Learning to Learn with the Informative Vector Machine

Neil D. Lawrence NEIL@DCS.SHEF.AC.UK
Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield, S1 4DP, U.K.

John C. Platt JPLATT@MICROSOFT.COM
Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, U.S.A.

Abstract

This paper describes an efficient method for learning the parameters of a Gaussian process (GP). The parameters are learned from multiple tasks which are assumed to have been drawn independently from the same GP prior. An efficient algorithm is obtained by extending the informative vector machine (IVM) algorithm to handle the multi-task learning case. The multi-task IVM (MT-IVM) saves computation by greedily selecting the most informative examples from the separate tasks. The MT-IVM is also shown to be more efficient than random sub-sampling on an artificial data-set and more effective than the traditional IVM in a speaker dependent phoneme recognition task.

1. Introduction

Recently, there has been a large amount of interest in the machine learning community in learning the kernel (equivalently, the covariance) of a Gaussian Process (GP), for either regression or classification (Williams, 1998). By learning the kernel from data, the accuracy of the GP can be improved. Typically, a GP kernel is parameterised by some vector $\boldsymbol{\theta}$, where regardless of the value of $\boldsymbol{\theta}$, the covariance will always be valid (a positive definite Mercer kernel).

Depending on the availability of data, the parameters $\boldsymbol{\theta}$ can be learned one of three different ways:

1. *The labelled training set* — The same data-set that is used to train the linear parameters of a GP model can also be used to estimate the kernel parameters $\boldsymbol{\theta}$. This is the most common situation. Several different approaches have been used to estimate the kernel, including maximum likelihood estimation and hierarchical Bayesian learning (Williams, 1998). If the labelled training set is small, then these methods may not estimate the covariance accurately.

2. *Unlabelled data* — If unlabelled data is available, then it can be used to define a kernel. Examples of such work is the Fisher kernel (Jaakkola et al., 1999) or the work of Seeger (2002), where a generative model of unlabelled data can be used to create a kernel that can work well. In the context of GPs, however, it is unclear whether such a kernel estimates the true covariance of the GP that the target function is drawn from.

3. *Labelled data from related tasks* — Using labelled data from related tasks is known as "learning to learn" or "multi-task learning" (Baxter, 1995; Thrun, 1996; Caruana, 1997). In the context of GPs, we assume that these related functions are drawn from the same GP and use the related data to fit the parameters $\boldsymbol{\theta}$. Minka and Picard (1997) were the first to link multi-task learning with fitting a GP covariance. If the related task data is plentiful, we can fit the parameters with low variance. This paper addresses the multi-task learning problem.

There are two main issues in multi-task learning with GPs. The first is *computational efficiency*: CPU time as a function of training set size. Computing the gradient of the log-likelihood of the data from M tasks with respect to $\boldsymbol{\theta}$ requires solving M linear systems, each with dimension N_m, where N_m is the number of training points in the mth task. Thus, multi-task learning with GPs can be computationally daunting.

The second issue is *statistical efficiency*: accuracy as a function of training set size. Approximating maximum likelihood may slow down convergence to the correct parameter vector. Since it is always possible to speed up maximum likelihood by throwing away data, any proposed algorithm exhibit better statistical efficiency than randomly sub-sampling the data.

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

This paper proposes the *multi-task informative vector machine* (MT-IVM), which is a computationally efficient algorithm to fit a GP covariance to multiple tasks. MT-IVM adapts the IVM algorithm (Lawrence et al., 2003) to the multi-task case. Similar to the IVM, the MT-IVM selects a subset of data from the tasks. The data-points are selected by a simple, fast heuristic: choose the points that minimise the entropy of the posterior process. This heuristic tends to choose points that are maximally informative. MT-IVM is computationally efficient and more accurate than maximum likelihood with sub-sampled data. This is the first multi-task GP algorithm to be tested for both statistical and computational efficiency.

1.1. Structure of Paper

In order to describe MT-IVM, we first review Gaussian Processes in section 2 and the original IVM in section 3. In section 4, we extend the IVM to multiple tasks by mapping the multi-task GP back to a single larger GP. Section 5 presents experimental results: we show that MT-IVM is better than random subsampling for multi-task learning of GPs. Section 5.2 shows that we can get faster training by treating speaker-independent phoneme recognition as a multi-task problem (one task per speaker) rather than one unified task.

2. Gaussian Processes

In this section we briefly review Gaussian processes, introducing the notation that we will use in the following sections. Consider a simple latent variable model for data where the observations, $\mathbf{y} = [y_1 \dots y_N]^{\mathrm{T}}$, are independent from input data, $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N]^{\mathrm{T}}$, given a set of latent variables, $\mathbf{f} = [f_1 \dots f_N]^{\mathrm{T}}$. The prior distribution over the latent variables is given by a GP,

$$p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) = N(\mathbf{0}, \mathbf{K}),$$

with covariance function, or 'kernel', $\mathbf{K}$ which is parameterised by the vector $\boldsymbol{\theta}$ and evaluated at the points given in $\mathbf{X}$. This relationship is shown graphically in Figure 1.

The joint likelihood of the data can be written as

$$p(\mathbf{y}, \mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) = p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) \prod_{n=1}^{N} p(y_n|f_n) \qquad (1)$$

where $p(y_n|f_n)$ gives the relationship between the latent variable and our observations and is sometimes referred to as the noise model. For the least squares regression case (see *e.g.* (Williams & Rasmussen, 1996)) the noise model takes the form of a Gaussian[1] and the marginalised likelihood can be computed without approximations. Unfortunately when the noise model is non-Gaussian this marginalisation is not possible. One solution to this problem is to approximate each factor of the noise model, $p(y_n|f_n)$ with a Gaussian distribution, $N\left(z_n|f_n, \beta_n^{-1}\right)$ with mean z_n and precision β_n. If this is done in a sequential manner through minimisation of a KL divergence the approach is known as assumed-density filtering (ADF), see *e.g.* Csató (2002). Thus we start with the GP prior $q_0(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) = p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})$ and we include the nth factor of the noise model $\hat{p}_1(\mathbf{f}) \propto q_0(\mathbf{f}) p(y_n|x_n)$ to obtain a posterior distribution. This posterior is then approximated by minimising the KL divergence between $\hat{p}_1(\mathbf{f})$ and a Gaussian distribution $q_1(\mathbf{f})$,

$$\mathrm{KL}\left(\hat{p}_i(\mathbf{f}) | q_i(\mathbf{f})\right) = \int \hat{p}_i(\mathbf{f}) \ln \frac{\hat{p}_i(\mathbf{f})}{q_i(\mathbf{f})} d\mathbf{f}.$$

A further data-point is then included and the approximation is repeated. This is continued until all data has been included. The tractability of the approximation relies on the normalisation constant for $\hat{p}_i(\mathbf{f})$,

$$Z_i = \int p(y_n|f_n) q_{i-1}(\mathbf{f}) d\mathbf{f}$$

where

$$q_i(\mathbf{f}) = N(\mathbf{f}|\boldsymbol{\mu}_i, \Sigma_i)$$

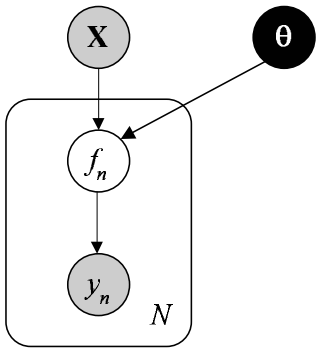

Figure 1. The Gaussian Process model drawn graphically. Nodes are shaded gray to indicate that they are observed variables, and white to indicate that they require marginalisation. A black node indicates that the variable will be optimised. We have made use of 'plate' notation to indicate the independence relationship between $\mathbf{f}$ and $\mathbf{y}$. This plate indicates that the nodes within it are repeated N times in the graph with their edges preserved.

[1] We use $N(\mathbf{x}|\boldsymbol{\mu}, \Sigma)$ to denote a Gaussian distribution over $\mathbf{x}$ with a mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. When dealing one dimensional Gaussians the vectors and matrices are replaced by scalars.

By differentiating $\ln Z_i$ with respect to $\mu_{i-1,n}$ (the nth element of $\boldsymbol{\mu}_{i-1}$) we can show that for $p(y_n|f_n) = N\left(y_n|f_n, \beta_n^{-1}\right)$

$$\frac{\partial \ln Z_i}{\partial \mu_{i-1,n}} = \frac{y_n - \mu_{i-1,n}}{\beta_n^{-1} + \varsigma_{i-1,n}} \doteq g_{in}, \tag{2}$$

Similarly we differentiate the log partition with respect to $\varsigma_{i-1,n}$, the nth diagonal element of Σ_{i-1}, to find

$$\frac{\partial \ln Z_i}{\partial \varsigma_{i-1,n}} = -\frac{1}{2}\left(\varsigma_{i-1,n} + \beta_n^{-1}\right)^{-1} + \frac{1}{2}g_{in}^2 \doteq \gamma_{in},$$

and defining $\nu_{in} = -\gamma_{in} + \frac{1}{2}g_{in}^2$ we have

$$\nu_{in} = \left(\varsigma_{i-1,n} + \beta_n^{-1}\right)^{-1}. \tag{3}$$

Using this notation, under the ADF scheme, the following update equations hold,

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + g_{in}\mathbf{s}_{i-1,n}, \tag{4}$$

$$\Sigma_i = \Sigma_{i-1} - \nu_{in}\mathbf{s}_{i-1,n}\mathbf{s}_{i-1,n}^{\mathrm{T}}, \tag{5}$$

where $\mathbf{s}_{i-1,n}$ is the nth row from $\Sigma_{i-1,n}$.

2.1. Classification Noise Model

For classification we consider the *probit* noise model,

$$p(y_n|f_n) = \phi(y_n(f_n + b))$$

where $\phi(\cdot)$ is the cumulative Gaussian[2] and $y_n \in \{1, -1\}$,. The partition function is again found by taking the expectation of $p(y_n|f_n)$ under the marginal distribution $q_{i-1}(\mathbf{f})$,

$$Z_i = \phi(u_{i-1,n})$$

where $u_{i-1,n} = c_{i-1,n}(\mu_{i-1,n} + b)$ and $c_{i-1,n} = y_n(1 + \varsigma_{i-1,n})^{-\frac{1}{2}}$. Performing the necessary derivatives to obtain g_{in} and ν_{in} we have[3]

$$g_{in} = c_{i-1,n}\mathcal{N}(u_{i-1,n}|0,1)\left[\phi(u_{i-1,n})\right]^{-1}, \tag{6}$$

and

$$\nu_{in} = g_{in}(g_{in} + u_{i-1}c_{i-1,n}). \tag{7}$$

In practice we wish to summarise our approximation to the likelihood in terms of β_n and z_n. This relationship can easily be found by replacing y_n with z_n in (2) and (3) and re-arranging to obtain

$$z_n = \frac{g_{in}}{\nu_{in}} + \mu_{i-1,n} \tag{8}$$

$$\beta_n = \frac{\nu_{in}}{1 - \nu_{in}\varsigma_{i-1,n}} \tag{9}$$

Updates for $\boldsymbol{\mu}_{i-1} \rightarrow \boldsymbol{\mu}_i$ and $\Sigma_{i-1} \rightarrow \Sigma_i$, the parameters of $q(\mathbf{f})$, are then as those given in (4) and (5).

[2]Given by $\phi(z) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z}\exp\left(-\frac{t^2}{2}\right)dt$.

[3]Care must be taken in computing g_{in} when $u_{i-1,n}$ has large magnitude as both $\phi(\cdot)$and $N(\cdot)$ become small.

3. The IVM Reduces Computation

The assumed density filtering (ADF) approach outlined above assumes that all data-points will be made use of in determining the model. One problem with this is that including all N data-points gives the algorithm $O\left(N^3\right)$ complexity. Even more of a concern is that if we wish to find the parameters of the kernel, $\boldsymbol{\theta}$, by gradient based optimisation of the (log) likelihood, each gradient evaluation will be $O\left(N^3\right)$. It is important to find a method for reducing this complexity. In this section we will review the informative vector machine (IVM) which aims to find a sparse representation of the data-set, thereby reducing the computational cost (Lawrence et al., 2003). Then, in section 4, we return to the the concept of learning from multiple tasks and show how it can be done efficiently with Gaussian processes.

3.1. Data-point Selection with the IVM

The informative vector machine approach to greedily obtaining a sparse representation of the data-set gives an approximation to the solution in $O\left(d^2N\right)$ operations, where d is the number of data-points included in the sparse representation. We will denote the set of these 'active points' with I and the set of those which are not included with J.

The data-points in I are greedily selected using a simple criterion inspired by information theory: the change in entropy of the posterior process. For an individual point, n, as a candidate for the ith inclusion this entropy change is given by

$$\Delta H_{in} = -\frac{1}{2}\log(1 - \nu_{in}\varsigma_{i-1,n}), \tag{10}$$

where $\varsigma_{i-1,n}$ is the nth diagonal element from Σ_{i-1}. Other criteria (such as information gain) are also straightforward to compute.

The idea behind the IVM is to greedily minimise the entropy of the true posterior through incorporating data-points that most reduce the entropy in a sequential manner. We note from (10) that in order to score each point we need to keep track of $\boldsymbol{\varsigma}_i$, the diagonal of Σ_{i-1}, and ν_{in}. If these can be efficiently computed or stored then we will have an efficient algorithm.

Note that maintaining Σ_{i-1} in memory would require $O\left(N^2\right)$ storage, which is undesirable, so our first quest is to seek an efficient representation of the posterior covariance. From (5) it is clear that Σ_i has a particular structure, where successive outer products are added to the original prior covariance $\Sigma_0 = \mathbf{K}$ to form the

current covariance. This can be represented by

$$\Sigma_i = \mathbf{K} - \mathbf{M}_i^{\mathrm{T}}\mathbf{M}_i$$

where the kth row of $\mathbf{M}$ is given by $\sqrt{\nu_{k,n_k}}\mathbf{s}_{k-1,n_k}$ and n_k represents kth included data-point. Naturally, if we are not storing Σ_{i-1} , we will not be able to represent (for instance) $\mathbf{s}_{i-1,n_i}$ directly. However it can be computed from $\mathbf{M}_{i-1}$ and $\mathbf{K}$.

$$\mathbf{s}_{i-1,n_i} = \mathbf{k}_{n_i} - \mathbf{a}_{i-1,n_i}^{\mathrm{T}}\mathbf{M}_{i-1} \tag{11}$$

where $\mathbf{k}_{n_i} = \mathbf{s}_{0,n_i}$ is the n_ith row of $\mathbf{K} = \Sigma_0$ and $\mathbf{a}_{i-1,n_i}$ is the n_ith column of $\mathbf{M}_{i-1}$. This computation will require $O\left((i-1)N\right)$ operations and dominates each point inclusion resulting in an algorithm of complexity $O\left(d^2N\right)$ for d inclusions.

From (4) and (5) it can be seen that the diagonal of Σ_i, $\boldsymbol{\varsigma}_i$, can be updated by

$$\boldsymbol{\varsigma}_i = \boldsymbol{\varsigma}_{i-1} - \nu_{i,n_i}\mathrm{diag}\left(\mathbf{s}_{i-1,n_i}\mathbf{s}_{i-1,n_i}^{\mathrm{T}}\right) \tag{12}$$

and the mean output vector can be updated by

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + g_{in_i}\mathbf{s}_{i-1,n_j}. \tag{13}$$

Storage requirements are dominated by $\mathbf{M}_i$ which at maximum is an $N \times d$ matrix.

The overall algorithm for the IVM is given in Algorithm 1.

Algorithm 1 The standard IVM algorithm.

Require: d a number of active points. For classification $\mathbf{z} = \mathbf{0}$ and $\boldsymbol{\beta} = \mathbf{0}$. For regression substitute appropriate target values. Take $\boldsymbol{\varsigma}_0 = \mathrm{diag}\left(\mathbf{K}\right)$, $\boldsymbol{\mu} = \mathbf{0}$, $J = \{1, \ldots, N\}$, $I = \{\cdot\}$, $\mathbf{M}_0$ is an empty matrix.
  for $i = 1$ to d **do**
    for all $n \in J$ **do**
      compute g_{in} according to (6) (not required for Gaussian).
      compute ν_{in} according to (7) or (3).
      compute ΔH_{in} according to (10).
    end for
    $n_i = \mathrm{argmax}_{n \in J} \Delta H_{in}$.
    Update z_{n_i} and β_{n_i} using (8) and (9).
    Compute $\boldsymbol{\varsigma}_i$ and $\boldsymbol{\mu}_i$ using (11), (12) and (13).
    Append $\sqrt{\nu_{in_i}}\mathbf{s}_{i-1,n_i}^{\mathrm{T}}$ to $\mathbf{M}_{i-1}$ to form $\mathbf{M}_i$.
    Add n_i to I and remove n_i from J.
  end for

Optimisation of kernel parameters can now be achieved through maximisation of the approximation to the marginal likelihood,

$$p\left(\mathbf{y}\right) \approx N\left(\mathbf{z}|\mathbf{0}, \mathbf{K} + \mathbf{B}^{-1}\right),$$

where $\mathbf{B} = \mathrm{diag}\left(\boldsymbol{\beta}\right)$. The dependence of the likelihood on $\mathbf{y}$ is indirect and through $\mathbf{z}$ and $\boldsymbol{\beta}$. Gradients of the log-likelihood with respect to kernel parameters, $\boldsymbol{\theta}$ may be computed and used in a non-linear optimiser such as scaled conjugate gradients to develop an estimate $\hat{\boldsymbol{\theta}}$.

This completes our review of the IVM, we now describe how the same principles may be applied when learning from multiple tasks.

4. Multi-task Learning

In this section we will extend the IVM approach to handle the situation where we have multiple independent tasks. We assume that we are given M training sets from tasks which are independent given a vector of parameters $\boldsymbol{\theta}$ and an input matrix $\mathbf{X}_m$ (see Figure 2). We model the target data for each task, $\mathbf{y}_m$, as a GP so that the probability distribution for the matrix $\mathbf{Y}$, whose columns are $\mathbf{y}_m$, is

$$p\left(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}\right) = \prod_{m=1}^{M} p\left(\mathbf{y}_m|\mathbf{X}_m, \boldsymbol{\theta}\right)$$

where each $p\left(\mathbf{y}_m|\mathbf{X}_m, \boldsymbol{\theta}\right)$ is a Gaussian process.

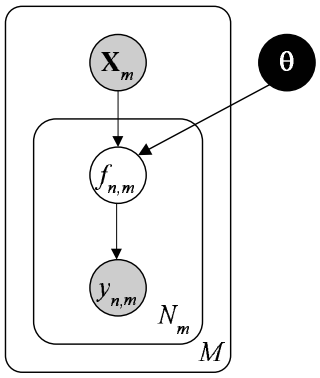

Figure 2. A graphical model which represents a multi-task GP.

The entire likelihood can be considered to be a Gaussian process over a vector $\mathbf{y}$ which is formed by stacking columns of $\mathbf{Y}$, $\mathbf{y} = \left[\mathbf{y}_1^{\mathrm{T}} \ldots \mathbf{y}_M^{\mathrm{T}}\right]^{\mathrm{T}}$. The covariance matrix is then

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_M \end{bmatrix}$$

and we write

$$p\left(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}\right) = N\left(\mathbf{0}, \mathbf{K}\right). \tag{14}$$

This establishes the equivalence of the multi-task GP to a standard GP. Once again we obtain an estimate, $\hat{\boldsymbol{\theta}}$, of the parameters by optimising the log-likelihood with respect to the parameters $\boldsymbol{\theta}$. These gradients require the inverse of $\mathbf{K}$ and, while we can take advantage of its block diagonal structure to compute this inverse, we are still faced with inverting $N_m \times N_m$ matrices, where N_m is the number of data-points associated with task m: so we look to the IVM algorithm to sparsify the GP specified by (14).

It is straightforward to show that the new posterior covariance structure after k inclusions, $q_k(\mathbf{f})$, will also be a Gaussian with a block-diagonal covariance matrix,

$$q_k(\mathbf{f}) = \prod_{m=1}^{M} N\left(\mathbf{f}_m | \boldsymbol{\mu}_i^{(m)}, \Sigma_i^{(m)}\right). \tag{15}$$

Note that k inclusions in total does not mean k inclusions for each task. The i in (15) represents the number of inclusions for each task. The value of i will vary with m, but we prefer to drop this dependence to avoid further cluttering of our notation. Each block of the posterior covariance is

$$\Sigma_i^{(m)} = \mathbf{K}_m - \mathbf{M}_i^{(m)\,\mathrm{T}} \mathbf{M}_i^{(m)},$$

where the rows of $\mathbf{M}_i^{(m)}$ are given by $\sqrt{\nu_{in_i}^{(m)}}\mathbf{s}_{i-1,n_i}^{(m)}$. The means associated with each task are given by

$$\boldsymbol{\mu}_i^{(m)} = \boldsymbol{\mu}_{i-1}^{(m)} + g_{in_i}^{(m)} \mathbf{s}_{i-1,n_i}^{(m)} \tag{16}$$

and updates of $\boldsymbol{\varsigma}_i^{(m)}$ can still be achieved through

$$\boldsymbol{\varsigma}_i^{(m)} = \boldsymbol{\varsigma}_{i-1}^{(m)} - \nu_{i,n_i}^{(m)} \mathrm{diag}\left(\mathbf{s}_{i-1,n_i}^{(m)} \mathbf{s}_{i-1,n_i}^{(m)\,\mathrm{T}}\right). \tag{17}$$

From (16) and (17) it is obvious that the updates of $q_i^{(m)}(\mathbf{f}_m)$ are performed independently for each of the M models. Point selection, however, should be performed across models, allowing the algorithm to select the most informative point both within and across the different tasks.

$$\Delta H_{in}^{(m)} = -\frac{1}{2}\log\left(1 - \nu_{in}^{(m)} \varsigma_{i-1,n}^{(m)}\right).$$

We must also need to maintain an active, $I^{(m)}$, and an inactive, $J^{(m)}$, set for each task. The details of the algorithm are given in Algorithm 2.

The effect of selecting across tasks, rather than selecting independently within tasks is shown by a simple experiment in Figure 3. Here there are three tasks, each contains 30 data-points sampled from sine waves with frequency $\frac{\pi}{5}$ and differing offsets. The tasks used

Algorithm 2 The multi-task IVM algorithm.

Require: d the number of active points.
  for $m = 1$ to M **do**
    For classification $\mathbf{z}^{(m)} = \mathbf{0}$ and $\boldsymbol{\beta}^{(m)} = \mathbf{0}$. For regression substitute appropriate target values. Take $\boldsymbol{\varsigma}_0^{(m)} = \mathrm{diag}(\mathbf{K}_m)$, $\boldsymbol{\mu}^{(m)} = \mathbf{0}$, $J^{(m)} = \{1, \ldots, N_m\}$, $\mathbf{M}_0^{(m)}$ is an empty matrix.
  end for
  for $k = 1$ to d **do**
    for $m = 1$ to M **do**
      for all $n \in J^{(m)}$ **do**
        compute $g_{kn}^{(m)}$ according to (6) (not required for Gaussian).
        compute $\nu_{kn}^{(m)}$ according to (7) or (3).
        compute $\Delta H_{kn}^{(m)}$ according to (10).
      end for
      {Comment: Select largest reduction in entropy for each task.}
      $\Delta H_k^{(m)} = \max_n \Delta H_{kn}^{(m)}$
      $n_k^{(m)} = \mathrm{argmax}_{n \in J} \Delta H_{kn}$.
    end for
    {Comment: Select the task with the largest entropy reduction.}
    $m_k = \mathrm{argmax}_{m \in J} \Delta H_k^{(m)}$, $n_i = n_k^{(m_k)}$
    Update m_{n_i} and $\beta_{n_i}^{(m_k)}$ using (8) and (9).
    Compute $\boldsymbol{\varsigma}_i^{(m_k)}$ and $\boldsymbol{\mu}_i^{(m_k)}$ using (11), (12) and (13).
    Append $\sqrt{\nu_{in_i}^{(m_k)}}\ \mathbf{s}_{i-1,n_i}^{(m_k)\,\mathrm{T}}$ to $\mathbf{M}_{i-1}^{(m_k)}$ using (11) to form $\mathbf{M}_i$.
    Add n_i to $I^{(m_k)}$ and remove n_i from $J^{(m)}$.
  end for

different distributions for the input data: in the first it was sampled from a strongly bimodal distribution; in the second it was sampled from a zero mean Gaussian with standard deviation of 2 and in the third task data was sampled uniformly from the range $[-15, 15]$. An MT-IVM with $d = 15$ and an RBF kernel of width 1 was trained on the data. The data-points that the MT-IVM used are circled. Note that all but six of the points came from the third task. The first and second task contain less information because the input data is less widely distributed, thus the MT-IVM algorithm relies most heavily on the third task. This toy example illustrates the importance of selecting the data-points from across the different tasks.

To determine the kernel parameters, we again maximise the approximation to the likelihood,

$$p(\mathbf{Y}) \approx \prod_{m=1}^{M} p\left(\mathbf{z}_m | \mathbf{0}, \mathbf{K}_m + \mathbf{B}_m^{-1}\right).$$

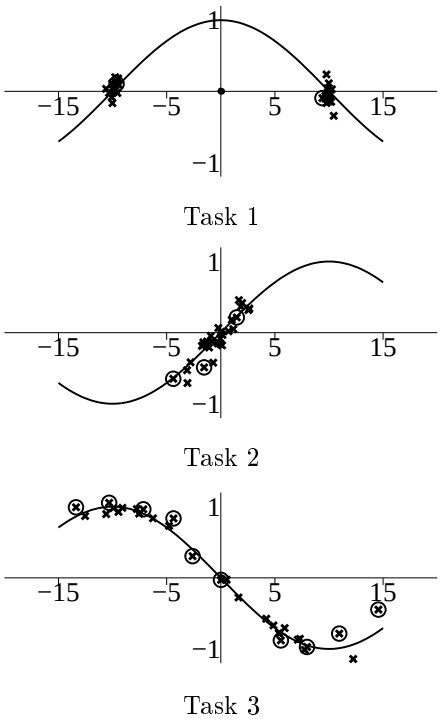

Figure 3. Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

5. Results

We show the effectiveness of the MT-IVM on two different tasks. First, in section 5.1, we take a GP with a covariance function with known parameters. We hide the parameters from the MT-IVM, then generate a series of tasks from the GP, feed the training data for the tasks to the MT-IVM, and measure how quickly the MT-IVM converges to the true results. The second test, in section 5.2, uses the MT-IVM as a classification algorithm on a small, real data-set for phoneme recognition. The traditional way to view phoneme recognition is to gather a data-set from many speakers, throw away the speaker labels, then train a single speaker-independent model for the phones. Given the MT-IVM, we can treat speakers as tasks, which are independent given the GP covariance for the phonemes. We then test to see if the MT-IVM converges faster than a speaker-independent GP model trained with IVM.

5.1. Regression with the Multi-task IVM

The first test generates data from an artificial M-task GP with a covariance function parameterised by $\boldsymbol{\theta}$

$$k_m\left(\mathbf{x}_i^{(m)}, \mathbf{x}_j^{(n)}\right) = \theta_2 \exp\left(-\theta_1 E_{ij}^{(m)}\right) + \theta_3^{-1}\delta_{ij} + \theta_4$$

where $\mathbf{x}_i^{(m)}$ is the ith data-point from the mth task, δ_{ij} is the Dirac delta function and

$$E_{ij}^{(m)} = \frac{1}{2}\left(\mathbf{x}_i^{(m)} - \mathbf{x}_j^{(m)}\right)^{\mathrm{T}}\left(\mathbf{x}_i^{(m)} - \mathbf{x}_j^{(m)}\right).$$

We generate 'training tasks' from a Gaussian process with $\boldsymbol{\theta} = [1, 1, 100, 0]$. For each task: the number of input dimensions was 4 and $N_m = 2000$; half the input vectors, $\mathbf{X}_m$, were sampled independently from a spherical Gaussian distribution with mean in each direction of 1 and variances of 0.125; the other half were sampled from a Gaussian with the same covariance but a mean in each direction of -1; the $\mathbf{y}_m$ values were sampled from a GP parameterised by $\boldsymbol{\theta}$.

To optimise the MT-IVM we initialised $\hat{\boldsymbol{\theta}} = [10, 10, 10, 10]$ and selected an active-set. The likelihood of the active-set was then optimised with respect to the parameters using scaled conjugate gradients until convergence or a maximum of 50 iterations. The active-set was then reselected using the new estimate for $\hat{\boldsymbol{\theta}}$. This process was repeated five times.

We compared the MT-IVM approach with random sub-sampling of the the data-set. To optimise in this case we simply maximised the likelihood of the sub-sampled data using scaled conjugate gradients for a maximum of 200 iterations.

For the MT-IVM we considered active set[4] sizes of 400 to 1000 at intervals of 100. For uniform sub-sampling we took 150 to 600 samples from each task[5] at intervals of 50 samples.

The quality of the parameter estimates was measured using the KL divergence evaluated for a separate set of test points:

$$\mathrm{KL}\left(\boldsymbol{\theta}||\hat{\boldsymbol{\theta}}\right) = \int p\left(\mathbf{y}_{\mathrm{test}}|\boldsymbol{\theta}\right) \ln \frac{p\left(\mathbf{y}_{\mathrm{test}}|\boldsymbol{\theta}\right)}{p\left(\mathbf{y}_{\mathrm{test}}|\hat{\boldsymbol{\theta}}\right)} d\mathbf{y},$$

Ten runs were made for each sub-sample or active set size. Results are presented in Figure 4. In the top figure, we simultaneously test for both statistical and computational efficiency by plotting the KL divergence as a function of CPU time. Note that with about 170 seconds of training time the MT-IVM is already close

[4]Note that this is the active set across all tasks so we are using d/M points on average per task.

[5]In other words 600 to 2400 total samples at intervals of 200.

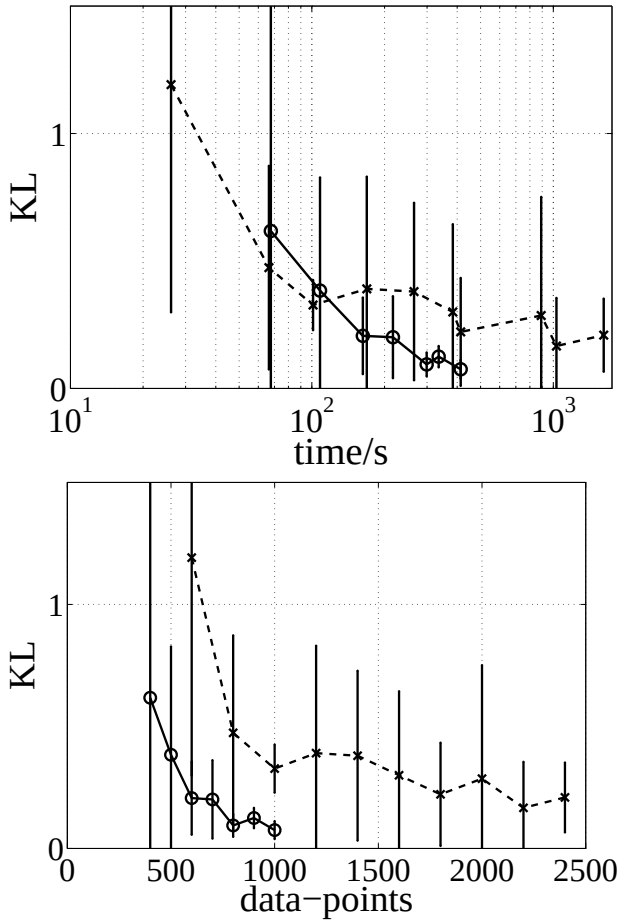

Figure 4. Plot of KL divergence vs average time (*top*) and KL divergence vs data-points used (*bottom)* for the two methods. The IVM is represented by circles and a solid line, sub-sampling is represented by crosses and a dashed line. The results summarise the ten 'runs' with means and error bars on the KL divergence shown at one standard deviation.

to a KL divergence of zero, whereas for sub-sampling even with over 1500 seconds of training time the mean of the KL divergence is still some distance from zero. This difference is explained by the statistical efficiency of the algorithm (bottom of Figure 4) because the MT-IVM actively selects data-points it achieves far lower KL divergences with fewer data-points. There is a small time penalty associated with the active point-selection, but it is insignificant when compared to its associated benefits.

5.2. Classification with the Multi-task IVM

In this section we turn to a speech example from the UCI repository (Blake & Merz, 1998) The data consists of 15 different speakers saying 11 different phonemes 6 times each (giving 66 training points for each speaker). Our aim will be to learn kernel parameters that are appropriate for classifying the different phonemes. To this end, we will consider that each speaker is independent given the kernel parameters associated with the phoneme, *i.e.* we treated each speaker as a separate task. We used 14 of the speakers to learn the kernel parameters for each phoneme giving 14 tasks. Model evaluation was then done by taking one example of each phoneme from the remaining speaker (11 points) and using this data to construct a new Gaussian process model based on those kernel parameters. Then we evaluated this model's performance on the remaining 55 points for that speaker. This mechanism was used for both an MT-IVM model and an ADF trained GP where points were randomly sub-sampled.

To demonstrate the utility of the multi-task framework we also built a IVM based Gaussian process model on the 14 speakers ignoring which speaker was associated with each data point (924 training points). The kernel parameters were optimised for this model and then the model was evaluated as above.

For enough information to be stored by the kernel about the phonemes it needs to be 'parameter rich'. We therefore used a kernel which could scale each input

$$k_m\left(\mathbf{x}_i^{(m)}, \mathbf{x}_j^{(n)}\right) = \theta_2 \exp\left(-\theta_1 E_{ij}^{(m)}\right) + \theta_5 \, \mathbf{x}_i^{(m)\,\mathrm{T}} \mathbf{D} \mathbf{x}_j^{(m)} + \theta_3^{-1}\delta_{ij} + \theta_4$$

where $\mathbf{x}_i^{(m)}$ is the ith data-point from the mth task, δ_{ij} is the Dirac delta function,

$$E_{ij}^{(m)} = \frac{1}{2}\left(\mathbf{x}_i^{(m)} - \mathbf{x}_j^{(m)}\right)^{\mathrm{T}} \mathbf{D} \left(\mathbf{x}_i^{(m)} - \mathbf{x}_j^{(m)}\right),$$

$\mathbf{D} = \mathrm{diag}\left(\theta_6 \ldots \theta_{6+K}\right)$ and K is number of elements in each input vector $\mathbf{x}$. This lead to a total of 15 parameters for the kernel.

The results on the speech data are shown in Figure 5. The convergence of MT-IVM to $\approx 10\%$ error is roughly 10 times faster than the IVM. The MT-IVM takes advantage of the assumed independence to train much faster than the regular IVM. While this data-set is relatively small, the structure of this experiment is important. One reason for the popularity of the combination HMM-Gaussian mixture model for modelling in speech is the ease with which these generative models can be modified to take account of an individual speakers — this is known as speaker-dependent recognition. Up until now it has not been clear how to achieve this

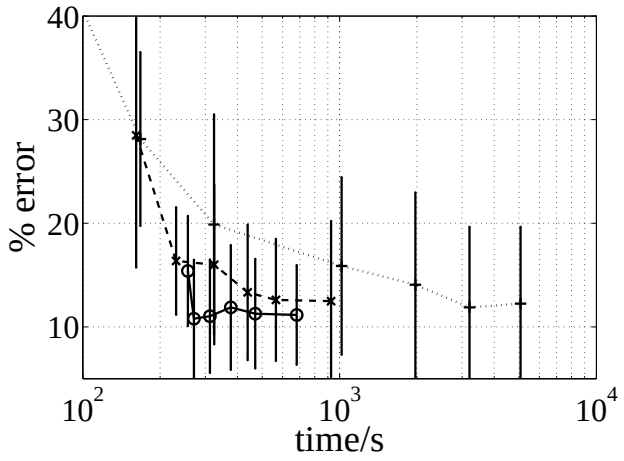

Figure 5. Average average time vs error rate for a MT-IVM (solid line with circles) and sub-sampled ADF-GP (dashed line with crosses) whose kernel parameters are optimised by considering each speaker to be an independent task and an IVM optimised by considering all points to belong to the same task (dotted line with pluses).

with discriminative models. The approach we are suggesting may be applied to large vocabulary word recognisers and used in speaker-dependent recognition.

6. Conclusions

In order to efficiently train multi-task Gaussian process regression and classification, we have generalised the informative vector machine to the multi-task case. We modelled the multi-task training sets as being independent, given the parameters of the covariance for all of the tasks. Given this model, the covariance for the multi-task data is block diagonal, where each block arises from each task. We then apply the standard IVM algorithm to find the maximum likelihood estimate of the underlying parameters of the Gaussian process. The IVM algorithm does not compute the full covariance matrix for all tasks. Rather, it efficiently selects points from all of the tasks, by a heuristic that every point should minimise the entropy of the posterior process.

We have shown that multi-task IVM is more efficient (from a combined statistical/computational efficiency standpoint) than random sub-sampling. This efficiency is shown by tests on artificial regression data and a phoneme recognition data-set. We also showed that re-expressing a speaker-independent phoneme task as multiple speaker-dependent tasks makes training much more efficient.

A software implementation of the MT-IVM is available from `http://www.dcs.shef.ac.uk/~neil/mtivm/`.

References

Baxter, J. (1995). Learning internal representations. *Proc. COLT* (pp. 311–320). Morgan Kaufmann Publishers.

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases.

Caruana, R. (1997). Multitask learning. *Machine Learning, 28*, 41–75.

Csató, L. (2002). *Gaussian processes — iterative sparse approximations.* Doctoral dissertation, Aston University.

Jaakkola, T. S., Diekhaus, M., & Haussler, D. (1999). Using the Fisher kernel method to detect remote protein homologies. *7th Intell. Sys. Mol. Biol.*, 149–158.

Lawrence, N. D., Seeger, M., & Herbrich, R. (2003). Fast sparse Gaussian process methods: The informative vector machine. *Advances in Neural Information Processing Systems* (pp. 625–632). Cambridge, MA: MIT Press.

Minka, T. P., & Picard, R. W. (1997). Learning how to learn is learning with point sets. Web. Revised 1999, available at `http://www.stat.cmu.edu/~minka/`.

Seeger, M. (2002). Covariance kernels from bayesian generative models. *Advances in Neural Information Processing Systems* (pp. 905–912). Cambridge, MA: MIT Press.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In (Touretzky et al., 1996), 640–646.

Touretzky, D. S., Mozer, M. C., & Hasselmo, M. E. (Eds.). (1996). *Advances in neural information processing systems*, vol. 8. Cambridge, MA: MIT Press.

Williams, C. K. I. (1998). Prediction with Gaussian processes: From linear regression to linear prediction and beyond. *Learning in Graphical Models.* Dordrecht, The Netherlands: Kluwer.

Williams, C. K. I., & Rasmussen, C. E. (1996). Gaussian processes for regression. In (Touretzky et al., 1996), 514–520.