# Gaussian Process Classification and SVM: Mean Field Results and Leave-One-Out Estimator

Manfred Opper
Neural Computing Research Group,
Department of Computer Science and Applied Mathematics,
Aston University, Birmingham B4 7ET, United Kingdom

Ole Winther
Theoretical Physics II, Lund University, Sölvegatan 14 A,
S-223 62 Lund, Sweden

March 19, 1999

**Abstract**

In this chapter, we elaborate on the well-known relationship between Gaussian Processes (GP) and Support Vector Machines (SVM). Secondly, we present approximate solutions for two computational problems arising in GP and SVM. The first one is the calculation of the posterior mean for GP classifiers using a 'naive' mean field approach. The second one is a leave-one-out estimator for the generalization error of SVM based on a linear response method. Simulation results on a benchmark dataset show similar performances for the GP mean field algorithm and the SVM algorithm. The approximate leave-one-out estimator is found to be in very good agreement with the exact leave-one-out error.

# 1 Introduction

It is well-known that Gaussian Processes (GP) and Support Vector Machines (SVM) are closely related, see eg. [7]. Both approaches are non-parametric. This means that they allow for infinitely many parameters to be tuned, but increasing with the amount of data, only a finite number of them are active. Both types of models may be understood as generalizations of single layer perceptrons, where each input node to the perceptron computes a distinct nonlinear feature of the original inputs to the machine. In principle, the number of such features (and of the corresponding perceptron weights) can be arbitrarily large. However, by the specific training method, such vast increase in complexity does not necessarily result in overfitting.

For the support vector machine (in its simplest version), a quadratic optimization algorithm maximizes the gap between positive and negative examples. A simple mathematical analysis of this optimization problem shows that all the weights become just linear combinations of the input feature vectors. Hence, the corresponding coefficients in this combination are the new parameters to be calculated. Their number never exceeds the number of examples. Moreover, it is not necessary to evaluate the many nonlinear feature vectors during the calculations, but all calculations are expressed by a kernel function which is the inner product of two vectors of features at different input points. In fact, one need not even specify the non-linear features explicitly, but any positive semidefinite kernel function will implicitly define such features.

A second way to regularize this problem comes from the Bayesian approach. Here, one introduces a prior distribution over the perceptron weights, which puts a smaller weight on the more complex features. If the prior distribution is a multivariate Gaussian (in the simplest case, just a product of univariate ones), the activation function of the single layer perceptron becomes a Gaussian process. Although a derivation of covariance functions based on a limiting process of multilayer networks is possible [8], one often simply uses a parameterized covariance functions instead. Besides the simple fact that any kernel function used in the SVM approach can be used as a covariance function of the Gaussian process approach and vice versa, there are more striking mathematical relations between the two approaches as we will discuss in following.

This chapter deals with two subjects. First, we will show how SVM can be understood as the maximum posterior (MAP) prediction from GP using a certain non-normalized likelihood. The second part deals with two approximation techniques that are useful in performing calculations for SVM or GP which would otherwise be intractable or time consuming. We will discuss a linear response method to derive an approximate leave-one-out estimator for the generalization error of SVM. Mean field methods (which have been originally developed within statistical mechanics) can be used to cope with posterior averages for GP which are not analytically tractable.

The rest of the chapter is organized as follows. Section 2 reviews the Gaussian process approach to noise-free classification. In section 3, we discuss how to extend this to modeling with noise. Section 4 deals with the relation of SVM to the maximum posterior prediction of GP. In section 5, we derive a leave-one-out estimator for the generalization error using linear response theory and a (mean field) assumption. Section 6 reviews the 'naive' mean field approach to Gaussian process classification. SVM and the naive mean field algorithm are compared in simulations in section 7. The chapter is concluded in section 8.

## 2 Gaussian Process Classification

Gaussian processes give a natural formulation of Bayesian learning in terms of prior distributions over functions. Here, we give a short summary of the basic concepts of Bayesian learning as applied to Gaussian Processes.

We consider a binary classifier with output $\mathrm{sgn} h(\mathbf{x})$, where $h(\mathbf{x})$ called (using neural network terminology) the 'activation' at input point $\mathbf{x}$. In a Bayesian approach, all information about $h(\mathbf{x})$, when example data are known, is encoded in a posterior distribution of activations functions. The first ingredient to such an approach is the Likelihood of $h(\mathbf{x})$ which for noise-free classification and output label $y$ is

$$p(y|h(\mathbf{x})) = \Theta(\, y\, h(\mathbf{x})) = \begin{cases} 1 & y\, h(\mathbf{x}) > 0 \\ 0 & y\, h(\mathbf{x}) < 0 \end{cases} \quad . \tag{1}$$

The second ingredient needed to form the posterior is the prior distribution over activations. A simple choice is a Gaussian process prior. This means that any finite set of function values

$$\mathbf{h} = (h(\mathbf{x}_1), \ldots, h(\mathbf{x}_m))$$

at arbitrary points $\mathbf{x}_1, \ldots, \mathbf{x}_m$ of the input space have a joint Gaussian distribution

$$p(\mathbf{h}) = \frac{1}{\sqrt{(2\pi)^m \det \mathbf{C}}} e^{-\frac{1}{2}(\mathbf{h}-\mathbf{m})^T \mathbf{C}^{-1}(\mathbf{h}-\mathbf{m})}$$

where $\mathbf{m} = (m(\mathbf{x}_1), \ldots, m(\mathbf{x}_m))$ is the mean and

$$\mathbf{C} \equiv E(\mathbf{h}\mathbf{h}^T) - \mathbf{m}\mathbf{m}^T$$

is the *covariance matrix* having elements

$$C(\mathbf{x}_i, \mathbf{x}_j), \quad i, j \in 1, \ldots, m \ .$$

The so-called *covariance function*, $C(\mathbf{x}, \mathbf{x}')$ is an explicit function of the pair of input points and determines correlations of activations at different points. A popular choice is the radial basis covariance function

$$C(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_{l=1}^{N} w_l (x_l - x_l')^2\right) \tag{2}$$

where the hyperparameter $w_l$ determines the length-scale of the $l$th component of the input vector. The covariance functions reflects our prior beliefs about the variability of the function $h(\mathbf{x})$. The mean function $m(\mathbf{x})$ is usually set to a constant. The covariance function is completely equivalent to the Kernel function in the SVM approach as will be shown below.

## 2.1 Statistical inference for Gaussian Processes

Given the training set

$$D_m = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, m\},$$

the inference task is to predict the correct label $y$ on a new point $\mathbf{x}$. In the Bayesian framework, this is done by using the posterior distribution of $h(\mathbf{x})$ (which in the following will also be abbreviated by $h$). To calculate the posterior, the new activation is included in the prior: $p(\mathbf{h}, h(\mathbf{x}))$. The posterior is then given by

$$p(\mathbf{h}, h(\mathbf{x})|\mathbf{y}) = \frac{1}{Z} \underbrace{p(\mathbf{y}|\mathbf{h})}_{\text{Likelihood}} \underbrace{p(\mathbf{h}, h(\mathbf{x}))}_{\text{Prior}} \ ,$$

where we have denoted the training set outputs by $\mathbf{y} = y_1, \ldots, y_m$ and the Likelihood of the training set activations is

$$p(\mathbf{y}|\mathbf{h}) = \prod_{i=1}^{m} p(y_i|h(\mathbf{x}_i)) = \prod_{i=1}^{m} \Theta(y_i\, h(\mathbf{x}_i)) \ .$$

Finally the normalization constant is

$$Z = \int d\mathbf{h} p(\mathbf{y}|\mathbf{h})\, p(\mathbf{h}) \ .$$

The *predictive distribution* is

$$p(h(\mathbf{x})|\mathbf{y}) = \int d\mathbf{h} p(\mathbf{h}, h(\mathbf{x})|\mathbf{y}) \ .$$

Using this distribution we can calculate the probability for output $y$: $p(y|\mathbf{y}) = \int dh\ p(y|h)p(h|\mathbf{y})$. In the ideal case, (Bayes) optimal predictions are obtained by choosing the output with highest probability. For binary $\pm1$-classification, the Bayes classifier may be written as

$$y^{\mathrm{Bayes}}(D_m, \mathbf{x}) = \mathrm{sgn} \int dh\, p(h|\mathbf{y})\, \mathrm{sgn}\, h \ .$$

The mean field approach–discussed in section 6–aims at calculating an approximation to the Bayes classifier.

# 3 Modeling the Noise

So far we have only considered noise-free classification. In real situations, noise or ambiguities will almost always be present and can–in the Bayesian framework–be modeled conceptually straightforwardly.

We will consider two noise models: 'input' (or additive) noise and output (multiplicative) noise. Input noise is defined as a random term added to the activation function in the likelihood:

$$p(y|h(\mathbf{x}), \xi(\mathbf{x})) = \Theta(y\,(h(\mathbf{x}) + \xi(\mathbf{x}))\,) \tag{3}$$

The output noise is flip noise, i.e.

$$p(y|h(\mathbf{x}), \xi(\mathbf{x})) = \Theta(y\, \xi(\mathbf{x})\, h(\mathbf{x})) \tag{4}$$

where $\xi \in \{-1, +1\}$.

There are two ways to incorporate the input noise in the Gaussian Process framework: either to average it out by directly modifying the Likelihood according to

$$p(y|h) = \int d\xi \, p(y|h, \xi) p(\xi)$$

or to change variables to the 'noisy' process $h + \xi$ with a modified prior and unchanged Likelihood eq. (1).

The simplest example is Gaussian noise with zero mean and variance $v$: The first approach gives the new Likelihood

$$p(y|h) = \Phi\left(\frac{yh}{\sqrt{v}}\right),$$

where $\Phi(x) = \int_{-\infty}^{x} \frac{dy}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}$ is an error-function. In the second approach, we use the fact that the process $h+\xi$–due to the Gaussianity of the noise–is also a Gaussian process with the following covariance matrix

$$\mathbf{C}^{\text{noisy}} = \mathbf{E}\left[(\mathbf{h} + \boldsymbol{\xi})(\mathbf{h} + \boldsymbol{\xi})^T\right] - \mathbf{E}\left[\mathbf{h} + \boldsymbol{\xi}\right] \mathbf{E}\left[(\mathbf{h} + \boldsymbol{\xi})^T\right] = \mathbf{C} + v\mathbf{I} \ .$$

For output noise, we take an iid flip process which flips the classification label with a probability given by $\kappa$, thus

$$\begin{aligned} p(y|h) &= \sum_{\xi=\pm 1} p(\xi) p(y|h, \xi) \\ &= \kappa \Theta(-yh) + (1 - \kappa) \Theta(yh) \\ &= \kappa + (1 - 2\kappa) \Theta(yh) \ . \end{aligned} \tag{5}$$

Such a noise process could model the effects of outliers, i.e. examples which are wrongly classified independently of the corresponding value of the activation function.

In the following, we will show how SVM can be obtained from Gaussian processes with a modified (non-normalized) Likelihood. The realization of the noise $\xi$ corresponds to the slack variable in the SVM approach.

# 4 From Gaussian Processes to SVM

We will start by discussing the additive noise model and in the end of this section shortly consider the multiplicative noise model.

To obtain support vector machines from Gaussian processes, we may first look at the maximum a posteriori (MAP) values for activations and noise variables which can be obtained by maximizing the joint distribution

$$p(\mathbf{y}, \boldsymbol{\xi}, \mathbf{h}) = \prod_i \left[ p(y_i|h_i, \xi_i) p(\xi_i) \right] p(\mathbf{h}) \ ,$$

where we have suppressed the explicit $\mathbf{x}$ dependence. Equivalently, we may minimize the negative log posterior, $L = -\log p(\mathbf{y}, \boldsymbol{\xi}, \mathbf{h})$. Shifting the activation variables to a zero mean Gaussian process, i.e. $h(\mathbf{x}) \to h(\mathbf{x}) + m(\mathbf{x})$ with constant mean $m(\mathbf{x}) = b$ and enforcing the inequality constraints of the Likelihood $p(y|h, \xi) = \Theta(y(h + b + \xi))$ by non-negative Lagrange multipliers $\boldsymbol{\alpha}$, we arrive at

$$L = -\sum_i \log p(\xi_i) - \log p(\mathbf{h}) - \sum_i \alpha_i \left[ y_i(h_i + b + \xi_i) \right] \ .$$

The MAP-parameters are obtained from the saddlepoint of $L$. A straightforward optimization $\frac{\partial L}{\partial h_i} = 0$ leads to the well known SVM expression

$$h_i^{\mathrm{SVM}} = \sum_j C_{ij} y_j \alpha_j$$

and the MAP prediction is given by

$$y^{\mathrm{SVM}}(\mathbf{x}) = \mathrm{sgn}\Big(\sum_j C(\mathbf{x}, \mathbf{x}_j) y_j \alpha_j + b\Big) \ .$$

Unfortunately, if the noise distribution has zero mean, the variation with respect to the other variables gives the trivial solution $\mathbf{h} = \boldsymbol{\alpha} = 0$. To obtain the SVM solution, a further *ad hoc* modification (equivalent to the introduction of a margin) is necessary. The final expression reads

$$L = -\sum_i \log p(\xi_i) - \log p(\mathbf{h}) - \sum_i \alpha_i \left[ y_i(h_i + b + \xi_i) - 1 \right] \ .$$

The expression for $\alpha_i$ and $\xi_i$ obtained by a variation of this expression depends explicitly on the noise model. For Laplace noise $p(\xi) = \frac{C}{2} \exp(-C|\xi|)$, we obtain the Kuhn-Tucker conditions corresponding to the linear slack penalty $C \sum_i \xi_i$ (with $\xi_i \geq 0$) and Gaussian noise leads to the Kuhn-Tucker conditions corresponding to the quadratic slack penalty $\frac{1}{2v} \sum_i \xi_i^2$ [2], Note that the mean of the Gaussian process $b$ plays the role of the threshold (or bias) in the SVM framework.[1]

The *ad hoc* introduction of the extra margin destroys the probabilistic interpretation of the corresponding 'Likelihood' $p(y|h,\xi) = \Theta(y(h+b+\xi)-1)$ which does not correspond to a true probability, because it is not normalized, i.e. $\sum_{y=\pm 1} p(y|h,\xi) \leq 1$. Hence, a direct Bayesian probabilistic interpretation of SVM is not fully possible (at least in the simple MAP approach that we have sketched). In practice however, it turns out that often the predictions made by both approaches are very similar when the same covariance function(kernel) and and noise(slack) model are used.

It is *not* possible to follow the same scheme for the *output* noise realization $\xi = \pm 1$ because this leads to a combinatorial optimization problem which cannot be solved easily. Alternatively, one could use the Likelihood eq. (5) where the noise realization has been averaged out. However, eq. (5) is not a 0-1 probability corresponding to a simple inequality constraint that in the optimization may be enforced using a Lagrange multiplier. For inference with Gaussian processes–on the other hand–this is not a problem, since formally and practically, it is straightforward to deal with the Likelihood eq. (5) as we will see in section 6.

# 5 Leave-One-Out Estimator

In this section, we derive an approximate leave-one-out estimator for the generalization error of the SVM-classifier. Previously, we have given a derivation based on a limiting procedure of the TAP-mean field equations [6]. The derivation given here is based on a linear response approach which is similar to the one derived by [7], however for a different loss function. For a similar approach in the framework of neural networks, see [4]. The approximation made in this approach is similar to an assumption which is also hidden in mean field theories: For systems which are composed of a large number of interacting degrees of freedom, a perturbation of a single one of them will change the remaining ones only slightly. To keep the derivation as simple as possible, we consider zero bias, $b$. At the end of this section, we briefly sketch how to generalize the result to $b \neq 0$.

[1] It is also possible to include a (e.g Gaussian) prior over $b$. The usual choice for SVM corresponds to a flat prior.

The basic idea is to calculate the change of the solution $h_i$ for input $i$ in response to removing example $k$. We will denote the solution at $i$ without the $k$th example by $h_i^{\backslash k}$. Before and after the removal of example $k$, we have the following solutions

$$h_i = \sum_j C_{ij}\, y_j\, \alpha_j \tag{6}$$

$$h_i^{\backslash k} = \sum_{j \neq k} C_{ij}\, y_j\, \alpha_j^{\backslash k} \tag{7}$$

or

$$\delta h_i = \delta h_i^{\backslash k} \equiv h_i^{\backslash k} - h_i = \sum_{j \neq k} C_{ij}\, y_j\, \delta\alpha_j - C_{ik}\, y_k\, \alpha_k \ . \tag{8}$$

There are two basic contributions to the change $\delta h_i$. The first term above is the indirect change due to the change of $\alpha_j$ in response to removing $k$ and the second term is the direct contribution. The leave-one-out error is obtained as a simple error count

$$\epsilon_{\mathrm{loo}}^{\mathrm{SVM}} = \frac{1}{m} \sum_i \Theta\left(-y_i h_i^{\backslash i}\right) \ . \tag{9}$$

Unfortunately, the first term in eq. (8) cannot be calculated without making a specific approximation. The following derivation is for the SVM framework with linear slack penalty.

The Kuhn-Tucker conditions of SVM learning distinguishes between three different groups of examples. We make the assumption that example $j \in 1, \ldots, k-1, k+1, \ldots, m$, remains in the same group after retraining the SVM when example $k (\neq j)$ is removed. Explicitly,

1. Non-support vectors ($y_j h_j > 1$ and $\alpha_j = 0$), will remain non-support vectors: $\delta\alpha_j = 0$.

2. Margin support vectors ($y_j h_j = 1$ and $\alpha_j \in [0, C]$), will remain margin support vectors: $\delta h_j = 0$.

3. Misclassified patterns ($y_j h_j < 1$ and $\alpha_j = C$), will remain misclassified patterns: $\delta\alpha_j = 0$.

It is easy to construct a set of examples for which this assumption is not valid. We expect the approximation to be typically quite good when the number of support vectors is large because then upon removal of a support vector, the new solution will mainly be expressed as a (small) reorganization of the remaining margin support vectors. With this simplifying assumption, we may now solve eq. (8) in the form

$$\sum_{j \neq k}^{\mathrm{mSV}} C_{ij}\, y_j\, \delta\alpha_j - C_{ik}\, y_k\, \alpha_k = 0 \tag{10}$$

to find $\delta\alpha_j$ for the margin support vectors (the non-support vectors and misclassified patterns are assumed to have $\delta\alpha_j = 0$).

It is necessary to consider explicitly the group to which the removed example belongs. We see immediately that if example $k$ is a non-support vector then $\delta\alpha_j = 0$. If example $k$ is a margin support vector, we get

$$\delta\alpha_i = \sum_{j \neq k}^{\mathrm{mSV}} \left[ (\mathbf{C}_{\mathrm{mSV}}^{\setminus k})^{-1} \right]_{ij} C_{jk}\, y_k\, \alpha_k \ ,$$

where $\mathbf{C}_{\mathrm{mSV}}^{\setminus k}$ is the covariance matrix of the margin support sector patterns excluding the $k$th pattern. Inserting the result in $\delta h_i$ and setting $k = i$, we find

$$\begin{aligned} \delta h_i &= \left\{ \sum_{j,j' \neq i}^{\mathrm{mSV}} C_{ij} \left[ (\mathbf{C}_{\mathrm{mSV}}^{\setminus k})^{-1} \right]_{jj'} C_{j'i} - C_{ii} \right\} y_i\, \alpha_i \\ &= -\frac{1}{\left[ \mathbf{C}_{\mathrm{mSV}}^{-1} \right]_{ii}} y_i\, \alpha_i \ , \end{aligned} \tag{11}$$

where $\mathbf{C}_{\mathrm{mSV}}$ is the covariance matrix of the margin support vector inputs. In the last equality a matrix identity for the partitioned inverse matrix has been used.

For example $k$ being a misclassified pattern, the sum in eq. (10) runs over all margin support vectors, thus

$$\delta\alpha_i = \sum_{j}^{\mathrm{mSV}} \left[ \mathbf{C}_{\mathrm{mSV}}^{-1} \right]_{ij} C_{jk}\, y_k\, \alpha_k \ ,$$

and

$$\delta h_i = \left\{ \sum_{j,j'}^{\mathrm{mSV}} C_{ij} \left[ \mathbf{C}_{\mathrm{mSV}}^{-1} \right]_{jj'} C_{j'i} - C_{ii} \right\} y_i\, \alpha_i \ . \tag{12}$$

We see that the reaction $\delta h_i$ is proportional to a direct change term through the factor $\alpha_i$. We have now obtained the leave-one-out estimator eq. (9) for SVM with $y_i\, h_i^{\setminus i} = y_i\, h_i + y_i\, \delta h_i$ and $\delta h_i$ given by eqs. (11) and (12) for respectively margin support vectors and misclassified patterns. Note that the sum over patterns will only run over support vectors since the reaction is estimated to be zero for non-support vectors.[2]

One may argue that it is computationally expensive to invert $\mathbf{C}_{\mathrm{mSV}}$. However, we expect that the computational cost of this operation is comparable to finding the solution to the optimization problem since it–on top of identifying the support vectors–also requires the inverse of $\mathbf{C}_{\mathrm{SV}}$. This is also observed in simulations. Using this leave-one-out estimator is thus much cheaper than the exact leave-one-out estimate that requires running the algorithm $N$ times (although each run will probably only take a few iterations if one uses an iterative learning scheme like the Adatron algorithm [1] with the full training set solution as the starting point). Another possible way of decreasing the computational complexity of the estimator is to use methods in the spirit of Wahba's randomized GACV [7].

# 6 Naive Mean Field Algorithm

The aim of the mean field approach is to compute an approximation to the Bayes prediction $y^{\mathrm{Bayes}}(\mathbf{x}) = \langle \mathrm{sgn} h(x) \rangle$ for the GP classifier, where we have introduced the notation $\langle \ldots \rangle$ to denote a posterior average. We will only discuss a 'naive' mean field algorithm with the aim of stressing the similarities and differences between the SVM and Gaussian process approach. We will follow the derivation given in [6] based on the so-called Callen identity. An independent derivation is given in [5].

We will use the simplified prediction $y(\mathbf{x}) = \mathrm{sgn}\langle h(\mathbf{x}) \rangle$ which the Bayes

[2] Generalizing this to non-zero threshold means having to substitute $h_i$ with $h_i + b$. The Kuhn-Tucker condition for the margin support vectors therefore change to $y_i(h_i + b_i) = 1$ which implies $\delta h_i = -\delta b$. E.g. for $k$ being a margin support vector, we now have

$$\delta\alpha_i = \sum_{j \neq k}^{\mathrm{mSV}} \left[ (\mathbf{C}_{\mathrm{mSV}}^{\setminus k})^{-1} \right]_{ij} (C_{jk}\, y_k\, \alpha_k - \delta b) \ .$$

The saddlepoint condition for b, $\frac{\partial L}{\partial b} = 0$, gives $\sum_i y_i \alpha_i = 0$. This condition implies $\sum_i^{\mathrm{mSV}} y_i \delta\alpha_i = 0$ which together with the expression for $\delta\alpha_i$ above determines $\delta b$.

classifier reduces to when the posterior is symmetric around its mean. We first give exact expressions for the posterior $\langle h(\mathbf{x})\rangle$ which can be proved using integration by parts

$$\langle h(\mathbf{x})\rangle = \sum_{i=1}^{m} C(\mathbf{x},\mathbf{x}_i)\, y_i\, \alpha_i \ . \tag{13}$$

One observes immediately that this corresponds exactly to the SVM expression. However in this case, the 'embedding strength' $\alpha_i$ of example $i$ is given by

$$\alpha_i = \frac{y_i}{Z}\int d\mathbf{h} p(\mathbf{h}) \frac{\partial p(y_i|h_i)}{\partial h_i} \prod_{j\neq i} p(y_j|h_j) \tag{14}$$

The $\alpha_i$'s will always be non-negative when $p(y_i|h(\mathbf{x}_i))$ is an increasing function of $y_i h(\mathbf{x}_i)$.

We give now a mean field argument for the approximate computation of the $\alpha_i$. There are different ways of defining a mean field theory. The present one has the advantage over other approaches [6], that no matrix inversions are needed in the final algorithm. To proceed, auxiliary variables $\mathbf{t}$ are introduced using a standard Gaussian transformation

$$\begin{aligned}\alpha_i &= \frac{y_i}{Z}\int d\mathbf{h} d\mathbf{t} \exp\left(-\frac{1}{2}\mathbf{t}^T\mathbf{C}\mathbf{t} + i\mathbf{t}^T\mathbf{h}\right) \frac{\partial p(y_i|h_i)}{\partial h_i} \prod_{j\neq i} p(y_j|h_j) \qquad (15)\\ &= \frac{y_i}{Z}\int d\mathbf{h} d\mathbf{t} (-it_i) \exp\left(-\frac{1}{2}\mathbf{t}^T\mathbf{C}\mathbf{t} + i\mathbf{t}^T\mathbf{h}\right) \prod_{j} p(y_j|h_j) = -iy_i\langle it_i\rangle \ .\end{aligned}$$

In the second equality integration by parts is applied. In the last equality the bracket is understood as a formal average over the joint complex measure of the variables $\mathbf{h}$ and $\mathbf{t}$. Next, we separate the integrations over $h_i$ and $t_i$ from the rest of the variables to get

$$\alpha_i = y_i \left\langle \frac{\int dh_i dt_i \exp\left(-\frac{1}{2}C_{ii}(t_i)^2 + (-it_i)(\sum_{j\neq i} C_{ij}(-it_j) - h_i)\right) \frac{\partial p(y_i|h_i)}{\partial h_i}}{\int dh_i dt_i \exp\left(-\frac{1}{2}C_{ii}(t_i)^2 + (-it_i)(\sum_{j\neq i} C_{ij}(-it_j) - h_i)\right) p(y_i|h_i)} \right\rangle \ . \tag{16}$$

This identity can be proved by noting that the average over $h_i$ and $t_i$ in $\langle\ldots\rangle$ exactly cancels the denominator given us back the original expression for $\alpha_i$.

We may now carry out the explicitly written integrals over $h_i$ and $t_i$.

Using the Likelihood for output noise eq. (5), we find

$$
\begin{aligned}
\alpha_i &= y_i \left\langle \frac{\int dh_i \exp\left(-\frac{(h_i - \sum_{j\neq i} C_{ij}(-it_j)))^2}{2C_{ii}}\right) \frac{\partial p(y_i|h_i)}{\partial h_i}}{\int dh_i \exp\left(-\frac{(h_i - \sum_{j\neq i} C_{ij}(-it_j))^2}{2C_{ii}}\right) p(y_i|h_i)} \right\rangle \\
&= \left\langle \frac{(1-2\kappa) D\left(\frac{\sum_{j\neq i} C_{ij}(-it_j)}{\sqrt{C_{ii}}}\right)}{\kappa + (1-2\kappa)\Phi\left(y_i \frac{\sum_{j\neq i} C_{ij}(-it_j)}{\sqrt{C_{ii}}}\right)} \right\rangle , \qquad (17)
\end{aligned}
$$

where $D(z) = e^{-z^2/2}/\sqrt{2\pi}$ is the Gaussian measure. So far everything is exact. The 'naive' mean field approximation amounts to neglecting the fluctuations of the field $\sum_{j\neq i} C_{ij}(-it_j)$ and moving the expectation through the nonlinearities to get a self-consistent set of equations for $\alpha_i = -iy_i\langle t_i \rangle$. The explicit expression for $\alpha_i$ becomes

$$
\alpha_i = \frac{1}{\sqrt{C_{ii}}} \frac{(1-2\kappa) D(z_i)}{\kappa + (1-2\kappa)\Phi(z_i)}, \qquad z_i = y_i \frac{\langle h_i \rangle - C_{ii} y_i \alpha_i}{\sqrt{C_{ii}}} . \qquad (18)
$$

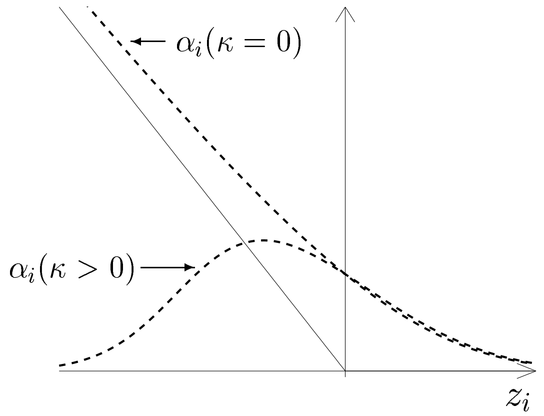


Figure 1: The 'embedding strength' $\alpha_i$ plotted as a function of $z_i$ with $C_{ii} = 1$.

In figure 1, $\alpha_i$ is plotted as function of $z_i$ (with $C_{ii} = 1$). The shape of the 'embedding'-function depends crucially upon whether we model with or without noise. For the noise-free case, $\kappa = 0$, $\alpha_i$ is a decreasing function of $y_i\langle h_i \rangle - C_{ii}\alpha_i = z_i\sqrt{C_{ii}}$ which may be thought of as a naive approximation to ($y$ times) the activation for input $i$ trained without the $i$th example. The result is intuitively appealing because it says that the harder the example is to 'predict' the larger weight, $\alpha_i$ should it have.[3] In the noisy case, $\alpha_i$ is a decreasing function of $z_i$ down to certain point at which the algorithm tends to consider the example as being corrupted by noise and consequently gives it a smaller weight. This illustrates the difference between flip noise and using the linear slack penalty for support vectors where the 'hardest' patterns are given the largest weight, $\alpha_i = C$.

It is interesting to note that $\alpha_i$, in contrast to SVM, is an explicit function of other variables of the algorithm. The fact that the function is non-linear makes it impossible to solve the equations analytically and we have to resort to numerical methods, e.g. iteration. We will show that despite the fact that this algorithm look very different from SVM, the solution obtained and the performance is quite similar. The mean field approach will tend to produce a smaller minimal margin, however we have not observed that this has any effect on performance.

For comparison, we also give the leave-one-out estimator for the naive mean field algorithm. It is derived from the mean field equations using linear response theory [6] in completely the same fashion as the leave-one-out estimator for SVM

$$\epsilon_{\mathrm{loo}}^{\mathrm{naive}} = \frac{1}{m} \sum_{i}^{\mathrm{SV}} \Theta \left( -y_i \langle h_i \rangle + \left\{ \frac{1}{[(\mathbf{\Omega} + \mathbf{C})^{-1}]_{ii}} - \Omega_i \right\} \alpha_i \right) , \tag{19}$$

where $\mathbf{\Omega}$ is a diagonal matrix with elements

$$\Omega_i = C_{ii} \left( \frac{1}{y_i \alpha_i \langle h_i \rangle} - 1 \right) .$$

We thus have the same basic structure as for the SVM estimator. However, this estimator requires the inversion of the full covariance matrix. In the next section, we will demonstrate on a benchmark dataset that the leave-one-out estimators are in very good agreement with the exact leave-one-out errors. This has also been observed previously on other benchmarks [5, 6].

[3] In the more advanced TAP mean field theory $z_i$ is proportional to the 'unlearned' mean activation [6].

# 7 Simulation Results

The two algorithms have been tested on the *Wisconsin breast cancer* dataset, which is a binary classification task (tumor is malignant or benign) based on 9 attributes, see e.g. [3]. We have removed the 16 examples with missing values and used standard preprocessing as to for every input setting the mean to zero and the variance to unity across the dataset of 683 examples. The performance is–as in previous studies–accessed using 10-fold cross validation [3].

For SVM, we used the parallel version of the Adatron algorithm [1] which extended to general covariance functions has turned out to be a fast iterative algorithm [3]. For naive mean field theory, we solved the mean field equations using iteration [6].

We choose to work with the radial basis covariance function eq. (2) and to reduce the number of hyperparameters, we have set all inputs to a common length-scale $w_l = \frac{1}{\sigma^2 N}$. The Gaussian noise model is used in noisy process formulation thus adding the input noise variance $v$ to the diagonal of the covariance matrix. For the mean field algorithm, we have the additional output noise parameter $\kappa$. These two(three) parameters are chosen as to minimize the leave-one-out (loo) error for one of the 10 training sets by scanning through a number of parameter values. We use the values $\sigma^2 = 0.15$ and $v = 1.3$ for both algorithms and $\kappa = 0$. The true minimum is probably not found by this very rough procedure, however the performance turned out to be quite insensitive to the choice of hyperparameters.

Since we use the training set to access the performance through the 10-fold cross validation scheme, the loo estimate is not independent of the test error. However, our main emphasis is not on generalization performance but rather on the learning speed and on the precision of the loo estimators. The 10-fold cross validation error for respectively SVM and naive mean field theory is $\epsilon = 0.0307$ (21) and $\epsilon = 0.0293$ (20), where the numbers in parentheses indicates the number of misclassifications. The loo errors are $\epsilon_{\mathrm{loo}} = 0.0293$ and $\epsilon_{\mathrm{loo}} = 0.0270$. The more advanced TAP mean field algorithm [5, 6] finds a solution very similar to the one of the naive mean field algorithm. In another study using the SVM-algorithm, [3] find $\epsilon = 0.0052$. The difference may be due to a number of reasons: the different splitting of the data set, a different way of parameter selection, use of bias and/or handling of missing values. With other methods such as multi-layer neural networks, linear discriminant, RBF neural networks and CART, the error is found to be in the range $\epsilon =$

Table 1: Results for the Wisconsin dataset.

| Algorithm | $\epsilon_{\rm loo}^{\rm exact}$ | $\epsilon_{\rm loo}$ | CPU w. loo | CPU wo. loo | It. |
|---|---|---|---|---|---|
| SVM | 0.0261 | 0.0261 | 5 | 4 | 195 |
| Naive Mean Field | 0.0293 | 0.0293 | 16 | 1 | 31 |

0.034 to $\epsilon = 0.058$.

In table 1, we compare the learning speed of the two algorithms–trained on one of the 10 training sets (with 614 examples)–both with and without evaluating the loo estimator (in CPU seconds on an Alpha 433au) and the number of iterations required to achieve the same precision ($10^{-5}$). We also compare the leave-one-out estimator $\epsilon_{\rm loo}$ with the exact loo estimator $\epsilon_{\rm loo}^{\rm exact}$ for both algorithms. In this case the loo estimators for both algorithms are in accordance with the exact values. Apart from the case where the value of $\sigma$ is very small corresponding closely to a nearest-neighbor classifier, we have always observed that the leave-one-out estimators are very precise, deviating at most one classification from the correct value [6].

Without evaluating the loo estimators, the naive mean field algorithm is about 4 times faster than the Adatron algorithm. With the leave-one-out estimator, the SVM is about 4 times faster than the naive mean field algorithm. This is due to the fact that for $\epsilon_{\rm loo}^{\rm SVM}$, eq. (9), we only need to invert the covariance matrix for the margin support vector examples, which in this example is 272-dimensional, whereas $\epsilon_{\rm loo}^{\rm naive}$, eq. (19) requires the inversion of the full covariance matrix (614-dimensional). If the linear slack penalty had been used, the number of support vectors would have been smaller and the advantage of using $\epsilon_{\rm loo}^{\rm SVM}$ would have been even greater.

In figure 2, we compare the solutions found by the two algorithms. The solutions for the 'embedding strengths' $\alpha_i$ are quite similar. However, the small differences in embedding strength give rise to different distributions of margins. The mean field algorithm achieves both smaller and larger margins than SVM. We have also indicated which of the examples are predicted as wrongly classified by the loo estimators. Interestingly, these are almost exclusively all the examples with the highest $\alpha_i$ starting around the point where the $\alpha_i$-curve's slope increases. This observation suggests that a heuristic cut-off for small $\alpha_i$ could be introduced to make the loo estimators faster without significantly deteriorating the quality of the estimators. Simple heuristics could be developed like e.g. only considering the covariance matrix for the 10% of the examples with highest $\alpha_i$, if one expects the error rate to be around 5%.

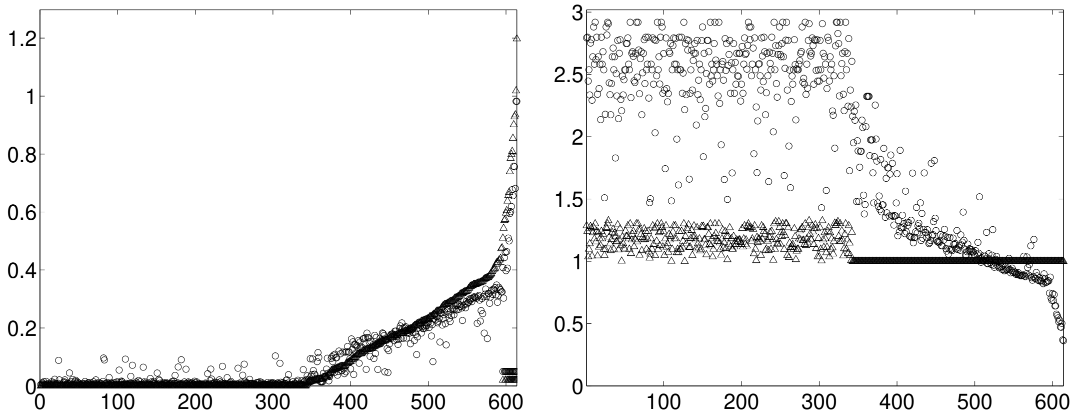


Figure 2: Left figure: The 'embedding strengths' $\alpha_i$ for each example. The right figure: The margins $y_i h_i$ for SVM and $y_i \langle h_i \rangle$ for naive mean field theory (same ordering as the left plot). The triangles are for support vectors and circles are for naive mean field theory. They are sorted in ascending order according to their support vector $\alpha_i$ value and the naive mean field solution is rescaled to the length of the support vector solution. In the lower right corner of the left figure, it is indicated which examples contribute to the loo error.

# 8 Conclusion

This contribution discusses two aspects of classification with Gaussian Processes and Support Vector Machines (SVM). The first one deals with the relation between the two approaches. We show that the SVM can be derived as a maximum posterior prediction of a GP model. However, the corresponding likelihood is not normalized and a fully satisfactory probabilistic interpretation is not possible.

The second aspect deals with approximate approaches for treating two different computational problems arising in GP and SVM learning. We show how to derive an approximate leave-one-out estimator for the generalization error for SVM using linear response theory. This estimator requires only the inversion of the covariance matrix of the margin support vector examples.

As the second problem we discuss the computation of the Bayes prediction for a GP classifier. We give a derivation of an algorithm based on a 'naive' mean field method. The leave-one-out estimator for this algorithm requires the inversion of the covariance matrix for the *whole* training set. This underlines a difference between SVM and GP which may have important practical consequences when working with large data sets: the GP solution lacks the sparseness property of SVM.

We have presented simulations for the Wisconsin breast cancer dataset, with the model hyperparameters determined by minimizing the approximate leave-one-out estimator. The performance of both algorithms was found to be very similar. The approximate leave-one-out estimators were in perfect agreement with the exact leave-one-out estimators.

An important problem for future research is to find efficient ways for tuning a larger number of hyperparameters in the kernel automatically. This will be necessary e.g., in order to adapt the length-scales of the input components individually. The minimization of a leave-one-out estimator is only one possible technique for finding reasonable values for such parameters. Bayesian approaches to model selection such as the evidence (or MLII) method could be interesting alternatives. They are obviously well suited for the Bayesian GP approach. But they may also be interesting for an application to SVM. However, in order to implement such approaches properly, it will be necessary to understand the quantitative relations and differences between GP and SVM in more detail.


### Acknowledgments

We are thankful to Thilo-Thomas Frieß, Pál Ruján, Sara A. Solla, Peter Sollich and Grace Wahba for discussions. This research is supported by the Swedish Foundation for Strategic Research.


## References


[1] J. K. Anlauf and M. Biehl, The AdaTron: An Adaptive Perceptron Algorithm, Europhys. Lett. **10**, 687 (1989).

[2] C. Cortes and V. N. Vapnik, Support Vector Machines, Machine Learning **20**, 1-25 (1995).

[3] T.-T. Frieß, N. Cristianini and C. Campbell, The Kernel Adatron Algorithm: A Fast and Simple Learning Procedure for Support Vector Machines. in *15th Intl. Conf. Machine Learning*, Morgan Kaufman Publishers (1998).

[4] J. Larsen and L. K. Hansen, Linear Unlearning for Cross-Validation, Advances in Computational Mathematics, **5**, 269-280 (1996).

[5] M. Opper and O. Winther, Mean Field Methods for Classification with Gaussian Processes, in *Advances in Neural Information Processing Systems 11 (NIPS'98)*, M. S. Kearns, S. A. Solla, and D. A. Cohn, eds., MIT Press, Cambridge, MA (1999).

[6] M. Opper and O. Winther, Gaussian Processes for Classification, submitted to Neural Computation (1999).

[7] G. Wahba, Support Vector Machines, Reproducing Kernel Hilbert Spaces and Randomized GACV, in Advances in Kernel Methods: Support Vector Learning, B. Schölkopf, C. J. C. Burges and A. J. Smola, editors, The MIT Press (1998).

[8] C. K. I. Williams, Computing with Infinite Networks, in *Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan and T. Petsche, eds., 295-301. MIT Press, Cambridge, MA (1997).