Department of Computer Science
Brown University
Providence, Rhode Island

Technical Report

CS-03-23

Gaussian Process Classification for Segmenting and Annotating Sequences

by

Yasemin Altun, Thomas Hofmann

December 2003

Gaussian Process Classification for Segmenting and Annotating Sequences

Yasemin Altun
Department of Computer Science
Brown University
altun@cs.brown.edu

Thomas Hofmann
Department of Computer Science
Brown University
th@cs.brown.edu

Abstract

Multiclass classification refers to the problem of assigning 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, using models that take possible interdependencies between label variables into account. This scenario subsumes problems of sequence segmentation and annotation. In this paper, we investigate the use of Gaussian Process (GP) classification for label sequences.

1 Introduction

Multiclass classification refers to the problem of assigning 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, using models that take possible interdependencies between label variables into account. Notice that 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 are Hidden Markov Models (HMMs), which define a generative probabilistic model of labeled observation sequences. HMMs can also be trained discriminatively and in doing so it is possible to drop some of the restrictive conditional independence assumptions of HMMs, for example, by allowing a label to depend on features of neighboring instances in the sequence. The latter was introduced by Lafferty et al. [10] and is often referred to as Conditional Random Fields (CRFs). 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. There have also been attempts to extend other discriminative methods such as AdaBoost [1], perceptron learning [4], and Support Vector Machines (SVM) [2, 17] to the label sequence learning problem. In particular, the so-called Hidden Markov Support Vector Machine [2] (HM-SVM) has experimentally compared favorably to other discriminative methods (including

CRFs) and has the conceptual advantage of being able to work with implicit data representations via kernel functions.

In this paper, we investigate the use of Gaussian Process (GP) classification [20, 7] for label sequences. The main motivation for pursuing this direction is to combine the best of both worlds from CRFs and HM-SVMs. More specifically, we would like to preserve the main strength of CRFs, which we see in the probabilistic interpretation of the outputs: in addition to predicting the best labels, one can compute posterior label probabilities. This is a valuable feature in applications that built on automatically generated annotations. On the other hand, we would also like to make use of kernel functions to construct and learn over Reproducing Kernel Hilbert Spaces (RKHS), thereby overcoming the limitations of (finite-dimensional) parametric statistical models.

A second objective of our work is to get some clarification with respect to two aspects on which CRFs and HM-SVMs 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. While computationally demanding, recent progress on sparse approximation methods for GPs (e.g. [15, 6, 14, 21]) suggest that scalable GP label sequence learning may be an achievable goal. In order to accomplish this, we investigate ways of extending Gaussian Process Classification to label sequence learning problem, by exploiting sparseness not only in terms of training instances, but also in terms of the labels considered for every instance.

The rest of the paper is organized as follows: In Section 2, we will first introduce Gaussian Process Classification and the sparse approximation algorithms that have been proposed previously. Different formulations of Gaussian Process Sequence classification (GPS) are discussed in Section 3. Then, we present an approximate solution for one of these formulations in Section 4 and, finally, report preliminary results using real-world data from Natural Language Processing in Section 5.

2 Gaussian Process Classification and Its Sparse Solutions

In a supervised classification problem, we are given a training set of labeled instances or observations, $\mathbf{Z} \equiv \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$ with $y_i \in \{1, \ldots, m\}$, which are drawn i.i.d. from an unknown, but fixed, distribution $P(\mathbf{x}, y)$. In the Gaussian process approach, we construct a model for the conditional probability distribution $P(y|\mathbf{x})$ based on the assumption that there are independent stochastic processes $u^c(\mathbf{x})$ for each class, $c = 1, \ldots, m$, parameterized by $\mathbf{x}$ such that the random variables $(u^c(\mathbf{x}^1), \ldots, u^c(\mathbf{x}^n))$ are normally distributed with zero mean and covariance matrix $\mathbf{K} = (K_{ij})$, where $K_{ij} = k(\mathbf{x}^i, \mathbf{x}^j)$ and k is a kernel function. Hence each $u^c(\mathbf{x})$ is a Gaussian process [13]. Moreover, one uses a so-called link function to model the dependency between the label variables y and the latent variables $u^c(\mathbf{x})$. In this paper we focus on the soft-max link functions, which generalizes the logistic link function from the binary case, i.e.

$$p(y_i|u_i) = P(Y_i = c|u_i) = \frac{\exp(u_i^c)}{\sum_{c'=1}^{m} \exp(u_i^{c'})} \tag{1}$$

To predict label y of a new test observation $\mathbf{x}$, we use the MAP estimate since the integration over $p(u|\mathbf{X},\mathbf{y})$ is computational intractable:

$$p(y|\mathbf{X},\mathbf{y},\mathbf{x}) = \int p(y|\mathbf{u},\mathbf{x})p(\mathbf{u}|\mathbf{X},\mathbf{y})d\mathbf{u} \approx p(y|\mathbf{u}_{MAP},\mathbf{x}) \tag{2}$$

where $\mathbf{u}_{MAP} = \mathrm{argmax}_{\mathbf{u}} \log p(\mathbf{u}|\mathbf{X},\mathbf{y})$. The posterior of $\mathbf{u}$ is given by:

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

Using the presumed conditional independence assumptions one gets for the negative log-posterior of $\mathbf{u}$:

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

The representer theorem [9] guarantees that the minimizer of (4) can be described by $u^c(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j^c k(\mathbf{x}_j,\mathbf{x})$. Then, the optimization problem can be rewritten in terms of α as follows:

$$R(\alpha|\mathbf{X},\mathbf{y}) = -\sum_{i=1}^{n} \log p(y_i|\mathbf{X},\alpha) + \sum_{c=1}^{m} (\alpha^c)'\mathbf{K}\alpha^c \tag{5}$$

A comparison between (5) and a similar multiclass SVM formulation [18, 5] points out the connection between GP classification and SVMs. Their difference lies primarily in the utilized loss functions: logistic loss vs. hinge loss. Because hinge loss truncates values smaller than ϵ to 0, it enforces sparseness in terms of α parameters. This is not the case for logistic regression as well as other choices of loss functions.[1]

For non-linear link functions, $\mathbf{u}_{MAP}$ cannot be found analytically, but one has resort to approximate solutions. Various approximating schemes have been studied to that extend: Laplace approximation [20, 19, 21], Variational Methods [13], Mean Field Approximations [12], and Expectation Propagation [11, 14]. Here, we briefly describe the Laplace approximation, since it is the computationally simplest one. Laplace's approximation uses a multivariate normal to approximate the posterior in a way that matches the curvature of the Gaussian and the true posterior at the mode [11]. It can be used in conjunction with Newton's method to minimize the negative log posterior distribution.

For binary classification problems, (5) can be rewritten as follows:

$$R(\alpha|\mathbf{X},\mathbf{y}) = \sum_{i=1}^{n} -y_i(K(\mathbf{x}_i,.)\alpha) + \log\left(1 + \exp(K(\mathbf{x}_i,.)\alpha)\right) + \alpha'\mathbf{K}\alpha \tag{6}$$

With Laplace's approximation, the solution to (6) is found iteratively using the update rule given by the Newton-Raphson method:

$$\alpha^{new} = (\mathbf{K}^T W\mathbf{K} + \mathbf{K})^{-1}\mathbf{K}^T W(\mathbf{K}\alpha^{old} + W^{-1}(\mathbf{y} - \mathbf{p})) \tag{7}$$

such that $W = \mathrm{diag}(p_1(1-p_1), \ldots, p_n(1-p_n))$ where p_i is the probability of $\mathbf{x}_i$ labeled with class 1. The update rule can be easily extended to the case of multiclass classification.

[1] Several studies focused on finding sparse solutions to optimization problems similar to (5) ([16, 3, 8, 21]).

Performing Newton-Raphson update steps involves the inversion of an $nm \times nm$ 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 $\mathbf{K}$ by some $\tilde{\mathbf{K}} = \mathbf{U}^T \mathbf{K}_{sub} \mathbf{U}$ (see [13] for references on such methods). This approximation is used for efficient inversion. One can also try to solve (5) using greedy optimization methods as proposed in [21, 3].

3 Gaussian Process Sequence Classification (GPSC)

In sequence classification, our goal is to learn a discriminant function for sequences, i.e. a mapping from observation sequences $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^t, \dots)$ to label sequences $\mathbf{y} = (y^1, y^2, \dots, y^t, \dots)$ where $y^t \in \mathcal{Y} = \{1, \dots, m\}$. y^t depends on the observation sequence $\mathbf{X}$ and the neighboring y^s values. The availability of a training set of labeled sequences $\mathcal{Z} \equiv \{(\mathbf{X}_i, \mathbf{y}_i) : i = 1, \dots, n\}$ to learn this mapping from data is assumed. The conditional probability distribution over label sequences $\mathbf{y}$ for every fixed observation sequence $\mathbf{X}$ is given by

$$P_\theta(\mathbf{y}|\mathbf{X}) = \exp(\langle \Phi(\mathbf{X}, \mathbf{y}), \theta \rangle - g(\theta))$$

where $g(\theta) = \log \sum_{\mathbf{y}} \exp(\langle \Phi(\mathbf{X}, \mathbf{y}), \theta \rangle)$ is the log-partition function.

We treat the sequence labeling problem as a multiclass classification problem, where the macro-label set $\bar{\mathcal{Y}}$ consists of the set of all possible micro-label sequence of a given length l. Obviously, the size of this label set M, is exponentially large in l, $M = m^l$. Notice that the latter formulation is also the one used in CRFs and HM-SVMs. We introduce a latent process $\mathbf{u} \in \Re^M$ such that given $\mathbf{u}(\mathbf{X})$, $\mathbf{X}$ and $\mathbf{y}$ are independent. The random variables $(\mathbf{u}(\mathbf{X}_1), \dots, \mathbf{u}(\mathbf{X}_n))$ (where $\mathbf{u}(\mathbf{X}_j) = (u_j^1, \dots, u_j^M)$, $u_j^s = u(\mathbf{X}_j, \mathbf{y})$ is the random variable associated with the relation of s-th **label sequence** candidate $\mathbf{y}$ and the j-th **observation sequence**) are normally distributed with zero mean and covariance matrix $\mathbf{K}$, $K_{ij}(\mathbf{y}, \bar{\mathbf{y}}) = k((\mathbf{X}_i, \mathbf{y}), (\mathbf{X}_j, \bar{\mathbf{y}}))$, where k now is a joint kernel function defined over *labeled* observations sequences. We later elaborate the joint kernel function.

As described in Section 2, the problem of computing the MAP estimate of $\mathbf{u}$ is equivalent to solving the optimization problem:

$$R(\mathbf{u}|\mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} - \left[u(\mathbf{X}_i, \mathbf{y}_i) - \log \sum_{\mathbf{y} \in \bar{\mathcal{Y}}} \exp(u(\mathbf{X}_i, \mathbf{y})) \right] + \frac{1}{2} \mathbf{u}^T \mathbf{K}^{-1} \mathbf{u} \tag{8}$$

Using the definition of $u(\mathbf{X}_i, \mathbf{y}) = \sum_j \sum_{\bar{\mathbf{y}} \in \bar{\mathcal{Y}}} \alpha_j(\bar{\mathbf{y}}) K_{ij}(\mathbf{y}, \bar{\mathbf{y}})$, the loss function can equivalently be stated in terms of α:

$$\begin{aligned} R(\alpha|\mathbf{X}, \mathbf{y}) = & \sum_{i=1}^{n} \left[- \sum_{j, \bar{\mathbf{y}}} \alpha_j(\bar{\mathbf{y}}) K_{ij}(\mathbf{y}_i, \bar{\mathbf{y}}) + \log \left(\sum_{\mathbf{y} \in \bar{\mathcal{Y}}} \exp(\sum_{j, \bar{\mathbf{y}}} \alpha_j(\bar{\mathbf{y}}) K_{ij}(\mathbf{y}, \bar{\mathbf{y}})) \right) \right] \\ & + \sum_{j, \bar{\mathbf{y}}} \sum_{i, \mathbf{y}} \alpha_j(\bar{\mathbf{y}}) \alpha_i(\mathbf{y}) K_{i,j}(\mathbf{y}, \bar{\mathbf{y}}) \end{aligned} \tag{9}$$

The set of macro-labels $\bar{\mathcal{Y}}$ is exponentially large in the length of the sequence and the exact optimization becomes intractable as the length of the sequences get larger. However, we have not yet taken into consideration that each label y^t depends only on the neighboring labels. We can decompose the above formulation using the sparseness of our graph.

First lets be more precise about the kernel function k:

$$K((\mathbf{X}, \mathbf{y}), (\bar{\mathbf{X}}, \bar{\mathbf{y}})) = \langle \Phi(\mathbf{X}, \mathbf{y}), \Phi(\bar{\mathbf{X}}, \bar{\mathbf{y}}) \rangle$$

Motivated from HMMs, $\Phi(\mathbf{X}, \mathbf{y})$ consists of observation-label features, $\Lambda(y_t) \otimes \Psi(\mathbf{x}_t)$, and label-label features $\Lambda(y_t) \otimes \Lambda(y_{t+1})$ where $\otimes$ is the tensor product operator. The entries of covariance matrix $\mathbf{K}$ can further be refined as:

$$\begin{aligned} K_{ij}(\mathbf{y}, \bar{\mathbf{y}}) &= \sum_{s,t=1}^{l} \langle \Lambda(y_t), \Lambda(\bar{y}_s) \rangle k(\mathbf{x}_t^i, \bar{\mathbf{x}}_s^j) \\ &+ \sum_{s,t=1}^{l-1} \langle \Lambda(y_t), \Lambda(\bar{y}_s) \rangle \langle \Lambda(y_{t+1}), \Lambda(\bar{y}_{s+1}) \rangle \end{aligned} \tag{10}$$

Now lets revisit the definition $u(\mathbf{X}_i, \mathbf{y})$:

$$\begin{aligned} u(\mathbf{X}_i, \mathbf{y}) &= \sum_i \sum_{\bar{\mathbf{y}} \in \mathcal{Y}} \alpha_j(\bar{\mathbf{y}}) K_{ij}(\mathbf{y}, \bar{\mathbf{y}}) \\ &= \sum_{s,\sigma,\tau} [\![y^s = \sigma \wedge y^{s-1} = \tau]\!] \sum_{j,t,\bar{\mathbf{y}} \in \mathcal{Y}} [\![\bar{y}^t = \sigma \wedge \bar{y}^{t-1} = \tau]\!] \alpha_j(\bar{\mathbf{y}}) \\ &+ \sum_{s,\sigma} [\![y^s = \sigma]\!] \sum_{j,t,\bar{\mathbf{y}} \in \mathcal{Y}} [\![\bar{y}^t = \sigma]\!] \alpha_j(\bar{\mathbf{y}}) k(\mathbf{x}_i^s, \mathbf{x}_j^t) \end{aligned} \tag{11}$$

Inspired from [17], we define

$$\alpha_j^t(\sigma, \tau) = \sum_{\bar{\mathbf{y}} \in \mathcal{Y}} \alpha_j(\bar{\mathbf{y}}) [\![\bar{y}^t = \sigma, \bar{y}^{t-1} = \tau]\!] \tag{12}$$

Also define

$$K_{is,jt}((\sigma, \tau), (\bar{\sigma}, \bar{\tau})) = [\![\sigma = \bar{\sigma}]\!] k(\mathbf{x}_i^s, \mathbf{x}_j^t) + [\![\sigma = \bar{\sigma} \wedge \tau = \bar{\tau}]\!] \tag{13}$$

Then $u(\mathbf{X}_i, \mathbf{y})$ can be rewritten as:

$$\begin{aligned} u(\mathbf{X}_i, \mathbf{y}) &= \sum_s \sum_{j,t,(\sigma,\tau)} \alpha_j^t(\sigma, \tau) \left([\![y^s = \sigma \wedge y^{s-1} = \tau]\!] + [\![y^s = \sigma]\!] k(\mathbf{x}_i^s, \mathbf{x}_j^t) \right) \\ &= \sum_s \sum_{j,t,(\sigma,\tau)} \alpha_j^t(\sigma, \tau) K_{is,jt}((y^s, y^{s-1}), (\sigma, \tau)) \end{aligned} \tag{14}$$

It is interesting to note that using the decomposition is equivalent to the following formulation: we introduce a latent variable $\mathbf{u} \in \mathcal{R}^{m \times m}$. $u(\mathbf{x}_t)$ defines a probability distribution for (y_t, y_{t-1}) label pairs. The label y_t thus depends on $u(\mathbf{x}_r)$ and the previous label y_{t-1} For the sake of notational simplicity, assuming that the observation sequences are of a fixed length l for notational convenience. The random variables $(\mathbf{u}(\mathbf{x}_{11}), \ldots, \mathbf{u}(\mathbf{x}_{1l}), \ldots, \mathbf{u}(\mathbf{x}_{n1}), \ldots, \mathbf{u}(\mathbf{x}_{nl}))$ (where $\mathbf{u}(\mathbf{x}_{it}) = (u(\mathbf{x}_{it})^1, \ldots, u(\mathbf{x}_{it})_{r \cdot r})$, $u(\mathbf{x}_{jt}^s$ is the random variable associated with the relation of $(\sigma, \tau) = s$-th **label pair** with $\mathbf{x}_{it}$-th **observation**) are normally distributed with zero mean and covariance $\mathbf{K}$ as defined in Equation 13.

4 Sparse Approximation Algorithm

4.1 Method 1

The storage of **K** matrix in Equation 10, let alone calculating its inverse to be used in a Newton-Raphson update is not possible for any reasonable data set.

As in the case of standard Gaussian Process Classification discussed in Section 2, we want to achieve sparseness in terms of the α parameters. This means that we want sparse solutions not only in terms of data points, but also in terms of the possible labels each data point can be labeled with, i.e. we enforce the probability distribution $P(\mathbf{Y}|\mathbf{X})$ to be skewed where only a small number of labels are possible for each data point.

One way is to use Newton-Raphson method iteratively as described by [21] in the import vector machine algorithm. However, adding a ($\mathbf{X}$,$\mathbf{y}$) pair into the ensemble involves inversion of the incremental **K** iteratively, which may still be very expensive. Alternatively, one can use a steepest gradient descent direction for column generation. We can fix the training example j that you want to include in the ensemble and find the labeling $\mathbf{Y}$ that maximizes the gradient of R. Before explaining how to calculate these values, let us first consider the optimal decoding with respect to parameters α.

The score of a labeling of an observation sequence $\mathbf{X}$ is given by

$$u(\mathbf{X},\mathbf{y}) = \sum_{i=1}^{n}\sum_{\bar{\mathbf{y}}} \alpha_i(\bar{\mathbf{y}}) k_y(\mathbf{y},\bar{\mathbf{y}}) k_x(\mathbf{X},\mathbf{X}_i) \, .$$

The score u can be decomposed into two contributions (u_1 and u_2) due to the additivity of the sequence feature map.

$$u_1(\mathbf{X},\mathbf{y}) = \sum_{\sigma,\tau} \eta(\sigma,\tau) \sum_s [\![y_{s-1}{=}\sigma \wedge y_s{=}\tau]\!] \, , \tag{15a}$$

$$\eta(\sigma,\tau) = \sum_{i,\bar{\mathbf{y}}} \alpha_i(\bar{\mathbf{y}}) \sum_t [\![\bar{y}_{t-1} = \sigma \wedge \bar{y}_t = \tau]\!] \tag{15b}$$

and where

$$u_2(\mathbf{X},\mathbf{y}) = \sum_{s,\sigma} [\![\mathbf{y}_s {=} \sigma]\!] \sum_{i,t} \beta(i,t,\sigma) k(\mathbf{x}_s,\mathbf{x}_{it}), \tag{16a}$$

$$\beta(i,t,\sigma) = \sum_{\mathbf{y}} [\![y_t = \sigma]\!] \, \alpha_i(\mathbf{y}) \, . \tag{16b}$$

The best label of an observation is find by performing a Viterbi decoding on observation cost matrix $\sum_{i,t} \beta(i,t,\sigma) k(\mathbf{x}_s,\mathbf{x}_{it})$ and label cost matrix η.

In order to find the steepest descent direction, we need to calculate the gradient:

$$\nabla_{\alpha_j(\mathbf{y})} R = -\sum_{i}^{N} K_{ji}(\mathbf{y},\mathbf{y}_i) + \sum_{i=1}^{n} E_{\bar{\mathbf{y}}}[K_{ji}(\mathbf{y},\bar{\mathbf{y}})] + \gamma \sum_{k,\mathbf{y}'} \alpha_k(\mathbf{y}') K_{kj}(\mathbf{y}',\mathbf{y}) \tag{17}$$

where the expectation is over all possible labels of the training example i. Again using the additivity of the feature map:

$$\begin{aligned}\sum_i^N E_{\bar{\mathbf{y}}}[K_{ji}(y,\bar{y})] &= \sum_i E_{\bar{\mathbf{y}}}\left[\sum_{s,t}[\![y_{s-1}=\bar{y}_{t-1}\wedge y_s=\bar{y}_t]\!]\right] \\ &+ \sum_i E_{\bar{\mathbf{y}}}\left[\sum_{s,t}[\![y_s=\bar{y}_t]\!]k_x(\mathbf{x}_{js},\mathbf{x}_{it})\right] \\ &= \sum_{\sigma,\tau}(\sum_s[\![y_{s-1}=\sigma\wedge y_s=\tau]\!])\sum_i\sum_t E_i[\bar{y}_{t-1}=\sigma\wedge\bar{y}_t=\tau] \\ &+ \sum_{s,\sigma}[\![y_s=\sigma]\!]\sum_i\sum_t k(\mathbf{x}_{js},\mathbf{x}_{it})E_i[\bar{y}_t=\sigma] \qquad (18)\end{aligned}$$

The labeling $\mathbf{y}$ that maximizes $\nabla_{\alpha_j(\mathbf{y})}R$ can be found by using the Viterbi decoding again. Since the bottleneck of this algorithm is the calculation of the expectations, after this calculation, we choose r pairs of $(\mathbf{X},\mathbf{y})$ to include in the ensemble and then use Conjugate Gradient Descent method to find the optimal α for the current iteration.

4.2 Method 2

The score of a labeling of an observation sequence $\mathbf{X}$ is given by

$$u(\mathbf{X}_i,\mathbf{y}) = \sum_s\sum_{j,t,(\sigma,\tau)}\alpha_j^t(\sigma,\tau)K_{is,jt}((y^s,y^{s-1}),(\sigma,\tau))$$

The score u can be decomposed into two contributions (u_1 and u_2) due to the additivity of the sequence feature map.

$$u_1(\mathbf{X},\mathbf{y}) = \sum_{\sigma,\tau}\eta(\sigma,\tau)\sum_s[\![y_{s-1}{=}\sigma\wedge y_s{=}\tau]\!]\,, \qquad (19a)$$

$$\eta(\sigma,\tau) = \sum_{j,t}\alpha_j^t(\sigma,\tau) \qquad (19b)$$

and where

$$u_2(\mathbf{X},\mathbf{y}) = \sum_{s,\sigma}[\![\mathbf{y}_s{=}\sigma]\!]\sum_{i,t}\beta(i,t,\sigma)k(\mathbf{x}_s,\mathbf{x}_{it}), \qquad (20a)$$

$$\beta(j,t,\sigma) = \sum_\tau\alpha_j^t(\sigma,\tau)\,. \qquad (20b)$$

The best label of an observation is find by performing a Viterbi decoding on observation cost matrix $\sum_{i,t}\beta(i,t,\sigma)k(\mathbf{x}_s,\mathbf{x}_{it})$ and label cost matrix η.

The loss function can be rewritten in terms of cumulative α parameters as follows:

$$\begin{aligned}R(\alpha|\mathbf{X},\mathbf{y}) &= \sum_{i=1}^n\left[-u(\mathbf{X}_i,\mathbf{y}_i)+\log(\sum_{\mathbf{y}}\exp(u(\mathbf{X}_i,\mathbf{y}))\right] \\ &+ \sum_{j,t,(\sigma,\tau)}\sum_{i,s,(\bar{\sigma},\bar{\tau})}\alpha_j^t(\sigma,\tau)\alpha_i^s(\bar{\sigma},\bar{\tau})K_{is,jt}((\bar{\sigma},\bar{\tau}),(\sigma,\tau)) \qquad (21)\end{aligned}$$

In order to find the steepest descent direction, we need to calculate the gradient:

$$\begin{aligned}\nabla_{\alpha_j^t(\sigma,\tau)}R &= \sum_i^N\sum_s\left(-K_{is,jt}((y_i^s,y_i^{s-1}),(\sigma,\tau))+E_{\bar{\mathbf{y}}}[K_{is,jt}((y_i^s,y_i^{s-1}),(\sigma,\tau))]\right) \\ &+ \gamma\sum_{i,s,(\bar{\sigma},\bar{\tau})}\alpha_s^i(\bar{\sigma},\bar{\tau})K_{is,jt}((\bar{\sigma},\bar{\tau}),(\sigma,\tau)) \qquad (22)\end{aligned}$$

where the expectation is over all possible labels of the training example i. Again using the additivity of the feature map:

$$\sum_i^N \sum_s E_{\bar{\mathbf{y}}}[K_{is,jt}((y_i^s, y_i^{s-1}), (\sigma, \tau))] = \sum_{i,s} E_{\bar{\mathbf{y}}}\left[[\![y^{s-1} = \sigma \wedge y^s = \tau]\!]\right] + \sum_{i,s} E_{\bar{\mathbf{y}}}\left[[\![y^s = \sigma]\!]\right] k(\mathbf{x}_i^s, \mathbf{x}_j^t) \quad (23)$$

The above expectations can be calculated with a forward-backward pass over the training data. The number of α parameters is $i * l * n * n$. One can use conjugate gradient descent method for this optimization problem.

To apply Newtons method, one need to calculate the second derivatives. However, that involves finding the expectation of $[\![y^s = \sigma \wedge y^t = \tau]\!]$ over all possible labeling of example $\mathbf{X}_i$, which cannot be calculated efficiently.

5 Experiments

We used the algorithm described above for some preliminary experiment on Named Entity Recognition task.

Named Entity Recognition (NER) is an information extraction problem which deals with finding phrases containing person, location and organization names, as well as temporal and number expressions. Each entry is annotated with the *type* of its expression and its *position* in the expression, i.e. the beginning or the continuation of the expression.

We generated a sub-corpus consisting of 100 sentences from the Spanish news wire article corpus which was provided for the Special Session of CoNLL2002 on NER. The expression types in this corpus are limited to person names, organizations, locations and miscellaneous names, resulting in a total of $|\Sigma| = 9$ different labels.

All input features are simple binary features. Most features are indicator functions for a word occurring within a fixed size window centered on the word being labeled. In addition there are features that do not only encode the identity of the word, but also more detailed properties (e.g. spelling features). Notice that these features are combined with particular label indicator functions in the joint feature map framework. Some example features are: *Is the two previous word = 'Mr.' and the current tag='Person-Beginning'?*, *Does the next word end with a dot and the current tag='Non-name'?*, and *Is the previous tag='Non-name' and current tag='Location-Intermediate?'*.

The Gaussian Process model optimized as described above achieved an accuracy of 91.4% whereas HM-SVMs achieved 91.9%. The nature of the extracted support sequences chosen by the GPS classification are very different than the ones chosen by HM-SVMs. In HM-SMVs, most of the support sequences only differ in a few positions from the correct label sequence, resulting in sparse solutions, whereas in GPS classification the support sequences are different from the correct label sequence in many positions. The number of the support sequences are comparable. There are on average 30 support sequences per sequence, which shows the sparseness of the problem.

We aim to perform experiments with more data and on different applications, comparing GPS, HM-SVMs and CRFs.

References

[1] Y. Altun, T. Hofmann, and M. Johnson. Discriminative learning for label sequences via boosting. In *Advances in Neural Information Processing Systems (NIPS*15)*, 2003.

[2] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hiddem markov support vector machines. In *20th International Conference on Machine Learning (ICML)*, 2003.

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

[4] M. Collins. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

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

[6] L. Csat'o and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14:641–668, 2002.

[7] M. N. Gibbs and D. J. C. MacKay. Variational gaussian process classifiers. *IEEE-NN*, 11(6):1458, November 2000.

[8] Federico Girosi. An equivalence between sparse approximation and support vector machines. Technical Report AIM-1606, 1997.

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

[10] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[11] T. Minka. A family of algorithms for approximate bayesian inference, 2001.

[12] Manfred Opper and Ole Winther. Gaussian processes for classification: Mean-field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

[13] B. Scholkopf and A. J. Smola. *Learning with Kernels.* MIT Press, 2002.

[14] Matthias Seeger, Neil D. Lawrence, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *in Advances in Neural Information Processing Systems, Workshop on Kernel Methods*, 2003.

[15] Alex J. Smola and Peter L. Bartlett. Sparse greedy gaussian process regression. In *NIPS*, pages 619–625, 2000.

[16] Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proc. 17th International Conf. on Machine Learning*, pages 911–918. Morgan Kaufmann, San Francisco, CA, 2000.

[17] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *to appear in Advances in Neural Information Processing Systems, Workshop on Kernel Methods*, 2004.

[18] J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings European Symposium on Artificial Neural Networks*, 1999.

[19] C. K. I. Williams and M. Seeger. Using the nystrom method to speed up kernel machines. In *NIPS*, 2000.

[20] Christopher K. I. Williams and David Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[21] Zhu and Trevor Hastie. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.