
Gaussian Processes and SVM: Mean Field and Leave-One-Out

Manfred Opper

*Department of Computer Science and Applied Mathematics,
Aston University
Birmingham B4 7ET
United Kingdom
m.opper@aston.ac.uk
<http://neural-server.aston.ac.uk/People/opperm/>*

Ole Winther

*Theoretical Physics II, Lund University
Sölvegatan 14 A
S-223 62 Lund
Sweden
winther@thep.lu.se
<http://www.thep.lu.se/tf2/staff/winther/>*

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.

17.1 Introduction

It is well-known that Gaussian Processes (GP) and Support Vector Machines (SVM) are closely related, see, e.g., [Wahba, 1999b, Williams, 1998]. Both approaches are non-parametric. This means that they allow (at least for certain kernels) 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 the 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 (see Chapter 1 for details).

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 [Neal, 1996, Williams, 1997], one often simply uses a parameterized covariance function 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 a posteriori (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 17.2 reviews the Gaussian process approach to noise-free classification. In Section 17.3, we discuss how to extend this to modeling with noise. Section 17.4 deals with the relation of SVM to the maximum a posteriori prediction of GP. In Section 17.5, we derive a leave-one-out estimator for the generalization error using linear response theory and a (mean field) assumption. Section 17.6 reviews the “naive” mean field approach to Gaussian process classification. SVM and the naive mean field algorithm are compared in simulations in Section 17.7. The chapter is concluded in Section 17.8.

17.2 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.

Likelihood

We consider a binary classifier with output $g(\mathbf{x}) = \text{sgn}f(\mathbf{x})$, where $f(\mathbf{x})$ called (using neural network terminology) the “activation” at input point \mathbf{x} . In a Bayesian approach, all information about $f(\mathbf{x})$, 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 $f(\mathbf{x})$ which for noise-free classification and output label y is

$$p(y|f(\mathbf{x})) = \Theta(y f(\mathbf{x})) = \begin{cases} 1 & y f(\mathbf{x}) > 0 \\ 0 & y f(\mathbf{x}) < 0 \end{cases} . \quad (17.1)$$

Gaussian Process prior

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

$$\mathbf{f} = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_m)) \quad (17.2)$$

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

$$p(\mathbf{f}) = \frac{1}{\sqrt{(2\pi)^m \det \mathbf{k}}} e^{-\frac{1}{2}(\mathbf{f}-\mathbf{m})^T \mathbf{k}^{-1} (\mathbf{f}-\mathbf{m})} \quad (17.3)$$

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

$$\mathbf{k} \equiv E(\mathbf{f}\mathbf{f}^T) - \mathbf{m}\mathbf{m}^T \quad (17.4)$$

is the *covariance matrix* having elements

$$k(\mathbf{x}_i, \mathbf{x}_j), \quad i, j \in 1, \dots, m . \quad (17.5)$$

covariance function (kernel)

The so-called *covariance function*, $k(\mathbf{x}, \mathbf{x}')$ 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 eq. (1.73), but any function that gives rise to a positive semidefinite covariance matrix can be used. The covariance function reflects our prior beliefs about the variability of the function

$f(\mathbf{x})$. 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.

17.2.1 Statistical Inference for Gaussian Processes

Given the training set

$$D_m = \{(\mathbf{x}_i, y_i) | i = 1, \dots, m\}, \quad (17.6)$$

posterior

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 $f(\mathbf{x})$ (which in the following will also be abbreviated by f). To calculate the posterior, the new activation is included in the prior: $p(\mathbf{f}, f(\mathbf{x}))$. The posterior is then given by

$$p(\mathbf{f}, f(\mathbf{x}) | \mathbf{y}) = \frac{1}{p(\mathbf{y})} \underbrace{p(\mathbf{y} | \mathbf{f})}_{\text{Likelihood}} \underbrace{p(\mathbf{f}, f(\mathbf{x}))}_{\text{Prior}}, \quad (17.7)$$

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

$$p(\mathbf{y} | \mathbf{f}) = \prod_{i=1}^m p(y_i | f(\mathbf{x}_i)) = \prod_{i=1}^m \Theta(y_i f(\mathbf{x}_i)). \quad (17.8)$$

Finally the normalization constant is

$$p(\mathbf{y}) = \int d\mathbf{f} p(\mathbf{y} | \mathbf{f}) p(\mathbf{f}). \quad (17.9)$$

The *predictive distribution* is

$$p(f(\mathbf{x}) | \mathbf{y}) = \int d\mathbf{f} p(\mathbf{f}, f(\mathbf{x}) | \mathbf{y}). \quad (17.10)$$

Bayes optimal prediction

Using this distribution we can calculate the probability for output y : $p(y | \mathbf{y}) = \int df p(y | f) p(f | \mathbf{y})$. In the ideal case, (Bayes) optimal predictions are obtained by choosing the output with highest probability. For binary ± 1 -classification, the Bayes classifier may be written as

$$y^{\text{Bayes}}(D_m, \mathbf{x}) = \text{sgn} \int df p(f | \mathbf{y}) \text{sgn} f. \quad (17.11)$$

The mean field approach—discussed in Section 17.6—aims at calculating an approximation to the Bayes classifier.

17.3 Modeling the Noise

So far we have only considered noise-free classification. In real situations, noise or ambiguities will almost always be present and are—in the Bayesian framework—at least conceptually straightforward to model.

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:

$$p(y|f(\mathbf{x}), \xi(\mathbf{x})) = \Theta(y(f(\mathbf{x}) + \xi(\mathbf{x}))) \quad (17.12)$$

output noise The output noise is flip noise, i.e.,

$$p(y|f(\mathbf{x}), \xi(\mathbf{x})) = \Theta(y\xi(\mathbf{x})f(\mathbf{x})) \quad (17.13)$$

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

$$p(y|f) = \int d\xi p(y|f, \xi)p(\xi) \quad (17.14)$$

or to change variables to the “noisy” process $f + \xi$ with a modified prior and unchanged Likelihood eq. (17.1).

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

$$p(y|f) = \Phi\left(\frac{yf}{\sqrt{v}}\right), \quad (17.15)$$

where $\Phi(x) = \int_{-\infty}^x \frac{dy}{\sqrt{2\pi}} e^{-\frac{y^2}{2}}$ is an error-function. This Likelihood corresponds to *probit regression* [Neal, 1997]. In the second approach, we use the fact that the process $f + \xi$ —due to the Gaussianity of the noise—is also a Gaussian process with the following covariance matrix

$$\mathbf{k}^{\text{noisy}} = \mathbf{E}[(\mathbf{f} + \boldsymbol{\xi})(\mathbf{f} + \boldsymbol{\xi})^T] = \mathbf{E}[\mathbf{f} + \boldsymbol{\xi}]\mathbf{E}[(\mathbf{f} + \boldsymbol{\xi})^T] = \mathbf{k} + v\mathbf{I}. \quad (17.16)$$

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

$$\begin{aligned} p(y|f) &= \sum_{\xi=\pm 1} p(\xi)p(y|f, \xi) \\ &= \kappa\Theta(-yf) + (1 - \kappa)\Theta(yf) \\ &= \kappa + (1 - 2\kappa)\Theta(yf). \end{aligned} \quad (17.17)$$

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. Usually, we expect that the probability of a flip is small, when $f(\mathbf{x})$ is large and we have high confidence on the label. However, there may be some fraction of outliers in the data which may not be treated well by such a model. For those, we include the possibility that the probability of flip is independent of the location.

In the following, we will show 1. how SVM can be obtained from Gaussian processes with a modified (non-normalized) Likelihood and 2. the slack variable for SVM corresponds to the realization of the input noise ξ in the GP framework.

17.4 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

$$p(\mathbf{y}, \boldsymbol{\xi}, \mathbf{f}) = \prod_i [p(y_i | f_i, \xi_i) p(\xi_i)] p(\mathbf{f}) , \quad (17.18)$$

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

$$L = - \sum_i \log p(\xi_i) - \log p(\mathbf{f}) - \sum_i \alpha_i [y_i (f_i + b + \xi_i)] . \quad (17.19)$$

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

$$f_i^{\text{SVM}} = \sum_j k_{ij} y_j \alpha_j \quad (17.20)$$

and the MAP prediction is given by

$$y^{\text{SVM}}(\mathbf{x}) = \text{sgn}(\sum_j k(\mathbf{x}, \mathbf{x}_j) y_j \alpha_j + b) . \quad (17.21)$$

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

$$L = - \sum_i \log p(\xi_i) - \log p(\mathbf{f}) - \sum_i \alpha_i [y_i (f_i + b + \xi_i) - 1] . \quad (17.22)$$

The expression for α_i and ξ_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 $\frac{1}{2v} \sum_i \xi_i^2$ [Cortes and Vapnik, 1995], Note that the mean of the Gaussian process b plays the role of the threshold (or bias) in the SVM framework.¹

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 *ad hoc* introduction of the extra margin destroys the probabilistic interpretation of the corresponding 'Likelihood' $p(y|f, \xi) = \Theta(y(f + b + \xi) - 1)$ which does not correspond to a true probability, because it is not normalized, i.e., $\sum_{y=\pm 1} p(y|f, \xi) \leq 1$. Hence, a direct Bayesian probabilistic interpretation of SVM is not fully possible (at least in the simple MAP approach that we have sketched). So if we want to associate probabilities with output predictions, it is most natural to work in the Gaussian process framework (but see also Chapter 5). In practice however, it turns out that often the predictions made by both approaches are very similar when the same covariance function (kernel) 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. (17.17) where the noise realization has been averaged out. However, eq. (17.17) 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. (17.17) as we will see in Section 17.6.

17.5 Leave-One-Out Estimator

In this section, we derive an approximate leave-one-out (loo) estimator for the generalization error of the SVM-classifier. Although we do not know if our leave-one-out estimator can be cast into a bound on the true loo error (for bounds see [Jaakkola and Haussler, 1999b], Chapters 1 and 16), it seems to be at an excellent approximation (at least in the cases that we have applied it). Previously, we have given a derivation based on a limiting procedure of the TAP-mean field equations [Oppen and Winther, 1999a]. The derivation given here is based on a linear response approach which is similar to the one derived by Wahba [1999b], however for a different loss function. For a similar approach in the framework of neural networks, see [Larsen and Hansen, 1996]. 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, $b = 0$. At the end of this section, we briefly sketch how to generalize the result to $b \neq 0$.

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

$$f_i = \sum_j k_{ij} y_j \alpha_j \quad (17.23)$$

$$f_i^{\setminus l} = \sum_{j \neq l} k_{ij} y_j \alpha_j^{\setminus l} \quad (17.24)$$

or

$$\delta f_i = \delta f_i^{\setminus l} \equiv f_i^{\setminus l} - f_i = \sum_{j \neq l} k_{ij} y_j \delta \alpha_j - k_{il} y_l \alpha_l . \quad (17.25)$$

There are two basic contributions to the change δf_i . The first term above is the indirect change due to the change of α_j in response to removing l and the second term is the direct contribution. The leave-one-out error is obtained as a simple error count

$$\epsilon_{\text{loo}}^{\text{SVM}} = \frac{1}{m} \sum_i \Theta(-y_i f_i^{\setminus l}) . \quad (17.26)$$

Unfortunately, the first term in eq. (17.25) cannot be calculated without making a specific approximation. The following derivation is for the SVM framework with linear slack penalty.

leave-one-out
approximation

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

1. Non-support vectors ($y_j f_j > 1$ and $\alpha_j = 0$), will remain non-support vectors: $\delta \alpha_j = 0$.
2. Margin support vectors ($y_j f_j = 1$ and $\alpha_j \in [0, C]$), will remain margin support vectors: $\delta f_j = 0$.
3. Misclassified patterns ($y_j f_j < 1$ 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. (17.25) in the form

$$\sum_{j \neq l}^{\text{mSV}} k_{ij} y_j \delta \alpha_j - k_{il} y_l \alpha_l = 0 \quad (17.27)$$

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 l is a non-support vector then $\delta \alpha_j = 0$. If example l is a margin support vector, we get

$$\delta \alpha_i = \sum_{j \neq l}^{\text{mSV}} \left[(\mathbf{k}_{\text{mSV}}^{\setminus l})^{-1} \right]_{ij} k_{jl} y_l \alpha_l , \quad (17.28)$$

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

$$\delta f_i = \left\{ \sum_{j, j' \neq i}^{\text{mSV}} k_{ij} \left[(\mathbf{k}_{\text{mSV}}^{\setminus i})^{-1} \right]_{jj'} k_{j'i} - k_{ii} \right\} y_i \alpha_i = - \frac{1}{[\mathbf{k}_{\text{mSV}}^{-1}]_{ii}} y_i \alpha_i . \quad (17.29)$$

In the last equality a matrix identity for the partitioned inverse matrix has been used.

For example l being a misclassified pattern, the sum in eq. (17.27) runs over all margin support vectors, thus

$$\delta \alpha_i = \sum_j^{\text{mSV}} [\mathbf{k}_{\text{mSV}}^{-1}]_{ij} k_{jl} y_l \alpha_l , \quad (17.30)$$

and

$$\delta f_i = \left\{ \sum_{j, j'}^{\text{mSV}} k_{ij} [\mathbf{k}_{\text{mSV}}^{-1}]_{jj'} k_{j'i} - k_{ii} \right\} y_i \alpha_i . \quad (17.31)$$

We see that the reaction δf_i is proportional to a direct change term through the factor α_i . We have now obtained the leave-one-out estimator eq. (17.26) for SVM with $y_i f_i^{\setminus i} = y_i f_i + y_i \delta f_i$ and δf_i given by eqs. (17.29) and (17.31) 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.

One may argue that it is computationally expensive to invert \mathbf{k}_{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{k}_{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 [Anlauf and Biehl, 1989] 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 the randomized GACV by Wahba [1999b].

These results may easily be generalized non-zero threshold: To include threshold f_i should be substituted with $f_i + b$. The Kuhn-Tucker condition for the margin support vectors therefore changes to $y_i(f_i + b) = 1$ which implies $\delta f_i = -\delta b$. E.g. for l being a margin support vector, we now have

$$\delta \alpha_i = \sum_{j \neq l}^{\text{mSV}} \left[(\mathbf{k}_{\text{mSV}}^{\setminus l})^{-1} \right]_{ij} (k_{jl} y_l \alpha_l - \delta b) . \quad (17.32)$$

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

17.6 Naive Mean Field Algorithm

The aim of the mean field approach is to compute an approximation to the Bayes prediction $y^{\text{Bayes}}(\mathbf{x}) = \text{sgn}\langle \text{sgn} f(\mathbf{x}) \rangle$ for the GP classifier, where we have introduced the notation $\langle \dots \rangle$ 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 [Opper and Winther, 1999a] based on the so-called Callen identity [Parisi, 1988]. An independent derivation is given by Opper and Winther [1999b].

We will use the simplified prediction $y(\mathbf{x}) = \text{sgn}(f(\mathbf{x}))$ which the Bayes classifier reduces to when the posterior is symmetric around its mean. We first give exact expressions for the posterior

$$\langle f(\mathbf{x}) \rangle = \frac{1}{p(\mathbf{y})} \int d\mathbf{f} df f p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, f(\mathbf{x})) . \quad (17.33)$$

Using the following identity $f_j p(\mathbf{f}) = -\sum_i k(\mathbf{x}_j, \mathbf{x}_i) \frac{\partial}{\partial f_i} p(\mathbf{f})$ (or rather its extension to $p(\mathbf{f}, f)$), which is easily derived from (17.3) setting $\mathbf{m} = 0$, we can write

$$\langle f(\mathbf{x}) \rangle = -\frac{1}{p(\mathbf{y})} \int d\mathbf{f} df p(\mathbf{y}|\mathbf{f}) \sum_i k(\mathbf{x}, \mathbf{x}_i) \frac{\partial}{\partial f_i} p(\mathbf{f}, f(\mathbf{x})) \quad (17.34)$$

We may now use integration by parts to shift the differentiation from the prior to the Likelihood:

$$\begin{aligned} \langle f(\mathbf{x}) \rangle &= \sum_i k(\mathbf{x}, \mathbf{x}_i) \frac{1}{p(\mathbf{y})} \int d\mathbf{f} df p(\mathbf{f}, f(\mathbf{x})) \frac{\partial}{\partial f_i} p(\mathbf{y}|\mathbf{f}) \\ &= \sum_{i=1}^m k(\mathbf{x}, \mathbf{x}_i) y_i \alpha_i . \end{aligned} \quad (17.35)$$

Remarkably, this has the same form as the prediction of the SVM eq. (1.81). While for the SVM, the corresponding representation follows directly from the representer theorem of Kimeldorf and Wahba [1971], we can not use this argument for the mean field method, because (17.35) is not derived from minimizing a cost function. For the mean field approach, the "embedding strength" α_i of example i is given by

$$\alpha_i = \frac{y_i}{p(\mathbf{y})} \int d\mathbf{f} p(\mathbf{f}) \frac{\partial}{\partial f_i} p(\mathbf{y}|\mathbf{f}) \quad (17.36)$$

Note that the α_i 's will always be non-negative when $p(y_i|f(\mathbf{x}_i))$ is an increasing function of $y_i f(\mathbf{x}_i)$.

We give now a mean field argument for the approximate computation of the α_i . There are different ways of defining a mean field theory. The present one has the advantage over other approaches [Opper and Winther, 1999a], that no matrix inversions are needed in the final algorithm. To proceed, auxiliary variables \mathbf{t} are introduced using a standard Gaussian transformation

$$\alpha_i = \frac{y_i}{p(\mathbf{y})} \int \frac{d\mathbf{f} d\mathbf{t}}{(2\pi)^m} \exp\left(-\frac{1}{2}\mathbf{t}^T \mathbf{k} \mathbf{t} + i \mathbf{t}^T \mathbf{f}\right) \frac{\partial}{\partial f_i} p(\mathbf{y}|\mathbf{f}) \quad (17.37)$$

$$= \frac{y_i}{p(\mathbf{y})} \int \frac{d\mathbf{f}d\mathbf{t}}{(2\pi)^m} (-it_i) \exp\left(-\frac{1}{2}\mathbf{t}^T\mathbf{k}\mathbf{t} + it^T\mathbf{f}\right) p(\mathbf{y}|\mathbf{f}) = -iy_i\langle it_i \rangle,$$

where the i not appearing as an index is the imaginary unit $i = \sqrt{-1}$. 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{f} and \mathbf{t} . Next, we separate the integrations over f_i and t_i from the rest of the variables to get

$$\alpha_i = y_i \left\langle \frac{\int df_i dt_i \exp\left(-\frac{1}{2}k_{ii}(t_i)^2 + (-it_i)(\sum_{j \neq i} k_{ij}(-it_j) - f_i)\right) \frac{\partial p(y_i|f_i)}{\partial f_i}}{\int df_i dt_i \exp\left(-\frac{1}{2}k_{ii}(t_i)^2 + (-it_i)(\sum_{j \neq i} k_{ij}(-it_j) - f_i)\right) p(y_i|f_i)} \right\rangle \quad (17.38)$$

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

We may now carry out the explicitly written integrals over f_i and t_i . Using the Likelihood for output noise eq. (17.17), we find

$$\begin{aligned} \alpha_i &= y_i \left\langle \frac{\int df_i \exp\left(-\frac{(f_i - \sum_{j \neq i} k_{ij}(-it_j))^2}{2k_{ii}}\right) \frac{\partial p(y_i|f_i)}{\partial f_i}}{\int df_i \exp\left(-\frac{(f_i - \sum_{j \neq i} k_{ij}(-it_j))^2}{2k_{ii}}\right) p(y_i|f_i)} \right\rangle \\ &= \frac{1}{\sqrt{k_{ii}}} \left\langle \frac{(1 - 2\kappa)D\left(\frac{\sum_{j \neq i} k_{ij}(-it_j)}{\sqrt{k_{ii}}}\right)}{\kappa + (1 - 2\kappa)\Phi\left(y_i \frac{\sum_{j \neq i} k_{ij}(-it_j)}{\sqrt{k_{ii}}}\right)} \right\rangle, \end{aligned} \quad (17.39)$$

“naive” mean field approximation

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 variable $\sum_{j \neq i} k_{ij}(-it_j)$ and substituting it with its expectation $\sum_{j \neq i} k_{ij}\langle -it_j \rangle = \sum_{j \neq i} k_{ij}y_j\alpha_j$. This corresponds to moving the expectation through the nonlinearities. One should however keep in mind, that the integrations are over a complex measure and that the t_j are not random variables in a strict sense. The result of this approximation is a self-consistent set of equations for $\alpha_i = -iy_i\langle t_i \rangle$. The explicit expression for α_i becomes

$$\alpha_i = \frac{1}{\sqrt{k_{ii}}} \frac{(1 - 2\kappa)D(z_i)}{\kappa + (1 - 2\kappa)\Phi(z_i)}, \quad z_i = y_i \frac{\langle f_i \rangle - k_{ii}y_i\alpha_i}{\sqrt{k_{ii}}}. \quad (17.40)$$

In Figure 17.1, α_i is plotted as function of z_i (with $k_{ii} = 1$). The shape of the “embedding”-function depends crucially upon whether we model with or without output noise. For the noise-free case, $\kappa = 0$, α_i is a decreasing function of $y_i\langle f_i \rangle - k_{ii}\alpha_i = z_i\sqrt{k_{ii}}$ 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 it is to predict an example’s label, the larger weight α_i it should have.² In the noisy case, α_i is a decreasing

2. In the more advanced TAP (named after Thouless, Anderson & Palmer) mean field theory z_i is proportional to the “unlearned” mean activation [Oppen and Winther, 1999a].

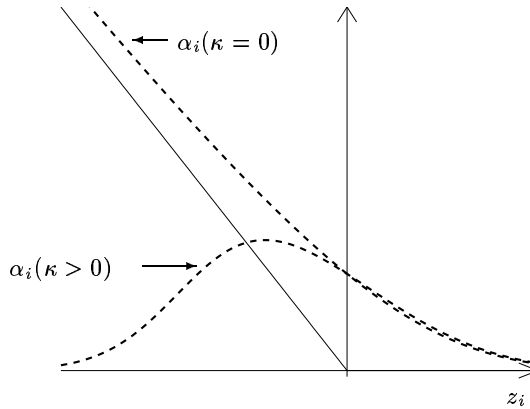


Figure 17.1 The “embedding strength” α_i plotted as a function of z_i with $k_{ii} = 1$.

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 for the mean field algorithm α_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. In Table 17.1, we give pseudo-code for a parallel iterative scheme for the solution of the mean field equations. An

Algorithm 17.1 : Naive mean field

Initialization:

Start from *tabula rasa*, $\alpha := 0$.

Learning rate, $\eta := 0.05$.

Fault tolerance, $\text{ftol} := 10^{-5}$.

Iterate:

while $\max_i |\delta\alpha_i|^2 > \text{ftol}$ **do:**

for all i :

$$\langle f_i \rangle := \sum_j k_{ij} y_j \alpha_j$$

$$\delta\alpha_i := \frac{1}{\sqrt{k_{ii}}} \frac{(1 - 2\kappa)D(z_i)}{\kappa + (1 - 2\kappa)\Phi(z_i)} - \alpha_i, \quad z_i \equiv y_i \frac{\langle f_i \rangle - k_{ii} y_i \alpha_i}{\sqrt{k_{ii}}}$$

endfor

for all i :

$$\alpha_i := \alpha_i + \eta \delta\alpha_i$$

endwhile

important contributing factor to ensure (and to get fast) convergence is the use of an adaptable learning rate: We set $\eta := 1.1\eta$ if “the error” $\sum_i |\delta\alpha_i|^2$ decreases in

the update step and $\eta := \eta/2$ otherwise. Clearly, the algorithm does not converge for all values of the hyperparameters.³ However, if the SVM has a solution for a certain choice of hyperparameters, the mean field algorithm will almost always converge to a solution and vice versa. The important question of how to tune the hyperparameters is discussed in the following.

leave-one-out
estimator

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 [Oppen and Winther, 1999a] in completely the same fashion as the leave-one-out estimator for SVM

$$\epsilon_{\text{loo}}^{\text{naive}} = \frac{1}{m} \sum_i^{\text{SV}} \Theta \left(-y_i \langle f_i \rangle + \left\{ \frac{1}{[(\mathbf{\Omega} + \mathbf{k})^{-1}]_{ii}} - \Omega_i \right\} \alpha_i \right), \quad (17.41)$$

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

$$\Omega_i = k_{ii} \left(\frac{1}{y_i \alpha_i \langle f_i \rangle} - 1 \right). \quad (17.42)$$

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 [Oppen and Winther, 1999b,a]. We also show that despite the fact that this algorithm looks very different from SVM, the solution obtained and the performance is quite similar. The mean field approach will tend to produce smaller minimal margin, however we have not observed that this has any effect on performance.

17.7 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., [Ster and Dobnikar, 1996]. We have removed the 16 examples with missing values and used standard preprocessing as to set the mean for every input equal 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 [Ster and Dobnikar, 1996].

For SVM, we used the parallel version of the Adatron algorithm of Anlauf and Biehl [1989] which, extended to general covariance functions, has turned out to

3. In Bayesian modeling, hyperparameters refer to “higher level” parameters which are not determined directly in the algorithm (in contrast to, e.g., α). The hyperparameters for this algorithm are the output flip probability κ , the input noise variance v and the input lengthscale(s) in the kernel, e.g., σ in the radial basis kernel eq. (1.73). The algorithm depends on the two latter hyperparameters only through the covariance matrix eq. (17.16).

be a fast iterative algorithm [Frieß et al., 1998]. For naive mean field theory, we solved the mean field equations using the iterative scheme described in the previous section.

We chose to work with the radial basis covariance function eq. (1.73). The Gaussian noise model is used in noisy process formulation thus adding the input noise variance v to the diagonal of the covariance matrix as in eq. (17.16). For the mean field algorithm, we have the additional output noise parameter κ . 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 found the values $\sigma^2 = 0.15/N$ 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 assess the performance through the 10-fold cross validation scheme, the loo estimate and test error are not independent. However, our main emphasis is not on generalization performance but rather on 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 indicate the number of misclassifications. The loo errors are $\epsilon_{100} = 0.0293$ and $\epsilon_{100} = 0.0270$. The more advanced TAP mean field algorithm [Opper and Winther, 1999b,a] finds a solution very similar to the one of the naive mean field algorithm. In another study using the SVM-algorithm, Frieß et al. [1998] find $\epsilon = 0.0052$. The difference may be due to a number of reasons: different splitting of the data set, different choice of hyperparameters, use of bias and/or handling of missing values. With other methods the following error rates are found: multi-layer neural networks $\epsilon = 0.034$, linear discriminant $\epsilon = 0.040$, RBF neural networks $\epsilon = 0.041$ and CART $\epsilon = 0.058$ [Ster and Dobnikar, 1996].

In Table 17.1, 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 required precision, $\max_i |\delta\alpha_i|^2 < \text{ftol} = 10^{-5}$. We also compare the leave-one-out estimator ϵ_{100} with the exact loo estimator $\epsilon_{100}^{\text{exact}}$ 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 σ 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 [Opper and Winther, 1999a].

Without evaluating the loo estimators, the naive mean field algorithm is about 4 times faster than the Adatron. 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 $\epsilon_{100}^{\text{SVM}}$, eq. (17.26), we only need to invert the covariance matrix for the margin support vector examples, which in this example is 272-dimensional, whereas $\epsilon_{100}^{\text{naive}}$, eq. (17.41) 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 $\epsilon_{100}^{\text{SVM}}$ would have been even greater.

Table 17.1 Results for the Wisconsin dataset.

Algorithm	$\epsilon_{\text{loo}}^{\text{exact}}$	ϵ_{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

In Figure 17.2, we compare the solutions found by the two algorithms. The solutions for the “embedding strengths” α_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 α_i starting around the point where the α_i -curve’s slope increases. This observation suggests that a heuristic cut-off for small α_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 α_i , if one expects the error rate to be around 5%.

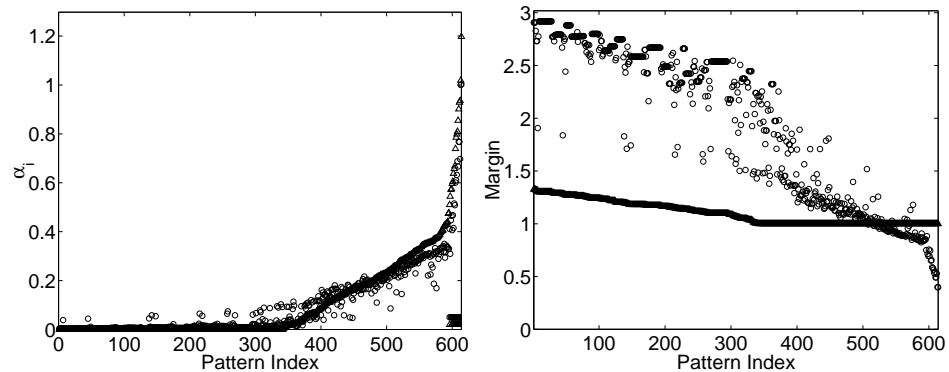


Figure 17.2 Left figure: The “embedding strengths” α_i for each example. The right figure: The margins $y_i f_i$ for SVM and $y_i \langle f_i \rangle$ for naive mean field theory (same ordering as the left plot). The triangles are for support vectors and circles are for naive mean field theory. They are sorted in ascending order according to their support vector α_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.

17.8 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 [Berger, 1985, MacKay, 1992]. 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.