Gaussian Process Classification for Segmenting and Annotating Sequences

Yasemin Altun ALTUN@CS.BROWN.EDU
Department of Computer Science, Brown University, Providence, RI 02912 USA

Thomas Hofmann TH@CS.BROWN.EDU
Department of Computer Science, Brown University, Providence, RI 02912 USA
Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany

Alexander J. Smola ALEX.SMOLA@ANU.EDU.AU
Machine Learning Group, RSISE , Australian National University, Canberra, ACT 0200, Australia

Abstract

Many real-world classification tasks involve the prediction of multiple, inter-dependent class labels. A prototypical case of this sort deals with prediction of a sequence of labels for a sequence of observations. Such problems arise naturally in the context of annotating and segmenting observation sequences. This paper generalizes Gaussian Process classification to predict multiple labels by taking dependencies between neighboring labels into account. Our approach is motivated by the desire to retain rigorous probabilistic semantics, while overcoming limitations of parametric methods like Conditional Random Fields, which exhibit conceptual and computational difficulties in high-dimensional input spaces. Experiments on named entity recognition and pitch accent prediction tasks demonstrate the competitiveness of our approach.

1. Introduction

Multiclass classification refers to the problem of assigning class labels to instances where labels belong to some finite set of elements. Often, however, the instances to be labeled do not occur in isolation, but rather in observation sequences. One is then interested in predicting the joint label configuration, i.e. the sequence of labels corresponding to a sequence of observations, using models that take possible interdependencies between label variables into account. This scenario subsumes problems of sequence segmentation and annotation, which are ubiquitous in areas such as natural language processing, speech recognition, and computational biology.

The most common approach to sequence labeling is based on Hidden Markov Models (HMMs), which define a generative probabilistic model for labeled observation sequences. In recent years, the state-of-the-art method for sequence learning is Conditional Random Fields (CRFs) introduced by Lafferty et al. (Lafferty et al., 2001). In most general terms, CRFs define a conditional model over label sequences given an observation sequence in terms of an exponential family; they are thus a natural generalization of logistic regression to the problem of label sequence prediction. Other related work on this subject includes Maximum Entropy Markov models (McCallum et al., 2000) and the Markovian model of (Punyakanok & Roth, 2000).

There have also been attempts to extend other discriminative methods such as AdaBoost (Altun et al., 2003a), perceptron learning (Collins, 2002), and Support Vector Machines (SVMs) (Altun et al., 2003b; Taskar et al., 2004) to the label sequence learning problem. The latter have experimentally compared favorably to other discriminative methods, including CRFs. Moreover, they have the conceptual advantage of being compatible with implicit data representations via kernel functions.

In this paper, we investigate the use of Gaussian Process (GP) classification (Gibbs & MacKay, 2000; Williams & Barber, 1998) for label sequences. The main motivation for pursuing this direction is to com-

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the first author.

bine the best of both worlds from CRFs and SVMs. More specifically, we would like to preserve the main strength of CRFs, which we see in its rigorous probabilistic semantics. There are two important advantages of a probabilistic model. First, it is very intuitive to incorporate prior knowledge within a probabilistic framework. Second, in addition to predicting the best labels, one can compute posterior label probabilities and thus derive confidence scores for predictions. This is a valuable property in particular for applications requiring a cascaded architecture of classifiers. Confidence scores can be propagated to subsequent processing stages or used to abstain on certain predictions. The other design goal is the ability to use kernel functions in order to construct and learn in Reproducing Kernel Hilbert Spaces (RKHS), thereby overcoming the limitations of (finite-dimensional) parametric statistical models.

A second, independent objective of our work is to gain clarification with respect to two aspects on which CRFs and the SVM-based methods differ, the first aspect being the loss function (logistic loss vs. hinge loss), and the second aspect being the mechanism used for constructing the hypothesis space (parametric vs. RKHS).

GPs are non-parametric tools to perform Bayesian inference, which – like SVMs – make use of the *kernel trick* to work in high (possibly infinite) dimensional spaces. Like other discriminative methods, GPs predict single variables and do not take into account any dependency structure in case of multiple label predictions. Our goal is to generalize GPs to predict label sequences. While computationally demanding, recent progress on sparse approximation methods for GPs, e.g. (Csat'o & Opper, 2002; Smola & Bartlett, 2000; Seeger et al., 2003), suggest that scalable GP label sequence learning may be an achievable goal. Exploiting the compositionality of the kernel function, we derive a gradient-based optimization method for GP sequence classification. Moreover, we present a column generation algorithm that performs a sparse approximation of the solution.

The rest of the paper is organized as follows: In Section 2, we introduce Gaussian Process classification. Then, we present our formulation of Gaussian Process sequence classification (GPSC) in Section 3 and describe the proposed optimization algorithms in Section 4. Finally, we report some experimental results using real-world data for named entity classification and pitch accent prediction in Section 5.

2. Gaussian Process Classification

In supervised classification, we are given a training set of n labeled instances or observations $(\mathbf{x}_i, y_i)$ with $y_i \in \{1, \ldots, m\}$, drawn i.i.d. from an unknown, but fixed, joint probability distribution $p(\mathbf{x}, y)$. We denote the training observations and labels by $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$, respectively.

GP classification constructs a two-stage model for the conditional probability distribution $p(y|\mathbf{x})$ by introducing an intermediate, unobserved stochastic process $\mathbf{u} \equiv (u(\mathbf{x}, y))$ where $u(\mathbf{x}, y)$ can be considered a *compatibility* measure of an observation $\mathbf{x}$ and a label y. Given an instantiation of the stochastic process, we assume that the conditional probability $p(y|\mathbf{x}, \mathbf{u})$ only depends on the values of $\mathbf{u}$ at the input $\mathbf{x}$ via a multinomial response model, i.e.

$$p(y|\mathbf{x}, \mathbf{u}) = p(y|u(\mathbf{x}, \cdot)) = \frac{\exp(u(\mathbf{x}, y))}{\sum_{y'=1}^{m} \exp(u(\mathbf{x}, y'))} \qquad (1)$$

It is furthermore assumed that the stochastic process $\mathbf{u}$ is a zero mean Gaussian process with covariance function C, typically a kernel function. An additional assumption typically made in multiclass GP classification is that the processes $\mathbf{u}(\cdot, y)$ and $\mathbf{u}(\cdot, y')$ are uncorrelated for $y \neq y'$ (Williams & Barber, 1998).

For notational convenience, we will identify $\mathbf{u}$ with the relevant restriction of $\mathbf{u}$ to the training patterns $\mathbf{X}$ and represent it as a $n \times m$ matrix. For simplicity we will (in slight abuse of notation) also think of $\mathbf{u}$ as a vector with multi-index (i, y). Moreover we will denote by $\mathbf{K}$ the kernel matrix with entries[1] $K_{(i,y),(j,y')} = C((\mathbf{x}_i, y), (\mathbf{x}_j, y'))$. Notice that under the above assumptions $\mathbf{K}$ has a block diagonal structure with blocks $\mathbf{K}(y) = (K_{ij}(y))$, $K_{ij}(y) \equiv C_y(\mathbf{x}_i, \mathbf{x}_j)$, where C_y is a class-specific covariance function.

Following a Bayesian approach, the prediction of a label for a new observation $\mathbf{x}$ is obtained by computing the posterior probability distribution over labels and selecting the label that has the highest probability:

$$p(y|\mathbf{X}, \mathbf{y}, \mathbf{x}) = \int p(y|\mathbf{u}(\mathbf{x}, \cdot))\, p(\mathbf{u}|\mathbf{X}, \mathbf{y})\, d\mathbf{u} \qquad (2)$$

Thus, one needs to integrate out all $n \cdot m$ latent variables of $\mathbf{u}$. Since this is in general intractable, it is common to perform a saddle-point approximation of the integral around the optimal point esti-

[1]Here and below, we will make extensive use of multi-indices. We will put parentheses around a comma-separated list of indices to denote a multi-index and use two comma-separated multi-indices to refer to matrix elements.

mate, which is the maximum a posterior (MAP) estimate: $p(y|\mathbf{X},\mathbf{y},\mathbf{x}) \approx p(y|\mathbf{u}^{\text{map}}(\mathbf{x},\cdot))$ where $\mathbf{u}^{\text{map}} = \text{argmax}_{\mathbf{u}} \log p(\mathbf{u}|\mathbf{X},\mathbf{y})$. Exploiting the conditional independence assumptions, the posterior of $\mathbf{u}$ can – up to a multiplicative constant – be written as

$$p(\mathbf{u}|\mathbf{X},\mathbf{y}) \propto p(\mathbf{u}) \prod_{i=1}^{n} p(y_i|\mathbf{u}(\mathbf{x}_i,\cdot)) \tag{3}$$

Combining the GP prior over $\mathbf{u}$ and the conditional model in (1) yields the more specific expression

$$\log p(\mathbf{u}|\mathbf{X},\mathbf{y}) = \sum_{i=1}^{n} \left[u(\mathbf{x}_i, y_i) - \log \sum_{y} \exp(u(\mathbf{x}_i, y)) \right] - \frac{1}{2}\mathbf{u}^T\mathbf{K}^{-1}\mathbf{u} + \text{const.} \tag{4}$$

The Representer Theorem (Kimeldorf & Wahba, 1971) guarantees that the maximizer of (4) is of the form

$$u^{\text{map}}(\mathbf{x}_i, y) = \sum_{j=1}^{n} \sum_{y'=1}^{m} \alpha_{(j,y')} K_{(i,y),(j,y')} \tag{5}$$

with suitably chosen coefficients α. In the block diagonal case, $K_{(i,y),(j,y')} = 0$ for $y \neq y'$ and this reduces to the simpler form $u^{\text{map}}(\mathbf{x}_i, y) = \sum_{j=1}^{n} \alpha_{(j,y)} C_y(\mathbf{x}_i, \mathbf{x}_j)$.

Using the representation in (5), we can rewrite the optimization problem as an objective R parameterized by α. Let $e_{(i,y)}$ be the (i,y)-th unit vector, then $\alpha^T\mathbf{K}e_{(i,y)} = \sum_{j,y'} \alpha_{(j,y')} K_{(i,y),(j,y')}$ and the negative of Eq. (4) can be written as follows:

$$R(\alpha|\mathbf{X},\mathbf{y}) = \alpha^T\mathbf{K}\alpha - \sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i, \alpha) \tag{6}$$

$$= \alpha^T\mathbf{K}\alpha - \sum_{i=1}^{n} \alpha^T\mathbf{K}e_{(i,y_i)} + \sum_{i=1}^{n} \log \sum_{y} \exp(\alpha^T\mathbf{K}e_{(i,y)})$$

A comparison between (6) and a similar multiclass SVM formulation (Crammer & Singer, 2001; Weston & Watkins, 1999) clarifies the connection between GP classification and SVMs. Their difference lies primarily in the utilized loss functions: logistic loss vs. hinge loss. Because the hinge loss truncates values smaller than ϵ to 0, it enforces sparseness in terms of the α parameters. This is not the case for logistic regression as well as other choices of loss functions.[2]

For non-linear link functions like the one induced by Eq. (1), $\mathbf{u}^{\text{map}}$ cannot be found analytically and one has to resort to approximate solutions. Various approximation schemes have been studied to that extent: Laplace approximation (Williams & Barber, 1998; Williams & Seeger, 2000), variational methods (Jaakkola & Jordan, 1996), mean field approximations (Opper & Winther, 2000), and expectation propagation (Minka, 2001; Seeger et al., 2003). Performing these methods usually involves the computation of the Hessian matrix as well as the inversion of $\mathbf{K}$, a $nm \times nm$ matrix, which is not tractable for large data sets (of size n) and/or large label sets (of size m). Several techniques have been proposed to approximate $\mathbf{K}$ such that the inversion of the approximating matrix is tractable (cf. (Schölkopf & Smola, 2002) for references on such methods). One can also try to solve (6) using greedy optimization methods as proposed in (Bennett et al., 2002).

3. GP Sequence Classification (GPSC)

3.1. Sequence Labeling and GPC

In sequence classification, our goal is to learn a discriminant function for sequences, i.e. a mapping from observation sequences $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^t, \ldots, \mathbf{x}^T)$ to label sequences $\mathbf{y} = (y^1, y^2, \ldots, y^t, \ldots, y^T)$. There exists a label $y^t \in \Sigma = \{1, \ldots, r\}$ for every observation $\mathbf{x}^t$ in the sequence. Thus, we have T multiclass classification problems. Because of the sequence structure of the labels (i.e. every label y^t depends on its neighboring labels), one needs to solve these T classification problems jointly. Then, the problem can be considered as a multiclass classification where for an observation sequence of length l, the possible label set $\mathcal{Y}$ is of size $m = r^l$.[3] We call Σ label set of observations or *micro-label set*, and $\mathcal{Y}$ the set of label sequences of observation sequences or *macro-label set*.

We assume that a training set of n labeled sequences $Z \equiv \{(\mathbf{X}_i, \mathbf{y}_i)|i = 1, \ldots, n\}$ is available. Using the notation introduced in the context of GP classification, we define $p(\mathbf{y}_i|u(\mathbf{X}_i))$ as in (1), treating every macro label as a separate label in GP multiclass classification and using the whole sequence $\mathbf{X}_i$ as the input.

3.2. Kernels for Labeled Sequences

The fundamental design decision is then the engineering of the kernel function k that determines the kernel matrix $\mathbf{K}$. Notice that the use of a block diagonal kernel matrix is not an option in the current setting, since it would prohibit generalizing across label sequences that differ in as little as a single micro-label.

[2]Several studies focused on finding sparse solutions of Eq. (6) or optimization problems similar to Eq. (6) (Bennett et al., 2002; Girosi, 1997; Smola & Schölkopf, 2000).

[3]For notational convenience we will assume that all training sequences are of the same length l.

We define the kernel function for labeled sequences with respect to the feature representation. Inspired by HMMs, we use two types of features: Features that capture the dependency of the micro-labels on the attributes of the observations $\Phi(\mathbf{x}^s)$ and features that capture the inter-dependency of micro-labels. As in other discriminative methods, $\Phi(\mathbf{x}^s)$ can include overlapping attributes of $\mathbf{x}^s$ as well as attributes of observations $\mathbf{x}^t$ where $t \neq s$. Using stationarity, the inner product between the feature vectors of two observation sequences can be stated as: $k = k^1 + k^2$, where

$$k^1((\mathbf{X},\mathbf{y}),(\bar{\mathbf{X}},\bar{\mathbf{y}})) \equiv \sum_{s,t} [\![y^s = \bar{y}^t]\!] \langle \Phi(\mathbf{x}^s), \Phi(\bar{\mathbf{x}}^t) \rangle \quad (7a)$$

$$k^2((\mathbf{X},\mathbf{y}),(\bar{\mathbf{X}},\bar{\mathbf{y}})) \equiv \sum_{s,t} [\![y^s = \bar{y}^t \wedge y^{s+1} = \bar{y}^{t+1}]\!] \quad (7b)$$

k^1 couples observations in both sequences that are classified with the same micro-labels at respective positions. k^2 simply counts the number of consecutive label pairs both label sequences have in common (irrespective of the inputs). One can generalize (7) in various ways, e.g. by using higher order terms between micro-labels in both contributions, without posing major conceptual challenges.

k is a linear kernel function for labeled sequences. This can be generalized to non-linear kernel functions for labeled sequences by replacing $\langle \Phi(\mathbf{x}^s), \Phi(\bar{\mathbf{x}}^t) \rangle$ with a standard kernel function defined over input patterns.

We can naively follow the same line of argumentation as in the GPC case of Section 2, evoke the Representer Theorem and ultimately arrive at the objective in (6). Since we need it for subsequent derivations, we will restate the objective here

$$R(\alpha|Z) = \alpha^T \mathbf{K} \alpha - \sum_{i=1}^{n} \alpha^T \mathbf{K} e_{(i,\mathbf{y}_i)} + \sum_{i=1}^{n} \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp\left(\alpha^T \mathbf{K} e_{(i,\mathbf{y})}\right) \quad (8)$$

Notice that in the third term, the sum ranges over the macro-label set, $\mathcal{Y}$, which grows exponentially in the sequence length. Therefore, this view suffers from the large cardinality of $\mathcal{Y}$. In order to re-establish tractability of this formulation, we use a trick similar to the one deployed in (Taskar et al., 2004) and reparametrize the objective in terms of an equivalent lower dimensional set of parameters. The crucial observation is that the definition of k in (7) is homogeneous (or stationary). Thus, the absolute positions of patterns and labels in the sequence are irrelevant. This observation can be exploited by re-arranging the sums inside the kernel function with the outer sums, i.e. the sums in the objective function.

3.3. Exploiting Kernel Structure

In order to carry out this reparameterization more formally we proceed in two steps. The first step consists of finding an appropriate low-dimensional summary of α. In particular, we are looking for a parameterization that does not scale with $m = r^l$. The second step consists of re-writing the objective function in terms of these new parameters.

As we will prove subsequently, the following linear map $\mathbf{\Lambda}$ extracts the information in α that is relevant for solving (8):

$$\gamma \equiv \mathbf{\Lambda} \alpha, \quad \mathbf{\Lambda} \in \{0,1\}^{n \cdot l \cdot r^2 \times n \cdot m} \quad (9)$$

where

$$\lambda_{(j,t,\sigma,\tau),(i,\mathbf{y})} \equiv \delta_{ij} [\![y^t = \sigma \wedge y^{t+1} = \tau]\!] \quad (10)$$

Notice that each variable $\lambda_{(j,t,\sigma,\tau),(i,y)}$ encodes whether the input sequence is the j-th training sequence and whether the label sequence y contains micro-labels σ and τ at position t and $t+1$, respectively. Hence, $\gamma_{(j,t,\sigma,\tau)}$ is simply the sum of all $\alpha_{(j,y)}$ over label sequences $\mathbf{y}$ that contain the $\sigma\tau$-motif at position t.

We define two reductions derived from γ via further linear dimension reduction,

$$\gamma^{(1)} \equiv \mathbf{P}\gamma, \text{ with } P_{(i,s,\sigma),(j,t,\tau,\rho)} = \delta_{ij}\delta_{st}\delta_{\sigma\tau}\,, \quad (11a)$$

$$\gamma^{(2)} \equiv \mathbf{Q}\gamma, \text{ with } Q_{(i,\sigma,\zeta),(j,t,\tau,\rho)} = \delta_{ij}\delta_{\sigma\tau}\delta_{\zeta\rho}\,. \quad (11b)$$

Intuitively, $\gamma^{(2)}_{(i,\sigma,\tau)}$ is the sum of all $\alpha_{(i,\mathbf{y})}$ over every position in the sequence $\mathbf{y}$ that contains $\sigma\tau$-motif. $\gamma^{(1)}_{i,s,\sigma}$, on the other hand, is the sum of all $\alpha_{(i,\mathbf{y})}$ that has σ micro-label at position s in macro-label $\mathbf{y}$.

We can now show how to represent the kernel matrix using the previously defined matrices $\mathbf{\Lambda}$, $\mathbf{P}$, $\mathbf{Q}$ and the gram matrix $\mathbf{G}$ with $G_{(i,s),(j,t)} = g(\mathbf{x}_i^s, \mathbf{x}_j^t)$.

Proposition 1. *With the definitions from above:*

$$\mathbf{K} = \mathbf{\Lambda}^T \mathbf{K}' \mathbf{\Lambda}, \quad \mathbf{K}' \equiv \left(\mathbf{P}^T \mathbf{H} \mathbf{P} + \mathbf{Q}^T \mathbf{Q}\right)$$

where $\mathbf{H} = \operatorname{diag}(\mathbf{G}, \ldots, \mathbf{G})$.

Proof. By elementary comparison of coefficients. □

We now have r^2 parameters for every observation $\mathbf{x}^s$ in the training data (nlr^2 parameters) and we can rewrite the objective function in terms of these variables:

$$R(\gamma|\mathbf{X},\mathbf{y}) = \gamma^T \mathbf{K}' \gamma - \sum_{i=1}^{n} \gamma^T \mathbf{K}' \mathbf{\Lambda} e_{(i,\mathbf{y}_i)} + \sum_{i=1}^{n} \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp\left(\gamma^T \mathbf{K}' \mathbf{\Lambda} e_{(i,\mathbf{y})}\right) \quad (12)$$

3.4. GPSC and Other Label Sequence Learning Methods

We now briefly point out the relationship between our approach and the previous discriminative methods of sequence learning, in particular, CRFs, HM-SVMs and MMMs.

CRF is a natural generalization of logistic regression to label sequence learning. The probability distribution over label sequences given an observation sequence is given in Eq. (1), where $u(\mathbf{X},\mathbf{y}) = \langle \theta, \Psi(\mathbf{X},\mathbf{y})\rangle$ is a linear discriminative function over some feature representation Ψ parameterized with θ. The objective function of CRFs is the minimization of the negative conditional likelihood of training data. To avoid overfitting, it is common to multiply the conditional likelihood by a Gaussian with zero mean and diagonal covariance matrix $\mathbf{K}$, resulting in an additive term in log scale.

$$\log p(\theta|\mathbf{X},\mathbf{y}) = -\sum_{i=1}^{n} \log p(\mathbf{y}_i|\mathbf{X}_i,\theta) + \theta^T \mathbf{K}\theta \qquad (13)$$

From a Bayesian point of view, CRFs assume a uniform prior $p(\mathbf{u})$, if there is no regularization term. When regularized, CRFs define a Gaussian distribution over a finite vector space θ. In GPSC, on the other hand, the prior is defined as a Gaussian distribution over the function space of possibly infinite dimension. Thus, GPSC generalizes CRFs by defining a more sophisticated prior on the discriminative function u. This prior leads to the ability of using kernel function in order to construct and learn over Reproducing Kernel Hilbert Spaces. So, GPSC, a non-parametric Bayesian inference tool for sequence labeling, can overcome the limitations of CRFs, parametric (linear) statistical models. When the kernel that defines the covariance matrix $\mathbf{K}$ in GPSC is linear, u in both models become equivalent.

The difference between SVM and GP approaches to sequence learning is the utilized loss function over the training data, i.e. hinge loss vs. log loss. GPSC objective function parameterized with α (Eq. (8)) corresponds to HM-SVMs where the number of parameters scale exponentially with the length of sequences. The objective function parameterized with γ (Eq. (12)) corresponds to MMMs, where the number of parameters scale only linearly.

4. GPSC Optimization Algorithm

4.1. A Dense Algorithm

Using optimization methods described in Section 2 requires the computation of the Hessian matrix. In sequence labeling, this corresponds to computing the expections of micro-labels within different cliques, which is not tractable to compute exactly for large training sets. In order to minimize R with respect to γ, we propose a 1^{st} order exact optimization method, which we call Dense Gaussian Process Sequence Classification (DGPS).

It is well-known that the derivatives of the log partition function with respect to γ is simply the expectation of sufficient statistics:

$$\nabla_\gamma \left[\log \sum_{\mathbf{y}\in\mathcal{Y}} \exp\left(\gamma^T \mathbf{K}'\mathbf{\Lambda} e_{(i,\mathbf{y})}\right)\right] = \mathbf{E}_Y\left[\mathbf{\Lambda} e_{(i,Y)}\right] \qquad (14)$$

where $\mathbf{E}_Y$ denotes an expectation with respect to the conditional distribution of the label sequence $\mathbf{y}$ given the observation sequence $\mathbf{X}_i$. Then, the gradients of R is trivially given by:

$$\nabla_\gamma R = 2\mathbf{K}'\gamma - \sum_{i=1}^{n} \mathbf{K}'\mathbf{\Lambda} e_{(i,\mathbf{y}_i)} + \sum_{i=1}^{n} \mathbf{K}'\mathbf{E}_Y\left[\mathbf{\Lambda} e_{(i,Y)}\right] \qquad (15)$$

The remaining challenge is to come-up with an efficient way to compute the expectations. First of all, let us more explicitly examine these quantities:

$$\mathbf{E}_Y\left[(\mathbf{\Lambda} e_{(i,Y)})_{(j,t,\sigma,\tau)}\right] = \delta_{ij}\mathbf{E}_Y\left[[\![Y^t=\sigma \wedge Y^{t+1}=\tau]\!]\right] \qquad (16)$$

In order to compute the above expectations one can once again exploit the structure of the kernel and is left with the problem of computing probabilities for every neighboring micro-label pair (σ,τ) at positions $(t,t+1)$ for all training sequences $\mathbf{X}_i$. The latter can be accomplished by performing the forward-backward algorithm over the training data using the transition probability matrix $\mathbf{T}$ and the observation probability matrices $\mathbf{O}^{(i)}$, which are simply decompositions and reshapings of $\mathbf{K}'$:

$$\bar{\gamma}^{(2)} \equiv \mathbf{R}\gamma^{(2)}, \text{ with } R_{(\sigma,\zeta),(i,\tau,\rho)} = \delta_{\sigma\tau}\delta_{\zeta\rho} \qquad (17a)$$

$$\mathbf{T} \equiv [\bar{\gamma}^{(2)}]_{r,r} \qquad (17b)$$

$$\mathbf{O}^{(i)} = [\gamma^{(1)}]_{n\cdot l,r}\mathbf{G}_{(i,.),(.,.)} \qquad (17c)$$

where $[x]_{m,n}$ denotes the reshaping operation of a vector x into an $m*n$ matrix, $\mathbf{A}_{I,J}$ denotes the $|I|*|J|$ sub-matrix of A and $(.)$ denotes the set of all possible indices.

A single optimization step of DGPS is described in Algorithm 1. The complexity of one optimization step is $O(t^2)$ dominated by the forward-backward algorithm

Algorithm 1 One optimization step of Dense Gaussian Process Sequence Classification (DGPS)

```
Require: Training data $(\mathbf{X}_i, \mathbf{y}_i)_{i=1:n}$; Proposed parameter values $\gamma_c$
1: Initialize $\gamma_c^{(1)}, \gamma_c^{(2)}$ (Eq. (11)).
2: Compute $\mathbf{T}$ wrt $\gamma_c^{(2)}$ (Eq. (17a), Eq. (17b)).
3: for $i = 1, \ldots, n$ do
4:     Compute $\mathbf{O}^{(i)}$ wrt $\gamma_c^{(1)}$ (Eq. (17c)).
5:     Compute $p(y_i|\mathbf{X}_i, \gamma_c)$ and $\mathbf{E}_Y\left[[\![Y^t = \sigma \wedge Y^{t+1} = \tau]\!]\right]$ for all $t, \sigma, \tau$ via forward-backward algorithm using $\mathbf{O}^{(i)}$ and $\mathbf{T}$
6: end for
7: Compute $\nabla_\gamma R$ (Eq. (15)).
```

over all instances where $t = nlr^2$. We propose to use a quasi-Newton method for the optimization process. Then, the overall complexity is given by $O(\eta t^2)$ where $\eta < t^2$. The memory requirement is given by the size of γ, $O(t)$.

During inference, one can find the most likely label sequence for an observation sequence $\mathbf{X}$ by performing Viterbi decoding using the transition and observation probability matrices described above.

4.2. A Sparse Algorithm

While the above method is attractive for small data sets, the computation or the storage of $\mathbf{K}'$ poses a serious problem when the data set is large. Also, classification of a new observation involves evaluating the covariance function at nl data points, which is more than acceptable for many applications. Hence, as in the case of standard Gaussian Process Classification discussed in Section 2, one has to find a method for sparse solutions in terms of the γ parameters to speed up the training and prediction stages.

We propose a sparse greedy method, Sparse Gaussian Process Sequence Classification (SGPS), that is similar to the method presented by (Bennett et al., 2002). SGPS starts with an empty matrix $\hat{\mathbf{K}}$. At each iteration, SGPS selects a training instance $\mathbf{X}_i$ and computes the gradients of the parameters associated with $\mathbf{X}_i$, $\gamma_{(i,.)}$, to select the steepest descent direction(s) of R over this subspace. Then $\hat{\mathbf{K}}$ is augmented with these columns and SGPS performs optimization of the current problem using a Quasi-Newton method. This process is repeated until the gradients vanish (i.e. they are smaller than a threshold value η) or a maximum number of γ coordinates, p, are selected (i.e. some sparseness level is achieved). Since the bottleneck of this method is the computation of the expectations, $\mathbf{E}_Y\left[[\![Y^t = \sigma \wedge Y^{t+1} = \tau]\!]\right]$, we pick the steepest d directions, once the expectations are computed.

One has two options to compute the optimal γ at every iteration: by updating all of the γ parameters selected until now, or alternatively, by updating only the parameters selected in the last iteration. We prefer the latter because of its less expensive iterations. This approach is in the spirit of a boosting algorithm or the cyclic coordinate optimization method.

Algorithm 2 Sparse Gaussian Process Sequence Classification (SGPS) algorithm.

```
Require: Training data $(\mathbf{X}_i, \mathbf{y}_i)_{i=1:n}$; Maximum number of coordinates to be selected, $p$, $p < nlr^2$; Threshold value $\eta$ for gradients
1: $\mathbf{K} \leftarrow []$
2: for $i = 1, \ldots, n$ do
3:     Compute $\nabla_{\gamma_{(i,.)}} R$ (Equation 15).
4:     $s \leftarrow$ Steepest $d$ directions of $\nabla_{\gamma_{(i,.)}} R$
5:     $\hat{\mathbf{K}} \leftarrow [\hat{\mathbf{K}}; \mathbf{K} e_s]$
6:     Optimize $R$ wrt $s$.
7:     Return if $\nabla_\gamma < \eta$ or $p$ coordinates selected.
8: end for
```

SGPS is described in Algorithm 2. Its complexity is $O(p^2 t)$ where p is the maximum number of coordinates allowed.

5. Experiments

5.1. Pitch Accent Prediction

Pitch Accent Prediction is the task of identifying more prominent words in a sentence. The micro-label set is of size 2, accented and not-accented. We used phonetically hand-transcribed Switchboard corpus consisting of 1824 sentences (13K words) (Greenberg et al., 1996). We extracted probabilistic, acoustic and textual information from the current, previous and next words for every position in the training data. We used 1^{st} order Markov features to capture the dependencies between neighboring labels.

We compared the performance of CRFs and HM-SVMs with the GPSC dense and sparse methods according to their test accuracy in 5-fold cross validation. CRFs were regularized and optimized using limited memory BFGS, a limited memory Quasi-Newton optimization method. When performing experiments on DGPS, we used polynomial kernels with different degrees (denoted with DGPS**X** in Figure 1a where $\mathbf{X} \in \{1, 2, 3\}$ is the degree of the polynomial kernel). We used third order polynomial kernel in HM-SVMs (denoted with SVM3 in Figure 1). As expected, CRFs

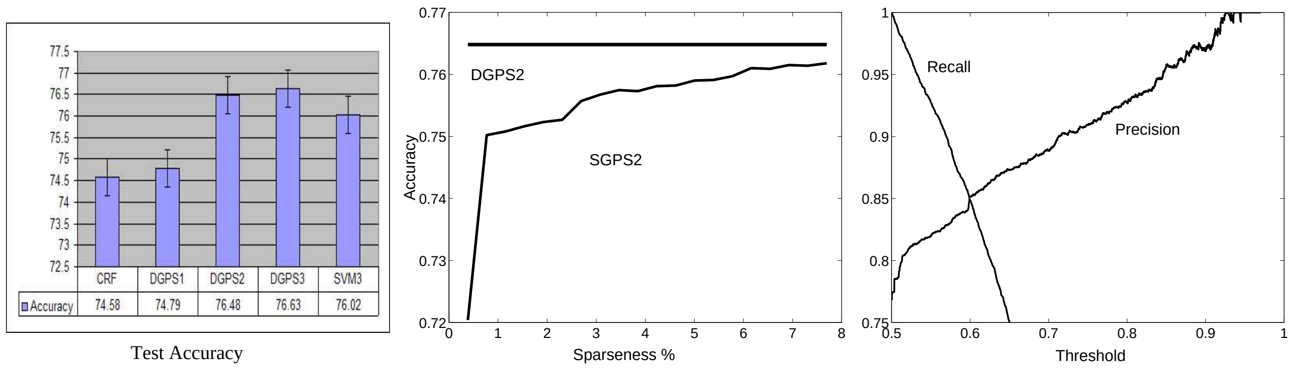

Figure 1. Pitch Accent Prediction task results a) Test accuracy of Pitch Accent Prediction task over a window of size 3 using 5-fold cross validation. b) Test accuracy of Pitch Accent Prediction w.r.t. the sparseness of the solution. c) Precision-Recall curves for different threshold probabilities to abstain.

and DGPS1 performed very similar. When 2^{nd} order features were incorporated implicitly using second degree polynomial kernel (DGPS2), the performance increased dramatically. Extracting 2^{nd} order features explicitly results in a 12 million dimensional feature space, where CRFs slow down dramatically. We observed that 3^{rd} order features do not provide significant improvement over DGPS2. HM-SVM3 performs slightly worse than DGPS2.

To investigate how the sparsity of SGPS affects its performance, we report the test accuracy with respect to the sparseness of SGPS solution in Figure 1b. Sparseness is measured by the percentage of the parameters selected by SGPS. The straight line is the performance of DGPS using second degree polynomial kernel. Using 1% of the parameters, SGPS achieves 75% accuracy (1.48% less than the accuracy of DGPS). When 7.8% of the parameters are selected, the accuracy is 76.18% which is not significantly different than the performance of DGPS (76.48%). We observed that these parameters were related to 6.2% of the observations along with 1.13 label pairs on average. Thus, during inference one needs to evaluate the kernel function only at 6% of the observations which reduces the inference time dramatically.

In order to experimentally verify how useful the predictive probabilities are as confidence scores, we forced DGPS to abstain from predicting a label when the probability of a micro-label is lower than a threshold value. In Figure 1c, we plot precision-recall values for different thresholds. We observed that the error rate for DGPS decreased 8.54%, abstaining on 14.93% of the test data. The improvement on the error rate shows the validity of the probabilities generated by DGPS.

5.2. Named Entity Recognition

Named Entity Recognition (NER), a subtask of Information Extraction, is finding phrases containing names in a sentences. The micro-label set consists of the beginning and continuation of person, location, organization and miscellaneous names and non-name. We used a Spanish newswire corpus, which was provided for the Special Session of CoNLL 2002 on NER, to randomly select 1000 sentences (21K words). We used the word and its spelling properties of the current, previous and next observations.

	DGPS1	DGPS2	SGPS2	CRF	CRF-B
Error	4.58	4.39	4.48	4.92	4.56

Table 1. Test error of NER over a window of size 3 using 5-fold cross validation.

The experimental setup was similar to pitch accent prediction task. We compared the performance of CRFs with and without the regularizer term (CRF-R, CRF) with the GPSC dense and sparse methods. Qualitatively, the behavior of the different optimization methods is comparable to the pitch accent prediction task. The results are summarized in Table 1. Second degree polynomial DGPS outperformed the other methods. We set the sparseness parameter of SGPS to 25%, i.e. $p = 0.25nlr^2$, where $r = 9$ and $nl = 21$K on average. SGPS with 25% sparseness achieves an accuracy that is only 0.1% below DGPS. We observed that 19% of the observations are selected along with 1.32 label pairs on average, which means that one needs to compute only one fifth of the gram matrix.

We also tried a sparse algorithm that does not exploit the kernel structure and optimizes Equation 8 to obtain sparse solutions in terms of observation sequences $\mathbf{X}$ and label sequence $\mathbf{y}$, as opposed to SPGS, where the sparse solution is in terms of observations and label pairs. This method achieved 92.7% of accuracy, hence, was clearly outperformed by all the other methods.

6. Conclusion and Future Work

We presented GPSC, a generalization of Gaussian Process classification to label sequence learning problem. This method combines the advantages of the rigorous probabilistic semantics of CRFs and overcomes *the curse of dimensionality* problem using kernels in order to construct and learn over RKHS. The experiments on named entity recognition show the competitiveness and the experiments on pitch accent prediction show the superiority of our approach in terms of the achieved error rate. We also experimentally verified the usefulness of the probabilities obtained from GPSC.

References

Altun, Y., Hofmann, T., & Johnson, M. (2003a). Discriminative learning for label sequences via boosting. *Advances in Neural Information Processing Systems.*

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003b). Hidden markov support vector machines. *20th International Conference on Machine Learning (ICML).*

Bennett, K., Momma, M., & Embrechts, J. (2002). Mark: A boosting algorithm for heterogeneous kernel models. *Proceedings of SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Collins, M. (2002). Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. *Empirical Methods of Natural Language Processing (EMNLP).*

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, *2*, 265–292.

Csat'o, L., & Opper, M. (2002). Sparse on-line Gaussian Processes. *Neural Computation*, *14*, 641–668.

Gibbs, M. N., & MacKay, D. J. C. (2000). Variational Gaussian Process Classifiers. *IEEE-NN*, *11*, 1458.

Girosi, F. (1997). *An equivalence between sparse approximation and support vector machines* (Technical Report AIM-1606).

Greenberg, S., Ellis, D., & Hollenback, J. (1996). Insights into spoken language gleaned from phonetic transcripti on of the Switchboard corpus. *ICSLP96* (pp. S24–27). Philadelphia, PA.

Jaakkola, T. S., & Jordan, M. I. (1996). Computing upper and lower bounds on likelihoods in intractable networks. *In Proceedings of the Twelfth Conference on Uncertainty in AI.*

Kimeldorf, G., & Wahba, G. (1971). A correspondence between Bayesian estimation and on stochastic processes and smoothing by splines. *Annals of Math. Stat.*, *41(2)*, 495–502.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. *Proc. 18th International Conf. on Machine Learning* (pp. 282–289). Morgan Kaufmann, San Francisco, CA.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum Entropy Markov Models for Information Extraction and Segmentation. *Machine Learning: Proceedings of the Seventeenth International Conference (ICML 2000)* (pp. 591–598). Stanford, California.

Minka, T. (2001). *A family of algorithms for approximate Bayesian inference.* PhD thesis, MIT Media Lab.

Opper, M., & Winther, O. (2000). Gaussian Processes for classification: Mean-field algorithms. *Neural Computation*, *12*, 2655–2684.

Punyakanok, V., & Roth, D. (2000). The use of classifiers in sequential inference. *Advances in Neural Information Processing Systems* (pp. 995–1001).

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels.* MIT Press.

Seeger, M., Lawrence, N. D., & Herbrich, R. (2003). Fast sparse Gaussian Process methods: The informative vector machine. *Advances in Neural Information Processing Systems.*

Smola, A. J., & Bartlett, P. L. (2000). Sparse greedy Gaussian Process regression. *Advances in Neural Information Processing Systems* (pp. 619–625).

Smola, A. J., & Schölkopf, B. (2000). Sparse greedy matrix approximation for machine learning. *Proc. 17th International Conf. on Machine Learning* (pp. 911–918). Morgan Kaufmann, San Francisco, CA.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin markov networks. *Advances in Neural Information Processing Systems.*

Weston, J., & Watkins, C. (1999). Support vector machines for multi-class pattern recognition. *Proceedings European Symposium on Artificial Neural Networks.*

Williams, C. K. I., & Barber, D. (1998). Bayesian classification with Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 1342–1351.

Williams, C. K. I., & Seeger, M. (2000). Using the nystrom method to speed up kernel machines. *Advances in Neural Information Processing Systems.*