Kernel Methods for Missing Variables

Alex J. Smola, S.V.N. Vishwanathan
Statistical Machine Learning Program
NICTA and ANU, Canberra, ACT, 0200
{Alex.Smola, SVN.Vishwanathan}@nicta.com.au

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

Abstract

We present methods for dealing with missing variables in the context of Gaussian Processes and Support Vector Machines. This solves an important problem which has largely been ignored by kernel methods: How to systematically deal with incomplete data? Our method can also be applied to problems with partially observed labels as well as to the transductive setting where we view the labels as missing data.

Our approach relies on casting kernel methods as an estimation problem in exponential families. Hence, estimation with missing variables becomes a problem of computing marginal distributions, and finding efficient optimization methods. To that extent we propose an optimization scheme which extends the Concave Convex Procedure (CCP) of Yuille and Rangarajan, and present a simplified and intuitive proof of its convergence. We show how our algorithm can be specialized to various cases in order to efficiently solve the optimization problems that arise. Encouraging preliminary experimental results on the USPS dataset are also presented.

1 Introduction

Kernel methods [12] have been remarkably successful for standard classification and regression problems. However, they have also been found very effective in dealing with a variety of related learning problems such as sequence annotation, conditional random fields, multi-instance learning, and novelty detection. Many algorithms for Gaussian Processes (GP) and Support Vector Machines (SVM) bear witness of this. One problem, however, has remained completely untouched so far: How to deal with datasets which exhibit missing variables?

In the following, we will develop a framework to deal with such cases in a systematic fashion. Our analysis is based on the observation that kernel methods can be written as estimators in an exponential family. More specifically, Gaussian Processes can be seen to be maximizing the negative log-posterior under a normal prior on the natural parameter of the exponential density, whereas Support Vector Machines maximize the likelihood ratio. Based on this observation, we provide a method for dealing with missing variables in such a way that standard kernel methods arise as a special case, whenever there are no missing variables.

To solve the optimization problems arising in this context – a concave-convex objective function with both convex and concave constraints – we extend the CCP algorithm of [16] for finding local optima and give an intuitive proof for its convergence.

The rest of the paper is organized as follows. In Section 2.1 we discuss exponential families in feature space and in Sections 2.2 -2.4 we present methods to deal with missing data. In Section 2.5 we show how Gaussian Processes and Support Vector Machines can be extended to deal with missing data. Section 3 is devoted to the discussion of the Constrained Concave Convex Procedure (CCCP) and its application to Gaussian Processes and Support Vector Machines. We discuss some implementation tips in Section 4 and present experimental results on the USPS dataset in Section 5. An outlook and a discussion in Section 6 conclude the paper.

2 The Model for Incomplete Data

2.1 Exponential Families

We begin with a definition of exponential families: Denote by $\mathcal{X}$ the domain of observations, and let $\phi(x)$ with $x \in \mathcal{X}$ refer to a vector of sufficient statistics.

Then, a member of the exponential family of densities can be defined in exponential normal form via

$$p(x;\theta) = p_0(x)\exp\left(\langle\phi(x),\theta\rangle - g(\theta)\right), \tag{1}$$

where

$$g(\theta) = \log\int_{\mathcal{X}} p_0(x)\exp(\langle\phi(x),\theta\rangle)\,dx. \tag{2}$$

Here, $p_0(x)$ is a suitably chosen underlying measure, θ is the natural parameter, $g(\theta)$ is the log-partition function, often called the cumulant generating function, and $\langle\cdot,\cdot\rangle$ denotes a scalar product in an Euclidean space, or more generally in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. Without loss of generality, and for ease of exposition, we will ignore the underlying measure $p_0(x)$ for the rest of the paper.

Let $\mathcal{Y}$ denote the space of labels, and $\phi(x,y)$ be the sufficient statistics of the joint distribution associated with $(x,y)\in\mathcal{X}\times\mathcal{Y}$. For the purpose of classification we are mainly concerned with estimating conditional probabilities. Therefore, we extend the exponential families framework to conditional probabilities. Here we have

$$p(y|x;\theta) = \exp\left(\langle\phi(x,y),\theta\rangle - g(\theta|x)\right), \tag{3}$$

and

$$g(\theta|x) := \log\int_{\mathcal{Y}} \exp(\langle\phi(x,y),\theta\rangle)\,dy. \tag{4}$$

In analogy to the above case, $g(\theta|x)$ is commonly referred to as the conditional log-partition function.

Both $g(\theta)$ and $g(\theta|x)$ are convex C^∞ functions in θ and they can be used to compute cumulants of the distribution [6, 4], for instance:

$$\begin{aligned}
\partial_\theta g(\theta) &= \mathbb{E}_{p(x;\theta)}[\phi(x)],\\
\partial_\theta^2 g(\theta) &= \mathrm{Var}_{p(x;\theta)}[\phi(x)],\\
\partial_\theta g(\theta|x) &= \mathbb{E}_{p(x,y;\theta)}[\phi(x,y)|x],\\
\partial_\theta^2 g(\theta|x) &= \mathrm{Var}_{p(x,y;\theta)}[\phi(x,y)|x].
\end{aligned}$$

2.2 Incomplete Training Data

In the following, we will deal with the problem of estimating $p(y|x;\theta)$ or a related quantity based on a set of observations $(x_i,y_i)\in\mathcal{X}\times\mathcal{Y}$ with $i=1,\ldots,m$. More specifically, we allow that some of the x_i have been observed only partially, that is, we may partition the observations as $x_i=(x_i^o,x_i^u)$, where x_i^o represents the observed part and x_i^u is the unobserved part of the data (see [3] for a detailed description of how missing data may arise and how it is typically treated in an Expectation Maximization (EM) context). Observe that we allow for different sets of missing variables for different data points.

The first step is to extend (3) to partially observed data. Clearly

$$p(x^u,y|x^o;\theta) = \exp\left(\langle\phi(x^o,x^u,y),\theta\rangle - g(\theta|x^o)\right). \tag{5}$$

Integration over the unobserved part of x, that is, x^u, and direct calculation yields

$$\begin{aligned}
p(y|x^o;\theta) &= \int_{\mathcal{X}^u}\exp\left(\langle\phi(x^o,x^u,y),\theta\rangle - g(\theta|x^o)\right)dx^u\\
&= \exp(g(\theta|x^o,y) - g(\theta|x^o)),
\end{aligned} \tag{6}$$

with a suitable definition of $g(\theta|x^o,y)$. In other words, the conditional probability $p(y|x^o;\theta)$ is now given by the exponential of the difference of two conditional log-partition functions. This poses two problems:

- Computing the log-partition function is a nontrivial problem [14]. In particular, the computation of $g(\theta|x^o)$ and $g(\theta|x^o,y)$ may pose additional difficulties. This is because the joint sufficient statistics might lead to an intractable integral. However, in many real life applications the data is discrete and only a small number of variables are missing. In these cases, one can either resort to brute force computation or exploit the algebraic structure of the integrand.
- The negative log-likelihood, $-\log(p(y|x^o;\theta))$, ceases to be a convex function. This means that the optimization problems arising from estimation with missing variables may involve many local optima. In Section 3, we will present an optimization method to deal with this problem by extending the CCP of [16] as well as a second method based on the EM algorithm.

2.3 Incomplete Labels

Using ideas similar to those used for handling missing training data we can also handle data with missing labels. As before, we partition $y_i=(y_i^o,y_i^u)$ and integrate out the unobserved part of the labels to yield

$$\begin{aligned}
p(y^o|x;\theta) &= \int_{\mathcal{Y}^u}\exp\left(\langle\phi(x,y^o,y^u),\theta\rangle - g(\theta|x)\right)dy^u\\
&= \exp(g(\theta|x,y^o) - g(\theta|x)).
\end{aligned} \tag{7}$$

(7) can then be used to perform Maximum Likelihood Estimation (MLE) or Maximum A Posteriori (MAP) estimation. As before, the conditional probability $p(y^o|x;\theta)$ is given by the exponential of the difference of two conditional log-partition functions. Note that both types of missing data can also be combined in a straightforward manner using the conditional density $p(y^o|x^o;\theta)$.

2.4 Transduction

Transduction can be viewed as an extreme case of incomplete labels. Typically, we are given a set of observations with a few missing labels. The task is to predict these missing labels. We now compute a probability distribution on the missing labels, given by

$$p(y^u|x, y^o; \theta) = \exp\left(\langle \phi(x, y^o, y^u), \theta\rangle - g(\theta|x, y^o)\right).$$

In order to compute the above density we need to compute

$$g(\theta|x, y^o) = \log \int_{\mathcal{Y}^u} \exp(\langle \phi(x, y^o, y^u), \theta\rangle \; dy^u. \quad (8)$$

If the space of labels $\mathcal{Y}$ is large, and many labels are missing, then computing the above integral is a nontrivial task and we need to resort to Monte-Carlo sampling methods or other similar high dimensional integration techniques in order to perform prediction.

2.5 Conditional Probabilities and Estimators

Our discussion so far has been very generic. In this section, we focus on two particular kernel algorithms. First, we show how Gaussian Processes can be viewed as estimators in exponential families. Then, we discuss the well known Support Vector Machines in the context of exponential families. Using our discussion above, we also show how both these algorithms can handle missing data in a natural way.

Gaussian Process Classification: If the training data is assumed to be generated IID from an exponential family distribution, the MLE problem for exponential families is to minimize

$$\begin{aligned} -\log p(\theta|X, Y) &= \sum_{i=1}^{m} -\log p(y_i|x_i, \theta) \\ &= \sum_{i=1}^{m} g(\theta|x_i) - \langle \phi(x_i, y_i), \theta\rangle. \end{aligned}$$

Since we are considering an exponential family in feature space, the sufficient statistics are possibly infinite dimensional. To avoid over-fitting the data we consider a prior over the parameter θ.

One can show [1] that Gaussian Processes can be seen as estimators, where the prior on the natural parameter is normal, that is,

$$p(\theta) \propto \exp\left(-\frac{1}{2\sigma^2}\|\theta\|^2\right).$$

To see this, observe that under the above prior $t(x, y) := \langle \phi(x, y), \theta\rangle$ is a Gaussian Process. This is because θ is normally distributed with zero mean, and $\mathbb{E}_\theta[t(x, y)] = 0$ and the covariance (kernel) matrix is given by

$$k((x, y), (x', y')) = \langle \phi(x, y), \phi(x', y')\rangle.$$

This argument is similar to the one used by [15] to establish a connection between Support Vector Machines and Gaussian Processes.

As a special case we let $\mathcal{Y} = \{\pm 1\}$ and consider the choice $\phi(x, y) = y\phi'(x)$. This gives us $k((x, y), (x', y')) = y_i y_j \cdot k'(x, x')$ where $k'(x, x') = \langle \phi'(x), \phi'(x')\rangle$.

Now using the normal prior, the MAP estimation problem for exponential families is to minimize

$$\begin{aligned} -\log p(\theta|X, Y) &= \sum_{i=1}^{m} -\log p(y_i|x_i, \theta) + \frac{\|\theta\|^2}{2\sigma^2} \quad (9) \\ &= \sum_{i=1}^{m} g(\theta|x_i) - \langle \phi(x_i, y_i), \theta\rangle + \frac{\|\theta\|^2}{2\sigma^2}. \end{aligned}$$

Observe that the MAP estimation problem (9) is convex, and by the representer theorem [11], the minimizer θ^* can be found in the span of $\{\phi(x_i, y)$ where $y \in \mathcal{Y}\}$. So far, this interpretation of Gaussian Processes is consistent with the *classical* viewpoint.

We now turn to the setting with incomplete input data (the setting with missing labels is analogous and can be handled similarly). Here, all we need to do is to replace $p(y_i|x_i, \theta)$ by $p(y_i|x_i^o, \theta)$. Using (6) this leads to the following problem:

$$\text{minimize} \sum_{i=1}^{m} [g(\theta|x_i^o) - g(\theta|x_i^o, y_i)] + \frac{1}{2\sigma^2}\|\theta\|^2. \quad (10)$$

Unlike (9), the above problem is no longer convex, and we will need a more sophisticated method to solve it. In Section 3 we show how the CCCP method can be used to solve this optimization problem efficiently.

It is easy to check that $g(\theta|x_i^o, y_i) = \langle \phi(x_i^o, y_i), \theta\rangle$ if $x_i^o = x_i$, that is, we recover the original Gaussian Process optimization problem whenever the set of observations is complete.

Support Vector Classification: Gaussian Processes maximize the log-likelihood using a normal prior on the parameters. Instead of directly maximizing the log-likelihood, one may want to maximize the log-likelihood ratio between the correct label and the most likely incorrect labeling [9]. This leads to the following cost function:

$$r(x, y; \theta) := \log \frac{p(y|x;\theta)}{\max_{\tilde{y}\neq y} p(\tilde{y}|x;\theta)} \tag{11}$$

$$= \langle \phi(x, y), \theta \rangle - \max_{\tilde{y}\neq y} \langle \phi(x, \tilde{y}), \theta \rangle . \tag{12}$$

In order to take the margin into account, we use

$$c(x, y; \theta) := \max(1 - r(x_i, y_i; \theta), 0)$$

which is essentially a clipped version of $r(x, y; \theta)$.

To see the connection to binary Support Vector Machines, assume $\mathcal{Y} \in \{\pm 1\}$ and $\phi(x, y) = \frac{y}{2}\phi'(x)$. Then, $r(x, y; \theta) = y_i \langle \phi'(x), \phi'(x') \rangle$ and $c(x, y; \theta) = \max(1 - y_i \langle \phi'(x), \phi(x') \rangle , 0)$ which essentially recovers the hinge loss. Therefore, our loss function is simply a generalization of the hinge loss to multi-class Support Vector Machines [9].

In fact, the MAP estimate in this case is found by solving

$$\underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{m} c(x_i, y_i; \theta) + \frac{1}{2\sigma^2}\|\theta\|^2. \tag{13}$$

To recover soft margin estimates, one simply needs to introduce slack variables into the above equation.

The main difference between the Support Vector Machine and the Gaussian process optimization problem is that, in the case of Support Vector Machines, the cost function $c(x, y; \theta)$ does *not* depend on the log-partition function. Instead, it is given by the difference between scalar products.

An extension to missing variables is now straightforward: all we need to do is to replace the conditional probability estimates in the fully observed case by their counterparts for partially observed data. Using (6) and (11) we have

$$r(x, y; \theta) = g(\theta|x^o, y) - \max_{\tilde{y}\neq y} g(\theta|x^o, \tilde{y}).$$

Finally, we can introduce slack variables and extend (13) into a constrained optimization problem for missing variables:

$$\text{minimize} \frac{1}{2\sigma^2}\|\theta\|^2 + \sum_{i=1}^{m} \xi_i \tag{14a}$$

$$\text{s.t. } g(\theta|x_i^o, y_i) - \max_{\tilde{y}\neq y_i} g(\theta|x_i^o, \tilde{y}) \geq 1 - \xi_i \tag{14b}$$

$$\xi_i \geq 0. \tag{14c}$$

The difference between (14) and (13) is that now the constraints, as specified by (14b), are non longer convex. Therefore, the minimization is no longer a convex problem, and we need, for instance, an iterative scheme to enforce these constraints.

As before, if $x_i^o = x_i$, that is, if no data is missing, we have $g(\theta|x_i^o, y_i) = \langle \phi(x_i^o, y_i), \theta \rangle$ and (14) reduces to a version of (13) which incorporates slack variables.

3 Optimization

As stated in Section 2.5, the optimization problems that arise when data is missing are no longer convex. Hence, it is a non-trivial task to solve them. In the case of Gaussian Processes one could invoke an EM like algorithm to perform maximum likelihood estimation over the joint set of parameters $(\theta, \{x_1^u, \ldots, x_m^u\})$ directly. But, it is not clear how such an algorithm can be extended to incorporate non-convex constraints which arise in the case of Support Vector Machines with missing variables.

Instead, we take a small detour: EM can also be viewed as a consequence of the CCP [16]. This provides us with a strategy to use similar algorithms for constrained problems by extending CCP to the Constrained CCP.

3.1 The Constrained Concave Convex Procedure

Theorem 1 (Constrained CCP) *Denote by f_i, g_j real-valued convex and differentiable functions on a vector space $\mathcal{X}$ for all $i \in \{0, \ldots, n\}$, and let $c_i \in \mathbb{R}$ for $i \in \{1, \ldots, n\}$. Then, Algorithm 1 converges to a local minimum of the following optimization problem, provided that the linearization of the nonconvex constraints in conjunction with the convex constraints satisfy suitable constraint qualifications at the point of convergence of the algorithm.*

$$\text{minimize} f_0(x) - g_0(x) \tag{15a}$$

$$\textit{s.t. } f_i(x) - g_i(x) \leq c_i \textit{ for all } 1 \leq i \leq n \tag{15b}$$

In the following, we denote by $T_n\{f, x\}(x')$ the n^{th} order Taylor expansion of f at location x, that is, $T_1\{f, x\}(x') = f(x) + \langle x' - x, \partial_x f(x) \rangle$.

Algorithm 1 Constrained Concave Convex Procedure

```
Initialize $x_0$ with a random value
repeat
    find $x_{t+1}$ as the solution of the convex optimization problem

        minimize $f_0(x) - T_1\{g_0, x_t\}(x)$                (16a)
        s.t. $f_i(x) - T_1\{g_i, x_t\}(x) \leq c_i \forall i$   (16b)

until convergence of $x_t$
```

Proof The key idea of the proof is that for any convex function, the first order Taylor expansion is a lower bound, that is, $g_i(x) \geq T_1\{g_i, x_t\}(x)$ for all $x, x_t \in \mathcal{X}$. Consequently for all $x, x_t \in \mathcal{X}$ and $0 \leq i \leq n$ we have

$$f_i(x) - T_1\{g_i, x_t\}(x) \geq f_i(x) - g_i(x). \qquad (17)$$

By construction, equality holds at the point of expansion $x = x_t$. This means that for every x_t, (16) is an upper restriction of (15). In other words, every x feasible in (16b) is also feasible in (15b). Moreover, the objective function (16a) is an upper bound of (15a).

When $x = x_t$, the values of (15) and (16) match. Consequently, minimizing (16) leads to x_{t+1} with a lower value of the objective function (15a). This is because of two facts: Firstly, (16) presents an upper bound on (15). Secondly, when replacing the expansion at x_t by the one at x_{t+1} again the objective function may only decrease. To see this, observe that, by convexity we have

$$f_0(x_{t+1}) - T_1\{g_0, x_t\}(x_{t+1}) \geq f_0(x_{t+1}) - g_0(x_{t+1}),$$

but, by definition we have

$$f_0(x_{t+1}) - g_0(x_{t+1}) = f_0(x_{t+1}) - T_1\{g_0, x_{t+1}\}(x_{t+1}).$$

Next, we need to prove that if the x_t converge, then we actually arrived at a minimum or a saddlepoint of the optimization problem. We show this by proving that at stationarity a saddlepoint in the Lagrange function corresponding to (15) is also a saddlepoint in the Lagrange function corresponding to (16) with the same set of dual variables.

Now, assume that the above algorithm converges to x^* and let α^* be the dual variables of (16). By stationarity, the convex restriction at x^* satisfies the constraint qualifications and the Lagrange function of (16) has a saddle point in x^*, α^*.

However, by construction, the linearization is tight at x^*, so α^* also satisfies the Kuhn-Tucker conditions for (15a) and the derivatives of the Lagrangian of (15a) match those of their counterpart from (16a) at x^*. So we showed that if the convex restriction has a saddle point in the Lagrangian, so does the original problem. ■

This gives us a simple procedure to perform optimization even in a constrained nonconvex problem: simply linearize the constraints at every step and solve the resulting convex problem.

Remark 2 (CCP) *The CCP is a special case of theorem 1, where there are no constraints. In this case the first order conditions for the solution of (16a) amount to $\partial_\theta f_0(\theta) - \partial_\theta g_0(\theta_t) = 0$. This is exactly what [16] propose.*

3.2 Application to GP Classification

Recall that for Gaussian Process classification with missing variables the MAP-estimation problem (10) becomes that of solving

$$\text{minimize} \sum_{i=1}^{m} [g(\theta|x_i^o) - g(\theta|x_i^o, y_i)] + \frac{1}{2\sigma^2}\|\theta\|^2.$$

We define

$$\partial_\theta g(\theta|x^o) = \mathbb{E}_{p(x,y;\theta)}[\phi(x,y)|x^o;\theta] := E(\theta, x, y),$$

and

$$\partial_\theta g(\theta|x^o, y) = \mathbb{E}_{p(x,y;\theta)}[\phi(x,y)|x^o, y;\theta] := F(\theta, x, y).$$

Using the above, and the first-order optimality conditions of Remark 2, the Gaussian Process optimization problem can now be expressed as:

$$\sum_i E(\theta, x_i, y) - F(\theta_t, x_i, y_i) + \frac{1}{\sigma^2}\theta = 0. \qquad (18)$$

Note that while the first expectation depends on θ, the second one is taken for a *fixed* value θ_t, which is the solution of the previous iteration of the optimization problem. We can now specialize Algorithm 1 to this case by iterating the above repeatedly with respect to θ. To show that our algorithm is identical to the EM algorithm, we show that an identical optimization problem arises out of the EM algorithm.

Recall that in the *expectation* step of EM one computes the value of the expected log-likelihood with respect to the given set of parameters θ_t, that is, we compute

$$\mathbf{E}_{p(x,y;\theta_t)}\left[\sum_{i=1}^{m} -\log p(y_i, x_i^u|x_i^o, \theta) + \frac{1}{2\sigma^2}\|\theta\|^2\right]. \qquad (19)$$

Observe that

$$\mathbf{E}_{p(x,y;\theta_t)}\left[g(\theta|x_i^o)\right] = g(\theta|x_i^o),$$

and

$$\mathbf{E}_{p(x,y;\theta_t)}\left[\frac{1}{2\sigma^2}\|\theta\|^2\right] = \frac{1}{2\sigma^2}\|\theta\|^2.$$

Using the linearity of expectation, the above observations, and (5) we can re-write (19) as

$$\sum_{i=1}^{m} [g(\theta|x_i^o) - \langle F(\theta_t, x_i, y_i), \theta\rangle] + \frac{1}{2\sigma^2}\|\theta\|^2. \qquad (20)$$

In the *maximization* step of EM, one computes the value of θ which maximizes the above expectation. First order optimality conditions for (20) are found by taking derivatives with respect to θ and setting them to 0. This is equivalent to solving

$$\sum_i E(\theta, x_i, y) - F(\theta_t, x_i, y_i) + \frac{1}{\sigma^2}\theta = 0,$$

which is exactly the same as (18). This is not surprising, since the CCP is a generalization of the EM algorithm [16]. Things are more interesting in the case of Support Vector Machine classification.

3.3 Application to SV Classification

It is clear that (14) satisfies the conditions of Theorem 1: simply define

$$f_0(\theta, \xi) = \frac{1}{2\sigma^2}\|\theta\|^2 + \sum_{i=1}^m \xi_i \tag{21a}$$

$$f_i(\theta, \xi) = 1 - \xi_i + \max_{\tilde{y} \neq y_i} g(\theta|x_i^o, \tilde{y}) \tag{21b}$$

$$g_0(\theta) = 0 \text{ and } g_i(\theta) = g(\theta|x_i^o, y_i). \tag{21c}$$

We also set $c_i = 0$ for all i and write

$$T_1\{g_i, \theta_t\}(\theta) = g_i(\theta_t) + \langle \theta - \theta_t, F(\theta_t, x_i, y_i)\rangle .$$

If we define

$$d_i := 1 - \xi_i - g_i(\theta_t) + \langle \theta_t, F(\theta_t, x_i, y_i)\rangle ,$$

then each iteration Algorithm 1 requires solving the following optimization problem:

$$\min \frac{1}{2\sigma^2}\|\theta\|^2 + \sum_{i=1}^m \xi_i \tag{22a}$$

$$\text{s.t. } \langle F(\theta_t, x_i, y_i), \theta\rangle - \max_{\tilde{y} \neq y_i} g(\theta|x_i^o, \tilde{y}) \geq d_i \tag{22b}$$

$$\xi_i \geq 0. \tag{22c}$$

Since this is a convex optimization problem, standard Quadratic Programming (QP) packages can be used to solve it. Basically, what happens is that the expected value of $\Phi(x, y)$ with respect to the unknown part of x is used for classification. This is theoretical justification for the sometimes-used heuristic of estimating the values of the missing parameters and subsequently performing classification based on them. The main difference to this simple heuristic is that the margin of classification is defined as the difference between *pairs* of log-partition functions. This means that the conditional expectations depend on the (x^u, y) pair rather than on x^u alone.

4 Implementation

To make the above algorithms feasible in practice, several technical problems need to be overcome: it may not be possible to compute the log-partition function or its derivatives exactly. The solutions cease to be sparse, as they are given by linear combinations of conditional expectations. Sometimes, the dimensionality of the space might be so large that high dimensional integration techniques may need to be employed. In this section we discuss a few ideas which can be used to overcome the above problems.

The Representer Theorem: It follows from the generalized representer theorem [11] that the solution θ^* of both Support Vector Machine and Gaussian Process classification satisfies

$$\theta^* \in \text{span}\left\{\Phi(x_i^o, x_i^u, y) \text{ where } x_i^u, y \text{ are free}\right\}. \tag{23}$$

This means that the cardinality of the basis for θ is typically very large, sometimes even infinite. This might happen, for instance, when either the input space $\mathcal{X}$ or the label space $\mathcal{Y}$ have large dimensionality. This is clearly not desirable and we need an alternative. This is given in the form of an incomplete Cholesky factorization of the kernel matrix, either by sparse greedy approximation [13] or by positive diagonal pivoting [2]. For practical purposes we used the latter based on the kernel matrix arising from complete data pairs. The advantage is that instead of conditional expectations of $\Phi(x, y)$, which could be infinite dimensional, we now only need to compute conditional expectations over kernel values, that is

$$\langle \mathbf{E}_{x^u}[\Phi(x, y)|x^o; \theta], \Phi(x', y')\rangle = \mathbf{E}_{x^u}[k((x, y), (x', y'))|x^o]. \tag{24}$$

Likewise, second derivatives with respect to θ are given by covariances over kernel values.

In other words, instead of allowing the solution to lie in a possibly infinite dimensional space we constraint it to lie in a subspace spanned by the fully observed variables. This can lead to significant computational advantages. Of course, the downside is that the solution that we obtain might be sub-optimal since we are enforcing our constraint satisfaction conditions on only a subspace.

The log-partition function: The second issue, and arguably a very thorny one, is that one needs to be able to compute the value of the log-partition function for both the conditional as well as the unconditional densities. If suppose the number of missing variables is very small, and furthermore, if they can take only a small number of discrete values, then brute force computation of the conditional log-partition function is feasible.

In all other cases, we need to resort to methods for numerical quadrature, such as those discussed in [8].

The key difference to before is that now we will not even be able to reach a local optimum exactly but only up to the level of precision provided by the numerical integration method. A simple approximation is to use a Monte-Carlo estimate over the domain of missing variables instead of an exact integral. In other words, to compute

$$\mathbf{E}_{p(x,y;\theta)}[k((x,y),(x',y'))|x^o],$$

we use the approximation

$$\frac{\sum_{x^u \in X^u} k((x,y),(x',y'))e^{\langle \Phi(x,y),\theta \rangle}}{\sum_{x^u \in X^u} e^{\langle \Phi(x,y),\theta \rangle}}$$

where x^u is drawn uniformly from the domain of observations.

Stochastic Gradient Descent Finally, instead of performing a new Taylor expansion at every new step, we may also perform stochastic gradient descent on the objective function itself. This may be preferable whenever the constrained optimization problem becomes highly nontrivial. Essentially, in this case we only perform conditional expectations for the particular observation at hand. Standard considerations for stochastic gradient descent methods apply [5].

5 Experiments

We use the well known US Postal Service (USPS) dataset. It contains 9298 handwritten digits (7291 for training and 2007 for testing), collected from mail envelopes in Buffalo [7]. Upto 25% of pixels (64 pixels out of 256) from each data point in the training set were randomly selected and their values were erased. A Sparse Greedy matrix approximation using a maximum of 1000 basis functions was used to approximate the kernel matrix. We use the Gaussian kernel

$$k(x,x') = \exp\left(\frac{\|x-x'\|^2}{2\sigma^2}\right),$$

and tune the σ parameter using cross validation. Regularization parameters previously reported in the literature [10] were used for all our experiments. To estimate the integrals we used a Monte-Carlo sampling technique using 50 configurations of missing data generated uniformly at random. We then used a block Jacobi method in conjunction with the CCCP algorithm in order to train a multi-class Gaussian Process. We obtained the best error rate of 5.8%. Contrast this with the best error rate of around 4.0% reported for the Gaussian kernel on the same dataset [10]. We noticed that estimating the integrals by using many samples decreases the error rate but takes significantly longer amounts of time to compute and converge.

In the second experiment, we replaced the missing values by their mean values from other observed data. This is commonly known as mean imputation [3]. We obtain the best error rate of 6.08% for this procedure.

As can be see the error rates achieved by our method is marginally better than that obtained by mean imputation. This phenomenon was also observed by [3]. We believe that more sophisticated numerical integration techniques to estimate integrals will significantly improve the performance of our algorithm.

6 Discussion and Outlook

In this paper, we presented a principled method for dealing with missing data using exponential families in feature space. We outlined methods to deal with missing training data as well as partially observed labels. Transduction can be viewed as a special case of our framework. We then showed how Gaussian Processes and Support Vector Machines can be extended to missing data by using our framework. In order to solve the non-convex optimization problem that arises we presented a generalization of the Convex Concave Procedure to incorporate non-convex constraints. We also discussed a simple proof of convergence for our algorithm. Preliminary experimental results are encouraging.

Clearly, computation of the log-partition function is the most expensive step in our algorithm. Faster approximation algorithms viz. Quasi Monte Carlo sampling methods need to be explored for computing the log-partition function. Extending our results to graphical models and other similar density estimators remains the focus of future research.

Acknowledgments National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. This work was supported by grants of the ARC and sponsored by an NSF-ITR grant, award number IIS-0312401.

References

[1] Y. Altun, T. Hofmann, and A.J. Smola. Exponential families for conditional random fields. In *Uncertainty in Artificial Intelligence UAI*, 2004.

[2] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243 – 264, Dec 2001. http://www.jmlr.org.

[3] Z. Ghahramani and M. I. Jordan. Supervised learning from incomplete data via an EM approach. In J. D. Cowan, G. Tesauro, and J. Al-

spector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120 – 127. Morgan Kaufmann Publishers, Inc., 1994.

[4] R. E. Kass and P. W. Vos. *Geomtrical Foundations of Asymptotic Inference.* Wiley series in Probability and Statistics. Wiley Interscience, 1997.

[5] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transactions on Signal Processing*, 2003. To Appear.

[6] S. L. Lauritzen. *Graphical Models.* Oxford University Press, 1996.

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. J. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541 – 551, 1989.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterhing, and B. P. Flannery. *Numerical Recipes in C. The Art of Scientific Computation.* Cambridge University Press, 1994.

[9] G. Rätsch, S. Mika, and A. J. Smola. Adapting codes and embeddings for polychotomies. In *Neural Information Processing Systems*, volume 15. MIT Press, 2002.

[10] B. Schölkopf. *Support Vector Learning.* R. Oldenbourg Verlag, München, 1997. Doktorarbeit, TU Berlin. Download: http://www.kernel-machines.org.

[11] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 416 – 426, 2001.

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

[13] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In P. Langley, editor, *Proceedings of the International Conference on Machine Learning*, pages 911 – 918, San Francisco, 2000. Morgan Kaufmann Publishers.

[14] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Department of Statistics, September 2003.

[15] C. K. I. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*, pages 599 – 621. MIT Press, 1999.

[16] A.L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915 – 936, 2003.