y}_t^{(j)} \neq y'$, and store them in $\mathcal{P}_k$, the set of such positions that, when ω_k increases, $f_k(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)})$ will change from 0 to 1.

The reason for us to require $\mathbf{v}_t^{(j)} = \mathbf{v}$ is that, at each position t, the feature $f_k(y', y, \mathbf{v})$ is satisfied if $\hat{y}_{t-1}^{(j)} = y'$, $\hat{y}_t^{(j)} = y$ and $\mathbf{v}_t^{(j)} = \mathbf{v}$. Since the observation sequence $\mathbf{v}^{(j)}$ is given, the change of the weight ω_k can only affect the changes of the labeling sequence $\hat{\mathbf{y}}^{(j)}$ found by the Viterbi algorithm. Therefore, if at time t, $\mathbf{v}_t^{(j)} \neq \mathbf{v}$, then a change of ω_k will not lead to a change in the satisfiability of $f_k(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)})$.

We require $\hat{y}_t^{(j)} = y$ because Viterbi algorithm works in a backtracking way, and the precondition of satisfying the transition (y', y) is to let $\hat{y}_t^{(j)} = y$. As we assume the weights for other features are fixed when we do approximation, if $\hat{y}_t^{(j)} \neq y$, then there is no way change of ω_k can change $f_k(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)})$ from 0 to 1.

The reason for us to require $\hat{y}_t^{(j)} \neq y'$ is obvious. If $\hat{y}_t^{(j)} = y'$, y' has already been chosen as the preceding state for y, then the feature f_k has already been satisfied. Increasing the weight ω_k will have no effect on the configuration of the most likely sequence.

For each $t \in \mathcal{P}_k$, we compute the minimum increase $\Delta_{\omega_k}^t$ of ω_k required to make y' the preceding state of y. Let $\hat{y}$ be the preceding state of y originally selected by the Viterbi algorithm. $\Delta_{\omega_k}^t$ can be efficiently calculated from $\tau(y, t-1)$, $\tau(\hat{y}, t-1)$, and the weights of other active features at time t.

Then, we find the position $t^\star$ satisfying, $\Delta_{\omega_k}^{t^\star} \leq \Delta_{\omega_k}^t, \forall t \in \mathcal{P}_k$ and increase ω_k by $\Delta_{\omega_k}^{t^\star}$. By doing so, only the preceding state of y at time $t^\star$ is changed to y'. For all the other positions $t \in \mathcal{P}_k$, since the increase of ω_k does not reach their minimum requirement, the configuration will remain the same. Then we can backtrack from y' at time $t-1$ and efficiently find the most likely sequence, without performing the forward part of the Viterbi algorithm. After having the new most likely sequence, we can acquire the change of counts for all the features with respect to the change of ω_k. The changes are store in a vector $\Delta A = \{\Delta A_1, \Delta A_2, \cdots, \Delta A_K\}$.

Finally, we approximate the first term of (12) as shown in (13) by

$$\sum_{k'=1}^{K} \sum_{t \in I_j} \omega_{k'} \cdot \frac{\partial}{\partial \omega_k} f_{k'}(\hat{y}_t^{(j)}, \hat{y}_{t-1}^{(j)}, \mathbf{v}_t^{(j)}) = \min(c, \sum_{k'=1}^{K} \frac{\Delta A_{k'}}{\Delta_{\omega_k^{t^\star}}}). \quad (14)$$

where $c > 0$ is a constant for avoiding excessively large derivatives.

Convergence analysis. Since LMVM uses the Armijo stepsize rule [2], convergence to local minimum of $L_c(W, \beta)$ of the inner loop is guaranteed under very relaxed assumptions, so long as the direction at any step is a descent direction. Our experience shows that the quasi-Newton directions generated by LMVM, based on the gradient in (11) and (12) always give descent directions of $L_c(W, \beta)$ at each iteration of LMVM. Therefore, the inner loop converges to a local minimum of $L_c(W, \beta)$ for a fixed β.

In the outer loop, we update the penalties by

$$c^0 = 10^{-9}, c^{m+1} = 2c^m \quad (15)$$

$$\beta_j^{m+1} = \beta_j^m + c^m g_j^+(W^m), \quad j = 1, ..., V \quad (16)$$

where m is the outer loop number.

Proposition 1. Suppose there is a *CLM* of (6) that satisfies the constraint qualification, then the ESP search algorithm finds a *CLM* of (6) in a finite number of outer loops, if the inner loop minimizes $L_c(W, \beta)$ at each major iteration.

We give a sketch the proof here. Consider any *CLM* W^* of (6). Since $W^\star$ is a *CLM*, we have, $f(W^\star) \leq f(W)$ and $g(W^*) \leq 0$, for any infeasible $W \in \mathcal{N}_{CN}(W^\star)$, we can show that there exists a finite threshold $\beta^* > 0$ such that

$$L_c(W^*, \beta^*) \leq L_c(W, \beta^*), \forall \text{ infeasible } W \in \mathcal{N}_{CN}(W^\star) \quad (17)$$

In a finite number of major iterations, β^m can exceed any finite threshold β^* using the update rules in (15) and (16). Let W^m be he point the inner loop stops. Then, as long as the inner loop minimizes $L_c(W, \beta)$, W^m is feasible. Thus, we have $g^+(W^m) = 0$ and, for any $\beta \in \mathcal{R}^V$,

$$L_c(W^m, \beta) \leq L_c(W^m, \beta^*) \leq L_c(W^m, \beta^m). \quad (18)$$

On the other hand, we have

$$L_c(W^m, \beta^m) \leq L_c(W, \beta^m), \ \forall W \in \mathcal{N}_{CN}(W^m). \quad (19)$$

Combining (18) and (19) and applying Theorem 1, we see that (W^m, β^m) is an ESP point and hence W^m is a *CLM*.

Noted that theoretically LMVM guarantees local optimality while Proposition 1 requires global optimality of the unconstrained solver. Empirically, our solver can always satisfy the cross-validation constraints in less than 20 major iterations. Since ESP search allows an extended region of suitable penalties, the solver is not sensitive to the penalty update rules. In contrast, it is difficult to control the single penalty in a traditional ℓ_1-penalty function when there are many constraints.

5. EXPERIMENTAL RESULTS

We present experimental results on two applications. We compare the original CRF-OPT package built in TAO with a new version that uses the constrained formulation and solver. We set $\gamma = 0.01$ for all runs.

5.1 Gene prediction

Given a DNA observation sequence, consisting of 'A', 'T', 'G', 'C' bases, the aim of gene prediction is to find out the protein coding regions, known as genes, and their associated components, including coding exons, start/stop exons, promoters, and poly-adenylation sites [3]. We first illustrate the improvement of the constrained approach on a simple model and then report results on a state-of-the-art gene predictor.

Figure 5 shows a finite state machine representation of the structure of genomic sequences in our implementation.

Table 1: The performance of original CRF, regularized CRF (CRF_r) and constrained CRF (CRF_c) for gene prediction.

Measure	Set 1			Set 2			Set 3			Set 4		
	CRF	CRF_r	CRF_c	CRF	CRF_r	CRF_c	CRF	CRF_r	CRF_c	CRF	CRF_r	CRF_c
Gene Sensitivity(%)	7.50	7.50	**10.00**	**2.50**	**2.50**	**2.50**	4.35	4.35	**6.52**	**2.04**	**2.04**	**2.04**
Gene Specificity(%)	6.98	6.82	**11.11**	2.33	2.13	**2.86**	3.85	4.35	**6.12**	2.13	1.79	**2.33**
Transcript Sensitivity(%)	7.50	7.50	**10.00**	**2.50**	**2.50**	**2.50**	4.35	4.35	**6.52**	**2.04**	**2.04**	**2.04**
Transcript Specificity(%)	6.98	6.82	**11.11**	2.33	2.13	**2.86**	3.85	4.35	**6.12**	2.13	1.79	**2.33**
Exon Sensitivity(%)	33.92	37.00	**37.89**	29.54	25.32	**31.22**	**34.43**	**34.43**	**34.43**	**32.41**	28.62	30.69
Exon Specificity(%)	40.53	**43.30**	42.57	35.53	30.00	**38.34**	36.86	**39.66**	38.84	32.41	33.88	**36.63**
Nucleotide Sensitivity(%)	89.96	90.35	**90.99**	86.53	84.21	**90.38**	90.20	**90.62**	90.00	**93.45**	92.84	93.35
Nucleotide Specificity(%)	87.84	**88.54**	87.06	**90.45**	89.64	88.66	89.74	89.54	**89.88**	**91.26**	90.67	90.35

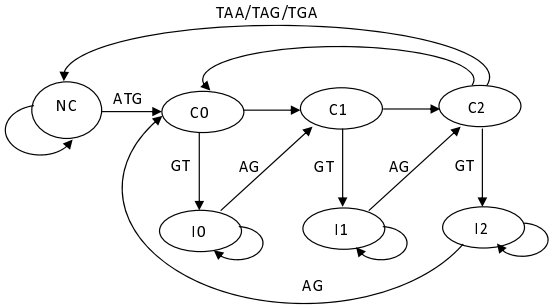

Figure 5: A finite state machine for gene prediction

In this model, each node represents a hidden state, such as exon (C), intron (I), and intergenic region (NC). In our model, the introns and exons are further divided into three phases according to the reading frame. Each edge represents a possible hidden state transition with required base observation.

The features used in our model is the state transition and base observation requirement as represented by the edges in Figure 5 and a 5-th order base emission information. In total we have around 7×4^6 features.

We test our algorithm on DNA sequences from the pathogenic fungus cryptococcus neoformans. We evaluate performance at the following four different levels: nucleotide-level, exon-level, transcript level, and gene level. For each level, we use the standard sensitivity (Sn) and specificity (Sp) measures defined as $Sn = \frac{TP}{TP+FN}$ and $Sp = \frac{TP}{TP+FP}$, where TP, FN, FP denote the numbers of true positive, false negative, and false positive labels, respectively [3].

In Table 1, we compare the performance of the constrained CRF and CRF with regularization. We set the regularization parameter $\delta^2 = 100$. We use all the 170 genes to construct four different data sets. We randomly split the data into 75% training data and 25% test data. For each set, three validation sets, each with 1/3 length of the training set, are constructed from the training data.

As shown in the result, our constrained CRF gives better performance than the unconstrained CRF model using the same feature set. Comparing to the regularization approach, our method has a notable performance boost, especially for the higher-level gene, transcript, and exon level measures. The higher level accuracy is typically more difficult to achieve than the nucleotide level accuracy. To achieve one gene level accuracy will require thousands of consecutive correct predictions on the nucleotide level. The computing time of constrained CRF is about 3 to 4 times longer than unconstrained CRF.

Although the gene prediction model used in previous experiment is still a much simplified one, the results show the effectiveness of the proposed work. In the following experiment, we integrate these techniques to CONTRAST, a state-of-the-art CRF-based gene predictor [8] to predict fly genomes. By effectively making use of multiple informants, CONTRAST is able to show substantial improvement over previous de novo gene predictors. The main component of this gene predictor is a CRF model with 33,003 features and 41 possible states. Similarly, we constructed validation sets, and added cross-validation guided constraints to the original formulation in CONTRAST. The performance is compared to CONTRAST without regularization, and CONTRAST with regularization terms added. For CONTRAST with regularization, a validation set is reserved and used to tune the regularization factor.

As shown in Table 2, both regularized formulation and our constrained formulation perform better comparing to the original unconstrained formulation. In most of the cases, our constrained formulation is able to improve the gene level accuracies by over 5% percentages, and better than the regularized formulation. We also find that our constrained formulation is able to converge with less iterations than that of the regularized formulation.

5.2 Stock price prediction

In this application, we try to predict if tomorrow's stock price will raise or decrease comparing to today's, based on historical stock price data. One thing to note is that, the prediction accuracies presented in this section is on the filtered stock price data, which has been smoothed, not the raw data.

In this task, the training sequence grows everyday, while the length of the testing sequence is only one, since we only try to predict for the next day. As the raw stock price data is noisy, we apply several preprocessing techniques, including stock screening, filtering, and transformation. Certain techniques are used later on to map the prediction results of smoothed data back into raw data.

Here we only focus on the prediction of **smoothed data**. For each stock, the preprocessed data is stored in sequence $(\mathbf{r}, \mathbf{y})$, where the observation $\mathbf{r}$ is the sequence of price change ratio, and $\mathbf{y}$ is the labeling sequence. Given the sequence $(r_1 \cdots r_T; y_1 \cdots y_T)$, we try to predict y_{T+1}. In our application, $y_t \in \{0, 1\}$ represents Raise or Fall. We use a 3rd order

Table 2: The performance of original CRF, regularized CRF (CRF_r) and constrained CRF (CRF_c) on CONTRAST.

Measure	Set 1			Set 2			Set 3			Set 4		
	CRF	CRF_r	CRF_c	CRF	CRF_r	CRF_c	CRF	CRF_r	CRF_c	CRF	CRF_r	CRF_c
Gene Sensitivity(%)	46.1	49.8	**52.1**	46.9	51.2	**53.8**	50.1	50.3	**52.2**	44.7	47.2	**48.5**
Gene Specificity(%)	46.1	51.3	**58.5**	53.2	60.9	**62.5**	55.3	57.4	**59.6**	42.5	57.2	**59.4**
Exon Sensitivity(%)	79.7	78.8	**80.0**	79.6	**82.5**	**82.5**	80.8	80.2	**81.3**	**80.9**	**80.9**	79.9
Exon Specificity(%)	67.8	68.0	**71.2**	70.7	75.1	**75.7**	71.1	73.0	**73.6**	65.5	75.5	**76.4**
Nucleotide Sensitivity(%)	**97.0**	96.6	96.3	96.1	**97.9**	97.5	96.4	**96.9**	96.8	**97.6**	97.4	96.5
Nucleotide Specificity(%)	79.5	85.8	**87.0**	86.5	88.5	**89.0**	86.2	**87.1**	86.9	75.5	88.4	**90.0**

feature function,

$$f(y_t, y_{t-1}, \mathbf{r}_t) = \begin{cases} 1 & lb_i < r_{t-3} < ub_i \text{ and} \\ & lb_j < r_{t-2} < ub_j \text{ and} \\ & lb_k < r_{t-1} < ub_k \text{ and} \\ & y_t = s, y_{t-1} = s' \\ 0 & \text{Otherwise} \end{cases}$$

We divide the price changing ratio into several ranges. Each atom of the feature function is true when the changing ratio falls into the corresponding range. When all the atoms return true, the feature is true. In total, 2000 features are included to aid learning.

We test our algorithm on 1741 stocks that are selected by our stock screening algorithm from all stocks in NASDAQ and NYSE, each of which contains the stock prices from 2002-02-01 to 2007-09-12. For each stock, when we try to predict the price tendency of day $T+1$, the data from day 1 to T is used for training. After that, the stock price on day $T+1$ and the real price tendency on that day is added to the training sequence for predicting day $T+2$. The process is repeated until the last day. We start our prediction on 2006-05-16. For our constrained formulation, the validation sequence is a sequence of data right before the predicted day. If we want to predict the price on day $T+1$, the validation sequence will be the data from day $T-V+1$ to T, where V is the length of the validation sequence. We have set $V = 100$.

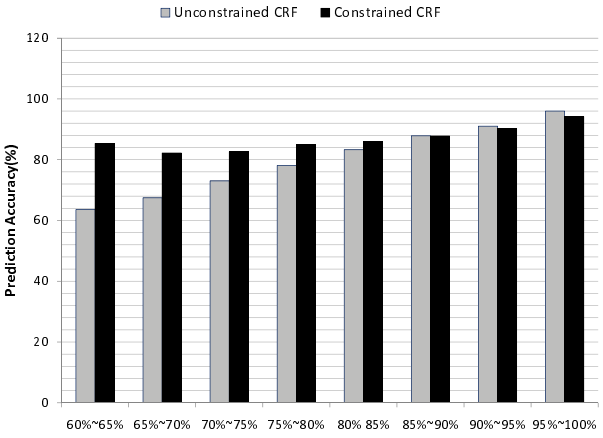

Figure 6: The prediction accuracies before and after adding constraints.

We classify the 1741 stocks into different categories according to the prediction accuracy of the unconstrained CRF. For stocks in each category, we calculate the average prediction accuracy of the original CRF model and the constrained CRF model. As shown in Figure 6, the constrained CRF gives much better accuracy when the original unconstrained formulation has low accuracy ($< 85\%$) and is comparable in other cases. The prediction accuracy of constrained CRF is consistently above 82% for all categories. Note that the accuracy is for predicting smoothed stock price curves. We can achieve around 55%-63% accuracy after translating the prediction back to raw prices. For each stock, the average training time is 25.5 seconds for the unconstrained CRF and 97.5 seconds for the constrained CRF.

6. CONCLUSIONS

In this paper, we present a novel constrained formulation for CRF training. Constraints reflecting cross-validation accuracies are added to prevent overfitting and to boost the generalizability of CRF models. The new formulation is difficult since it is discontinuous and nondifferentiable, and classical Lagrangian methods are not applicable. We develop a new constrained optimization algorithm based the recently proposed ESP theory to solve the problem. The constrained formulation and optimization algorithm are applied to gene prediction and stock price prediction tasks. The results show that our method is able to achieve significantly better prediction quality than unconstrained formulations, both with and without regularization. The CRF models with constrained formulation can improve the quality of a leading gene predictor as well as give high accuracy for predicting smoothed stock prices.

7. ACKNOWLEDGEMENT

This work is supported by NSF grant IIS-0713109, a DOE ECPI award, and a Microsoft Research New Faculty Fellowship.

8. REFERENCES

[1] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Prentice Hall, Englewood Cliffs, N.J., 1976.

[2] S. J. Benson, L. McInnes, J. Moré, and J. Sarich. TAO user manual (revision 1.8). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory, 2005.

[3] C. Burge. *Identification of genes in human genomic DNA*. PhD thesis, Stanford Univerisity, 1997.

[4] M. Chen, Y. Chen, and M. Brent. CRF-OPT: An efficient high-quality conditional random field solver. In *Proc. AAAI08*, 2008.

[5] S. Chen and R. Rosenfeld. A gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University, 1999.

[6] A. Culotta, D. Kulp, and A. McCallum. Gene prediction with conditional random fields. Technical Report UM-CS-2005-028, University of Massachusetts, Amherst, Apr. 2005.

[7] G. GRIMMETT. A theorem about random fields. *Bulletin of the London Mathematical Society*, 5:81–84, 1973.

[8] S. S. Gross, C. B. Do, M. Sirota, and S. Batzoglou. CONTRAST: a disriminative, phylogeny-free approach to multiple informant de novo gene prediction. *Genome Biology*, 8:R269, 2007.

[9] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proc. IJCAI*, pages 1137–1145, 1995.

[10] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML01*, 2001.

[11] W. I. Newman. Extension to the maximum entropy method. *IEEE Trans. on Information Theory*, (1):89–93, 1977.

[12] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. *IEEE Int Conference on Computer Vision*, 2:1150–1157, Jun. 2003.

[13] B. Roark, M. Saraclar, M. Collins, and M. Johnson. Discriminative language modeling with conditional random fields and the perceptron algorithm. *The 42nd Annual Meeting of the Association for Computational Linguistics*, 2004.

[14] R. Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech, and Language*, pages 187–228, 1996.

[15] S. Sarawagi and W. Cohen. Semi-markov conditional random fields for information extraction. In *Proc. NIPS*, 2004.

[16] F. Sha and F. Pereira. Shallow parsing with conditional random fields. *Human Language Technology*, 2003.

[17] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional models for contextual human motion recognition. In *IEEE International Conference on Computer Vision*, pages 1808–1815.

[18] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*. MIT Press, 2006.

[19] D. L. Vail, J. D. Lafferty, and M. M. Veloso. Feature selection in conditional random fields for activity recognition. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3379–3384, 2007.

[20] B. Wah and Y. Chen. Solving large-scale nonlinear programming problems by constraint partitioning. In *Proc. Constraint Programming*, pages 697–711, 2005.

[21] B. Wah and Y. Chen. Constrained partitioning in penalty formulations for solving temporal planning problems. *Artificial Intelligence*, 170(3):187–231, 2006.

[22] B. Wah and M. Qian. Constrained formulations for neural network training and their applications to solve the two-spiral problem. In *Proc. Fifth International Conference on Computer Science and Informatics*, volume 1, pages 598–601, 2000.