# Expectation Propagation for approximate Bayesian inference

Thomas P. Minka

December 7, 2000

**Abstract**

Approximate integration techniques are important for reducing the computational cost of Bayesian inference. Recently, two fast algorithms have been shown to be effective for Bayesian inference: loopy belief propagation and assumed-density filtering. Both of these algorithms can be viewed as approximating the true posterior with a simpler distribution. I describe a unifying framework, "Expectation Propagation," which combines the generality of assumed-density filtering with the accuracy of belief propagation. Whereas belief propagation is useful only for discrete belief networks, Expectation Propagation can be used on probabilistic models with both discrete and continuous variables. Any member of the exponential family can be used as an approximation—the minimal sufficient statistics are the "expectations." Simulations with Gaussian mixture densities show Expectation Propagation to be convincingly better than Laplace's method, variational Bayes, and Monte Carlo. Using Expectation Propagation, I derive a fast algorithm for training Bayes point machine classifiers and for choosing an appropriate feature set. The Bayesian approach compares favorably to the support vector machine for data classification.

## 1 Introduction

The dominant computational task in Bayesian inference is numerical integration. New methods for fast and accurate integration are therefore very important and can have great impact. This paper presents a new deterministic approximation algorithm, Expectation Propagation, which achieves higher accuracy than existing integration algorithms with similar computational cost.

Expectation Propagation is based on the idea of *assumed-density filtering* (ADF) (Maybeck, 1982; Lauritzen, 1992; Bernardo & Giron, 1988; Stephens, 1997; Boyen & Koller, 1998b; Barber & Sollich, 1999; Opper & Winther, 1999). Assumed-density filtering is a one-pass, sequential method for computing an approximate posterior distribution. Observations are processed one by one, updating the posterior distribution which is then approximated before processing the next observation. For example, we might replace the exact one-step posterior with a Gaussian having the same mean and same variance (Maybeck, 1982; Lauritzen, 1992; Barber & Sollich, 1999; Opper & Winther, 1999). Or we might replace a posterior over many variables with one that renders the variables independent (Boyen & Koller, 1998b). In both cases, the approximate posterior is found by minimizing KL-divergence, which amounts to preserving a specific set of posterior *expectations*. Note that the statistical model in question need not be Markovian, and the processing order of observations need not correspond with time of arrival. The weakness of ADF stems from its sequential nature: information which is discarded early on may turn out to be important later. Even if ADF is augmented with a backward pass (Boyen & Koller, 1998a), this information cannot be recovered. ADF is also sensitive to observation ordering, which is undesirable in a batch context.

Expectation Propagation extends ADF to incorporate *iterative refinement* of the approximations. Each approximation is made in the context of the entire dataset, not just those observations which have been processed so far. In this way, the most important information is always retained and the result is independent of ordering. Iterative refinement has previously been used in conjunction with sampling (Koller et al., 1999) and extended Kalman filtering (Shachter, 1990). Expectation Propagation differs by applying this idea to the *deterministic* approximation of *general* distributions.

Belief propagation in loopy networks can be shown to be a special case of Expectation Propagation, where the variables are all discrete and the approximate posterior is completely factorized. Expectation Propagation can thus be seen as a generalization of belief propagation to networks with discrete and continuous nodes and to approximations which are not completely factorized. For example, section 6 uses a full-covariance Gaussian posterior for the continuous variables in a hybrid network.

Expectation Propagation also has connections to statistical physics techniques. Yedidia et al. (2000) have shown that belief propagation tries to minimize an approximate free energy on discrete networks. This is known as the TAP approach in statistical physics. Opper & Winther (2000a) have recently generalized the TAP approach to networks with mixed continuous and discrete nodes via the cavity method, and applied it to Gaussian process classifiers (Opper & Winther, 2000c). Section 6 shows that Opper & Winther's algorithm, which has excellent performance, is a special case of Expectation Propagation where the posterior approximation is Gaussian. In this sense, Expectation Propagation as a generalization of belief propagation parallels the cavity method as a generalization of TAP.

# 2 Assumed-density filtering

This section reviews the idea of assumed-density filtering (ADF), to lay groundwork for Expectation Propagation. Assumed-density filtering is a general technique for computing approximate posteriors in Bayesian networks and other statistical models. ADF has been independently proposed in the statistics (Lauritzen, 1992; Bernardo & Giron, 1988; Stephens, 1997), artificial intelligence (Boyen & Koller, 1998b; Opper & Winther, 1999; Barber & Sollich, 1999), and control literature (Kushner & Budhiraja, 2000; Maybeck, 1982). "Assumed-density filtering" is the name used in control. ADF applies when we have postulated a joint distribution $p(D, \theta)$ where $D$ has been observed and $\theta$ is hidden. We would like to know the posterior over $\theta$, $p(\theta|D)$, as well as the probability of the observed data (or evidence for the model), $p(D)$. The former is useful for estimation while the latter is useful for model selection.

For example, suppose we have observations from a Gaussian distribution embedded in a sea of unrelated clutter, so that the observation density is a mixture of two Gaussians:

$$\begin{aligned} p(\mathbf{x}|\theta) &= (1-w)\mathcal{N}(\mathbf{x}; \theta, \mathbf{I}) + w\mathcal{N}(\mathbf{x}; \mathbf{0}, 10\mathbf{I}) & (1) \\ \mathcal{N}(\mathbf{x}; \mathbf{m}, \mathbf{V}) &= \frac{1}{|2\pi\mathbf{V}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^{\mathrm{T}}\mathbf{V}^{-1}(\mathbf{x}-\mathbf{m})) & (2) \end{aligned}$$

The first component contains the parameter of interest, while the other component describes clutter. $w$ is the known ratio of clutter. Let $\theta$ have a Gaussian prior distribution:

$$p(\theta) \sim \mathcal{N}(\mathbf{0}, 100\mathbf{I}) \quad (3)$$

The joint distribution of $\theta$ and $n$ independent observations $D = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ is therefore:

$$p(D, \theta) = p(\theta) \prod_i p(\mathbf{x}_i|\theta) \quad (4)$$

To apply ADF, we first factor the joint distribution $p(D, \theta)$ into a product of simple terms:

$$p(D, \theta) = \prod_i t_i(\theta) \quad (5)$$

There are many ways to do this. As a rule of thumb, fewer terms are better, since it entails fewer approximations. However, we need each term to be simple enough to propagate expectations through. In the mixture example, we can use the factoring into $n + 1$ terms implied by (4):

$$\begin{aligned} t_0(\theta) &= p(\theta) & (6) \\ t_i(\theta) &= p(\mathbf{x}_i|\theta) & (7) \end{aligned}$$

For a general Bayesian network, we would use the factoring into conditional probability tables: $\prod_{\text{nodes } X} p(X|\text{pa}(X))$.

$$\begin{array}{ccccc} & q(\theta) & & \hat{p}(\theta) & \\ \multicolumn{5}{c}{\longrightarrow\dashrightarrow} \\ t_1(\theta) & t_2(\theta) & t_3(\theta) & t_4(\theta) & t_5(\theta) \\ \multicolumn{5}{c}{\longrightarrow} \\ & & q^{new}(\theta) & & \end{array}$$

Figure 1: Assumed-density filtering computes an exact one-step posterior $\hat{p}(\theta)$ and then approximates it to get $q^{new}(\theta)$.

The second step is to choose a parametric approximating distribution. It is essential that the distribution be in the exponential family, so that only a fixed number of expectations need to be propagated. Usually the nature and domain of $\theta$ constrains the distribution enough that there is no choice left to make. In the clutter problem, a spherical Gaussian distribution is appropriate. The approximate posterior is therefore

$$q(\theta) \sim \mathcal{N}(\mathbf{m}_\theta, v_\theta \mathbf{I}_d) \tag{8}$$

Finally, we sequence through and incorporate the terms $t_i$ into the approximate posterior. At each step we move from an old $q(\theta)$ to a new $q(\theta)$, as shown in figure 1. (For notational simplicity, we drop the dependence of $q(\theta)$ on $i$.) Initialize with $q(\theta) = 1$. Incorporating the prior term is trivial, with no approximation needed. To incorporate a more complicated term $t(\theta)$, take the exact posterior

$$\hat{p}(\theta) = \frac{t(\theta)q(\theta)}{\int_\theta t(\theta)q(\theta)d\theta} \tag{9}$$

and minimize the KL-divergence $D(\hat{p}(\theta)||q^{new}(\theta))$ subject to the constraint that $q^{new}(\theta)$ is a Gaussian distribution. Zeroing the gradient with respect to $(\mathbf{m}_\theta, v_\theta)$ gives the conditions

$$\mathbf{m}_\theta^{new} = \int_\theta \hat{p}(\theta)\theta \; d\theta \tag{10}$$

$$v_\theta^{new} d + (\mathbf{m}_\theta^{new})^{\mathrm{T}}(\mathbf{m}_\theta^{new}) = \int_\theta \hat{p}(\theta)\theta^{\mathrm{T}}\theta \; d\theta \tag{11}$$

or in other words, expectation constraints:

$$E_{q^{new}}[\theta] = E_{\hat{p}}[\theta] \tag{12}$$

$$E_{q^{new}}[\theta^{\mathrm{T}}\theta] = E_{\hat{p}}[\theta^{\mathrm{T}}\theta] \tag{13}$$

For the spherical Gaussian distribution we see that the characteristic expectations are $(E[\theta], E[\theta^{\mathrm{T}}\theta])$. For other members of the exponential family, we will get constraints on different expectations—see section 4.

To compute these expectations, it is helpful to exploit the following relations (obtained from integration-by-parts):

$$Z(\mathbf{m}_\theta, v_\theta) = \int_\theta t(\theta)q(\theta) \; d\theta \tag{14}$$

$$E_{\hat{p}}[\theta] = \mathbf{m}_\theta + v_\theta \nabla_m \log Z(\mathbf{m}_\theta, v_\theta) \tag{15}$$

$$E_{\hat{p}}[\theta^{\mathrm{T}}\theta] - E_{\hat{p}}[\theta]^{\mathrm{T}} E_{\hat{p}}[\theta] = dv_\theta - v_\theta^2(\nabla_m^{\mathrm{T}}\nabla_m - 2\nabla_v \log Z(\mathbf{m}_\theta, v_\theta)) \tag{16}$$

These relations are a property of the Gaussian distribution and hold for any $t(\theta)$. In the clutter problem, we have

$$Z(\mathbf{m}_\theta, v_\theta) = (1-w)\mathcal{N}(\mathbf{x}; \mathbf{m}_\theta, (v_\theta+1)\mathbf{I}) + w\mathcal{N}(\mathbf{x}; \mathbf{0}, 10\mathbf{I}) \tag{17}$$

$$r = \frac{(1-w)\mathcal{N}(\mathbf{x}; \mathbf{m}_\theta, (v_\theta+1)\mathbf{I})}{(1-w)\mathcal{N}(\mathbf{x}; \mathbf{m}_\theta, (v_\theta+1)\mathbf{I}) + w\mathcal{N}(\mathbf{x}; \mathbf{0}, 10\mathbf{I})} \tag{18}$$

$$\nabla_m \log Z(\mathbf{m}_\theta, v_\theta) = r\frac{\mathbf{x} - \mathbf{m}_\theta}{v_\theta + 1} \tag{19}$$

$$\nabla_v \log Z(\mathbf{m}_\theta, v_\theta) = -\frac{rd}{2(v_\theta+1)} + \frac{r(\mathbf{x}-\mathbf{m}_\theta)^{\mathrm{T}}(\mathbf{x}-\mathbf{m}_\theta)}{2(v_\theta+1)^2} \tag{20}$$

$$\nabla_m^{\mathrm{T}}\nabla_m - 2\nabla_v \log Z(\mathbf{m}_\theta, v_\theta) = \frac{rd}{v_\theta+1} - r(1-r)\frac{(\mathbf{x}-\mathbf{m}_\theta)^{\mathrm{T}}(\mathbf{x}-\mathbf{m}_\theta)}{(v_\theta+1)^2} \tag{21}$$

An estimate of the probability of the data, $p(D)$, is a trivial byproduct of ADF. Simply accumulate the normalization factors $Z(\mathbf{m}_\theta, v_\theta)$ produced by each update, to get an overall scale factor for the posterior. This scale factor estimates $p(D)$ because $p(D, \theta) = p(\theta|D)p(D)$.

The final ADF algorithm is:

1. Initialize $\mathbf{m}_\theta = \mathbf{0}$, $v_\theta = 100$ (the prior). Initialize $s = 1$ (the scale factor).
2. For each data point $\mathbf{x}_i$, update $(\mathbf{m}_\theta, v_\theta, s)$ according to

$$\mathbf{m}_\theta^{new} = \mathbf{m}_\theta + v_\theta r_i \frac{\mathbf{x}_i - \mathbf{m}_\theta}{v_\theta + 1} \tag{22}$$

$$v_\theta^{new} = v_\theta - r_i\frac{v_\theta^2}{v_\theta+1} + r_i(1-r_i)\frac{v_\theta^2(\mathbf{x}_i-\mathbf{m}_\theta)^{\mathrm{T}}(\mathbf{x}_i-\mathbf{m}_\theta)}{d(v_\theta+1)^2} \tag{23}$$

$$s^{new} = s \times Z_i(\mathbf{m}_\theta, v_\theta) \tag{24}$$

This algorithm can be understood in an intuitive way: for each data point we compute its probability $r$ of not being clutter, make a soft update to our estimate of $\theta$ ($\mathbf{m}_\theta$), and change our confidence in the estimate ($v_\theta$). However, it is clear that this algorithm will depend on the order in which data is processed, because the clutter probability depends on the current estimate of $\theta$.

This algorithm was derived using a spherical Gaussian approximating distribution. If we use a richer set of distributions like diagonal or full-covariance Gaussians, we are guaranteed to get a better result, since more expectations will be preserved. There is no overfitting problem in approximating posterior distributions.

Figure 2 shows the output of this algorithm using three random orderings of the same data. Synthetic data was generated using $w = 0.5$ and $\theta = 2$. The next section shows how to modify ADF to eliminate the dependence on data ordering and garner additional accuracy.

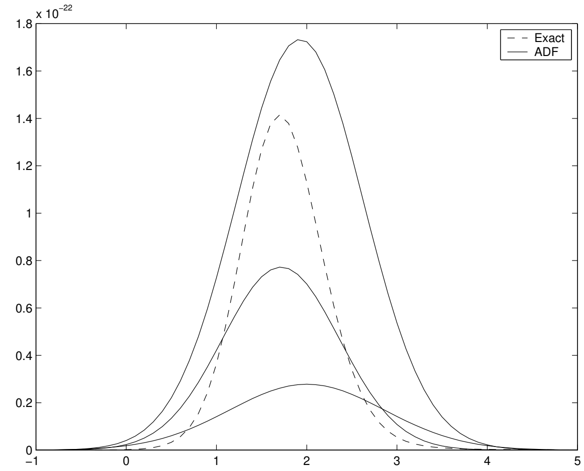



Figure 2: The approximate posterior resulting from ADF, using three different orderings of the same data. The posterior is scaled by the evidence estimate, $p(D)$.

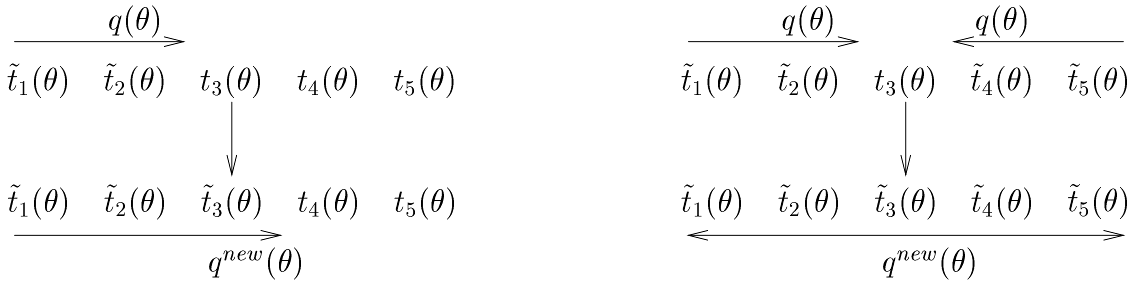



Figure 3: An alternative view of ADF as approximating each term and then computing $q^{new}(\theta)$ exactly. EP refines each term in the context of all other terms.

# 3 Expectation Propagation

This section describes the Expectation Propagation algorithm and demonstrates its use on the clutter problem. Expectation Propagation is based on a novel interpretation of assumed-density filtering. Normally we think of ADF as treating each observation term exactly and then approximating the posterior. But we can also think of it as approximating the observation and using an exact posterior (figure 3). This is always possible because we can simply define the approximate observation term to be the ratio of the new posterior to the old posterior times a constant. For example, Gaussian posterior approximations are equivalent to Gaussian term approximations, because the ratio of two Gaussians is a Gaussian.

The algorithm of the previous section can thus be interpreted as sequentially computing a Gaussian approximation $\tilde{t}_i(\theta)$ to every observation term $t_i(\theta)$, then combining these approximations analytically to get a Gaussian posterior on $\theta$. In this view, it is natural to consider refining the approximations $\tilde{t}_i(\theta)$ on the basis of all the data. Since the posterior is exact given the termwise approximations, it doesn't matter what order we refine them. From this idea we get Expectation Propagation.

Using spherical Gaussians as in the previous section, we get a term approximation defined by $\mathbf{m}_i$ and $v_i$:

$$Z(\mathbf{m}_\theta, v_\theta) = \int_\theta t(\theta) q(\theta) d\theta \tag{25}$$

$$\tilde{t}(\theta) = Z(\mathbf{m}_\theta, v_\theta) \frac{q^{new}(\theta)}{q(\theta)} \tag{26}$$

$$= Z(\mathbf{m}_\theta, v_\theta) \left(\frac{v_\theta}{v_\theta^{new}}\right)^{d/2} \exp\left(-\frac{1}{2v_\theta^{new}}(\theta - \mathbf{m}_\theta^{new})^{\mathrm{T}}(\theta - \mathbf{m}_\theta^{new})\right) \tag{27}$$

$$\exp\left(\frac{1}{2v_\theta}(\theta - \mathbf{m}_\theta)^{\mathrm{T}}(\theta - \mathbf{m}_\theta)\right)$$

$$= \frac{Z(\mathbf{m}_\theta, v_\theta)}{\mathcal{N}(\mathbf{m}_i; \mathbf{m}_\theta, (v_i + v_\theta)\mathbf{I})} \mathcal{N}(\theta; \mathbf{m}_i, v_i \mathbf{I}) \tag{28}$$

$$
\begin{aligned}
\mathbf{m}_\theta^{new} &= v_\theta^{new}(v_i^{-1}\mathbf{m}_i + v_\theta^{-1}\mathbf{m}_\theta) && (29)\\
v_\theta^{new} &= (v_i^{-1} + v_\theta^{-1})^{-1} && (30)
\end{aligned}
$$

Multiplying $q(\theta)$ by this approximation gives exactly the ADF update, with the proper scale factor.

So far we have just rewritten the ADF updates in a different way. To get EP, we refine the term variables $(\mathbf{m}_i, v_i)$ based on all of the other approximations. The general EP algorithm is thus:

1. Initialize the term approximations $\tilde{t}_i$
2. Compute the posterior for $\theta$ from the product of $\tilde{t}_i$:

$$
q^{new}(\theta) = \frac{\prod_i \tilde{t}_i(\theta)}{\int \prod_i \tilde{t}_i(\theta) d\theta} \tag{31}
$$

3. Until all $\tilde{t}_i$ converge:
   (a) Choose a $\tilde{t}_i$ to refine
   (b) Remove $\tilde{t}_i$ from the posterior to get an 'old' posterior $q(\theta)$, by dividing and normalizing:

$$
q(\theta) \propto \frac{q^{new}(\theta)}{\tilde{t}_i(\theta)} \tag{32}
$$

   (For notational simplicity, we drop the dependence of $q(\theta)$ on $i$.)
   (c) Compute the posterior $q^{new}(\theta)$ and scale factor $Z$ from $q(\theta)$ and $t_i(\theta)$ via ADF.
   (d) Set

$$
\tilde{t}_i = Z\frac{q^{new}(\theta)}{q(\theta)} \tag{33}
$$

4. Use the scale factor of the posterior as an approximation to $p(D)$:

$$
p(D) \approx \int \prod_i \tilde{t}_i(\theta) d\theta \tag{34}
$$

In this algorithm, we have used division to remove $\tilde{t}_i$ from the posterior. Step 3b can also be performed without division, by accumulating all terms except for $\tilde{t}_i$:

$$
q(\theta) \propto \prod_{j \neq i} \tilde{t}_j(\theta) \tag{35}
$$

but division is usually more efficient.

## 3.1 The clutter problem

For the clutter problem of the previous section, the EP algorithm is:

1. The term approximations have the form

$$\tilde{t}_i(\theta) = s_i \exp(-\frac{1}{2v_i}(\theta - \mathbf{m}_i)^{\mathrm{T}}(\theta - \mathbf{m}_i)) \tag{36}$$

   Initialize the prior term to itself:

$$\begin{aligned} v_0 &= 100 & (37)\\ \mathbf{m}_0 &= \mathbf{0} & (38)\\ s_0 &= (2\pi v_0)^{-d/2} & (39) \end{aligned}$$

   Initialize the data terms to 1:

$$\begin{aligned} v_i &= \infty & (40)\\ \mathbf{m}_i &= \mathbf{0} & (41)\\ s_i &= 1 & (42) \end{aligned}$$

2. $\mathbf{m}_\theta^{new} = \mathbf{m}_0, v_\theta^{new} = v_0$
3. Until all $(\mathbf{m}_i, v_i, s_i)$ converge:
   (a) Choose the next data term in sequence, cycling back to 1
   (b) Remove $\tilde{t}_i$ from the posterior to get an 'old' posterior:

$$\begin{aligned} v_\theta^{-1} &= (v_\theta^{new})^{-1} - v_i^{-1} & (43)\\ \mathbf{m}_\theta &= v_\theta (v_\theta^{new})^{-1}\mathbf{m}_\theta^{new} - v_\theta v_i^{-1}\mathbf{m}_i = \mathbf{m}_\theta^{new} + v_\theta v_i^{-1}(\mathbf{m}_\theta^{new} - \mathbf{m}_i) & (44) \end{aligned}$$

   (c) Recompute $(\mathbf{m}_\theta^{new}, v_\theta^{new}, Z_i)$ from $(\mathbf{m}_\theta, v_\theta)$ (this is the same as ADF):

$$\begin{aligned} r_i &= \frac{(1-w)\mathcal{N}(\mathbf{x}_i; \mathbf{m}_\theta, (v_\theta+1)\mathbf{I})}{(1-w)\mathcal{N}(\mathbf{x}_i; \mathbf{m}_\theta, (v_\theta+1)\mathbf{I}) + w\mathcal{N}(\mathbf{x}_i; \mathbf{0}, 10\mathbf{I})} & (45)\\ \mathbf{m}_\theta^{new} &= \mathbf{m}_\theta + v_\theta r_i \frac{\mathbf{x}_i - \mathbf{m}_\theta}{v_\theta + 1} & (46)\\ v_\theta^{new} &= v_\theta - r_i\frac{v_\theta^2}{v_\theta+1} + r_i(1-r_i)\frac{v_\theta^2(\mathbf{x}_i - \mathbf{m}_\theta)^{\mathrm{T}}(\mathbf{x}_i - \mathbf{m}_\theta)}{d(v_\theta+1)^2} & (47)\\ Z_i &= (1-w)\mathcal{N}(\mathbf{x}_i; \mathbf{m}_\theta, (v_\theta+1)\mathbf{I}) + w\mathcal{N}(\mathbf{x}_i; \mathbf{0}, 10\mathbf{I}) & (48) \end{aligned}$$

   (d) Set

$$\begin{aligned} v_i^{-1} &= (v_\theta^{new})^{-1} - v_\theta^{-1} & (49)\\ &= \left(\frac{r_i}{v_\theta+1} - r_i(1-r_i)\frac{(\mathbf{x}_i - \mathbf{m}_\theta)^{\mathrm{T}}(\mathbf{x}_i - \mathbf{m}_\theta)}{d(v_\theta+1)^2}\right)^{-1} - v_\theta & (50)\\ \mathbf{m}_i &= v_i(v_\theta^{new})^{-1}\mathbf{m}_\theta^{new} - v_i v_\theta^{-1}\mathbf{m}_\theta & (51) \end{aligned}$$

$$
\begin{aligned}
&= \mathbf{m}_\theta + (v_i + v_\theta) v_\theta^{-1} (\mathbf{m}_\theta^{new} - \mathbf{m}_\theta) & (52) \\
&= \mathbf{m}_\theta + (v_i + v_\theta) r_i \frac{\mathbf{x}_i - \mathbf{m}_\theta}{v_\theta + 1} & (53) \\
s_i &= \frac{Z_i}{(2\pi v_i)^{d/2} \mathcal{N}(\mathbf{m}_i; \mathbf{m}_\theta, (v_i + v_\theta)\mathbf{I})} & (54)
\end{aligned}
$$

4. Compute the scale factor:

$$
B = \frac{(\mathbf{m}_\theta^{new})^{\mathrm{T}} \mathbf{m}_\theta^{new}}{v_\theta^{new}} - \sum_i \frac{\mathbf{m}_i^{\mathrm{T}} \mathbf{m}_i}{v_i} \tag{55}
$$

$$
p(D) \approx (2\pi v_\theta^{new})^{d/2} \exp(B/2) \prod_{i=0}^{n} s_i \tag{56}
$$

If we run this algorithm for one pass through the data, the result is identical to ADF. On later passes, we have to be careful since some of the term variances $v_i$ may be negative. This is not necessarily a failure of the algorithm; it just means that the best approximation to the true $t_i$ happens to be a function which curves upward. The formula for $s_i$ only requires the square root of $1 + v_\theta v_i^{-1}$, which tends to be positive even when $v_i < 0$.

Unfortunately, for some update orderings it can happen that $v_\theta$, the variance of $q(\theta)$ without some $\tilde{t}_i$, becomes negative. At this point we can skip $\tilde{t}_i$ and refine another. If the other terms converge and this $v_\theta$ is still negative, we must abort. One way to avoid this situation is to relax some of the expectation constraints and force all $v(i)$ to be positive, at some loss in accuracy. A better solution would be to use an iteration scheme with better convergence properties—see the discussion at the end of section 5.

## 3.2 Results and comparisons

This section evaluates ADF and EP on the clutter problem and compares them to four other algorithms for approximate inference: Laplace's method, variational Bayes, importance sampling (specifically likelihood-weighted sampling), and Gibbs sampling.

In Laplace's method, we first run EM to get a MAP estimate of $\theta$. Then we construct a Gaussian approximation to the posterior by using the curvature of the posterior at $\hat{\theta}$. This curvature is (Minka, 2000b)

$$
\mathbf{H} = \nabla_{\theta\theta^{\mathrm{T}}} \log p(\theta|D) = -\frac{1}{100}\mathbf{I} - \sum_i r_i \mathbf{I} + \sum_i r_i (1 - r_i)(\mathbf{x}_i - \hat{\theta})(\mathbf{x}_i - \hat{\theta})^{\mathrm{T}} \tag{57}
$$

where $r_i$ is the same quantity computed in ADF. The Gaussian approximation and scale factor are

$$
p(\theta|D) \approx \mathcal{N}(\hat{\theta}, -\mathbf{H}^{-1}) \tag{58}
$$

$$
p(D) \approx p(D, \hat{\theta}) (2\pi)^{d/2} \left|-\mathbf{H}\right|^{-1/2} \tag{59}
$$

For variational Bayes, we lower bound the joint distribution by bounding each data term (Minka, 2000b):

$$p(\mathbf{x}_i|\theta) \geq \left(\frac{(1-w)\mathcal{N}(\mathbf{x};\theta,\mathbf{I})}{q_{i1}}\right)^{q_{i1}} \left(\frac{w\mathcal{N}(\mathbf{x};\mathbf{0},10\mathbf{I})}{q_{i2}}\right)^{q_{i2}} \tag{60}$$

The variational parameters $q_{ij}$ are optimized to give the tightest bound. Given the bounds, the posterior for $\theta$ is Gaussian:

$$K_j = \sum_i q_{ij} \tag{61}$$

$$\bar{\mathbf{x}}_j = \frac{1}{K_j}\sum_i q_{ij}\mathbf{x}_i \tag{62}$$

$$S_j = \sum_i q_{ij}(\mathbf{x}_i - \bar{\mathbf{x}}_j)(\mathbf{x}_i - \bar{\mathbf{x}}_j)^{\mathrm{T}} \tag{63}$$

$$\mathbf{V}_\theta = \left(K_1\mathbf{I} + \frac{\mathbf{I}}{100}\right)^{-1} \tag{64}$$

$$\mathbf{m}_\theta = \mathbf{V}_\theta K_1 \bar{\mathbf{x}}_1 \tag{65}$$

$$p(\theta|D) \approx \mathcal{N}(\mathbf{m}_\theta, \mathbf{V}_\theta) \tag{66}$$

$$p(D) \geq \frac{1}{(2\pi)^{d(K_1-1)/2}K_1^{d/2}}\mathcal{N}(\bar{\mathbf{x}}_1;\mathbf{0},\mathbf{I}/K_1 + 100)\exp(-\frac{1}{2}\mathrm{tr}(\mathbf{S}_1)) \tag{67}$$

$$\frac{1}{(2\pi 10)^{d(K_2-1)/2}K_2^{d/2}}\mathcal{N}(\bar{\mathbf{x}}_2;\mathbf{0},10\mathbf{I}/K_2)\exp(-\frac{1}{2}\mathrm{tr}(\mathbf{S}_2/10)) \tag{68}$$

$$\prod_i \left(\frac{w}{q_{i1}}\right)^{q_{i1}} \left(\frac{1-w}{q_{i2}}\right)^{q_{i2}} \tag{69}$$

For importance sampling, or more specifically likelihood-weighted sampling, we draw samples $\{\theta_1, ..., \theta_S\}$ from $p(\theta)$ and approximate the integrals via

$$p(D) \approx \frac{1}{S}\sum_{i=1}^{S} p(D|\theta_i) \tag{70}$$

$$E[\theta|D] \approx \frac{\sum_{i=1}^{S}\theta_i p(D|\theta_i)}{\sum_{i=1}^{S} p(D|\theta_i)} \tag{71}$$

For Gibbs sampling, we introduce hidden variables $c_i$ which indicate whether $\mathbf{x}_i$ is clutter or not. Given a choice for $\theta$, we sample $c_i$ from a Bernoulli distribution with mean $r_i$. Then given $c_i$, we form an exact Gaussian posterior for $\theta$ and sample a new $\theta$. The average of the $\theta$'s that arise from this process is our estimate of the posterior mean. Estimating the evidence is tricky via Gibbs sampling so we skip it.

The three deterministic algorithms, EP, Laplace, and VB all obtain approximate posteriors close to the true one. The sampling algorithms only estimate specific integrals. To compare the algorithms quantitatively, we compute the absolute difference between the estimated and exact evidence and the estimated and exact posterior mean. Figure 4 shows the results on a typical run with $n = 20$ and $n = 200$. It plots the accuracy vs. cost of the algorithms on the two integrals. Accuracy is measured by absolute difference from the true integral. Cost is measured by the number of floating point operations (FLOPS) in Matlab, via Matlab's `flops` function. This is better than using CPU time because FLOPS ignores interpretation overhead.

The exact shape of these curves, especially the EP curve, should not be taken too seriously since the convergence properties of EP, including whether it converges at all, can be substantially affected by the particular dataset and data ordering. The Laplace curve is generated by applying (59) to intermediate values of $\hat{\theta}$, even though the gradient may not be zero.

It is interesting to see how the different properties of the algorithms manifest themselves. EP, Laplace, and VB all try to approximate the posterior with a Gaussian, and their performance depends on how well this assumption holds. With more data, the posterior becomes more Gaussian so they all improve. Sampling methods, by contrast, assume very little about the posterior and consequently cannot exploit the fact that it is becoming more Gaussian. However, this disadvantage turns into an advantage when the posterior has a complex shape. Figure 5 shows a run with $n = 20$ where the true posterior has three distinct modes. EP crashes on this data, due to negative variances. Laplace's method and variational Bayes do converge, but to an erroneous result that captures only a single mode. Sampling methods are only slightly challenged by this posterior.

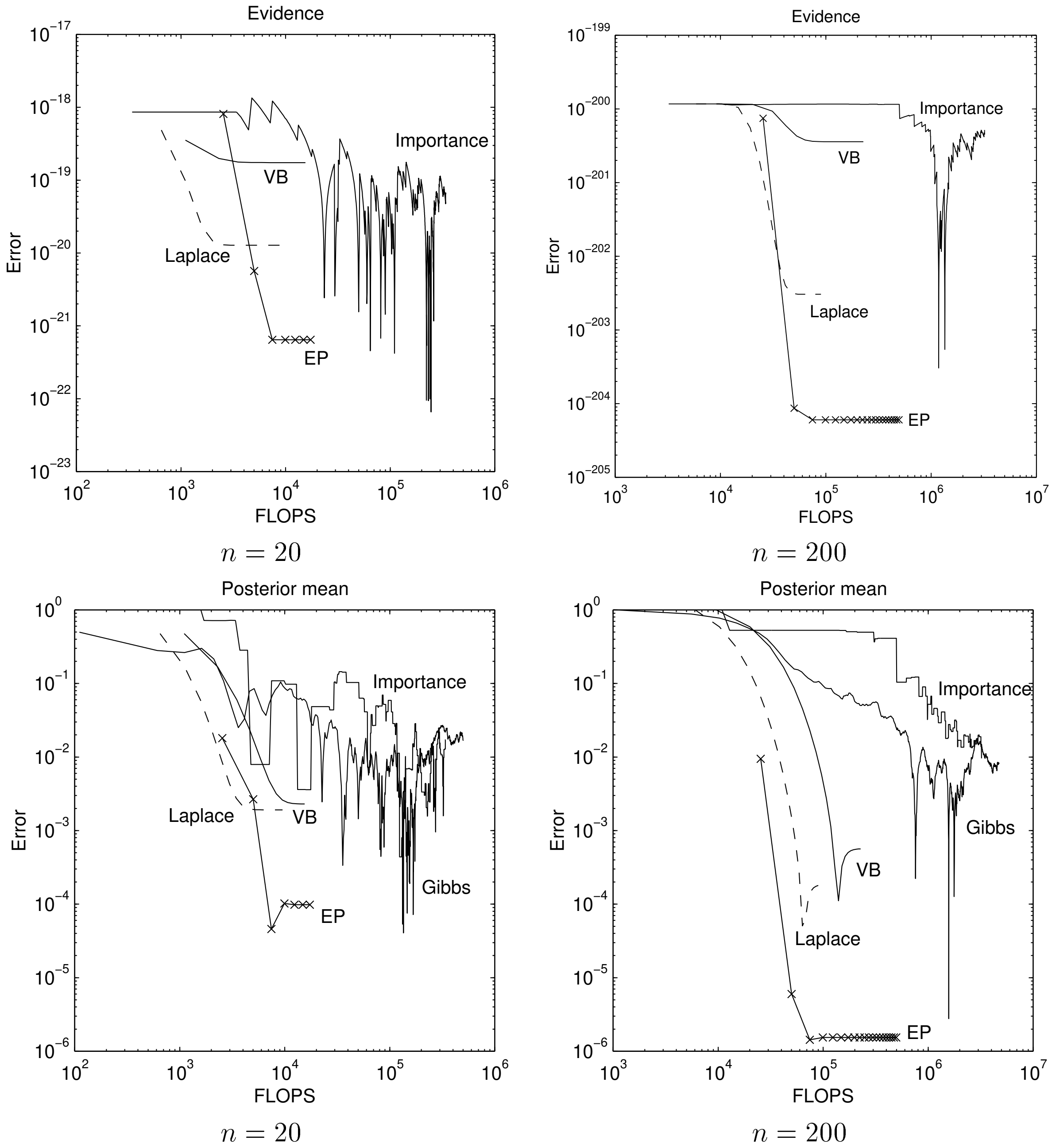



Figure 4: Cost vs. accuracy curves for expectation propagation (EP), Laplace's method, variational Bayes (VB), importance sampling, and Gibbs sampling on the clutter problem with $w = 0.5$ and $\theta = 2$. ADF is equivalent to the first 'x' on the EP curve.

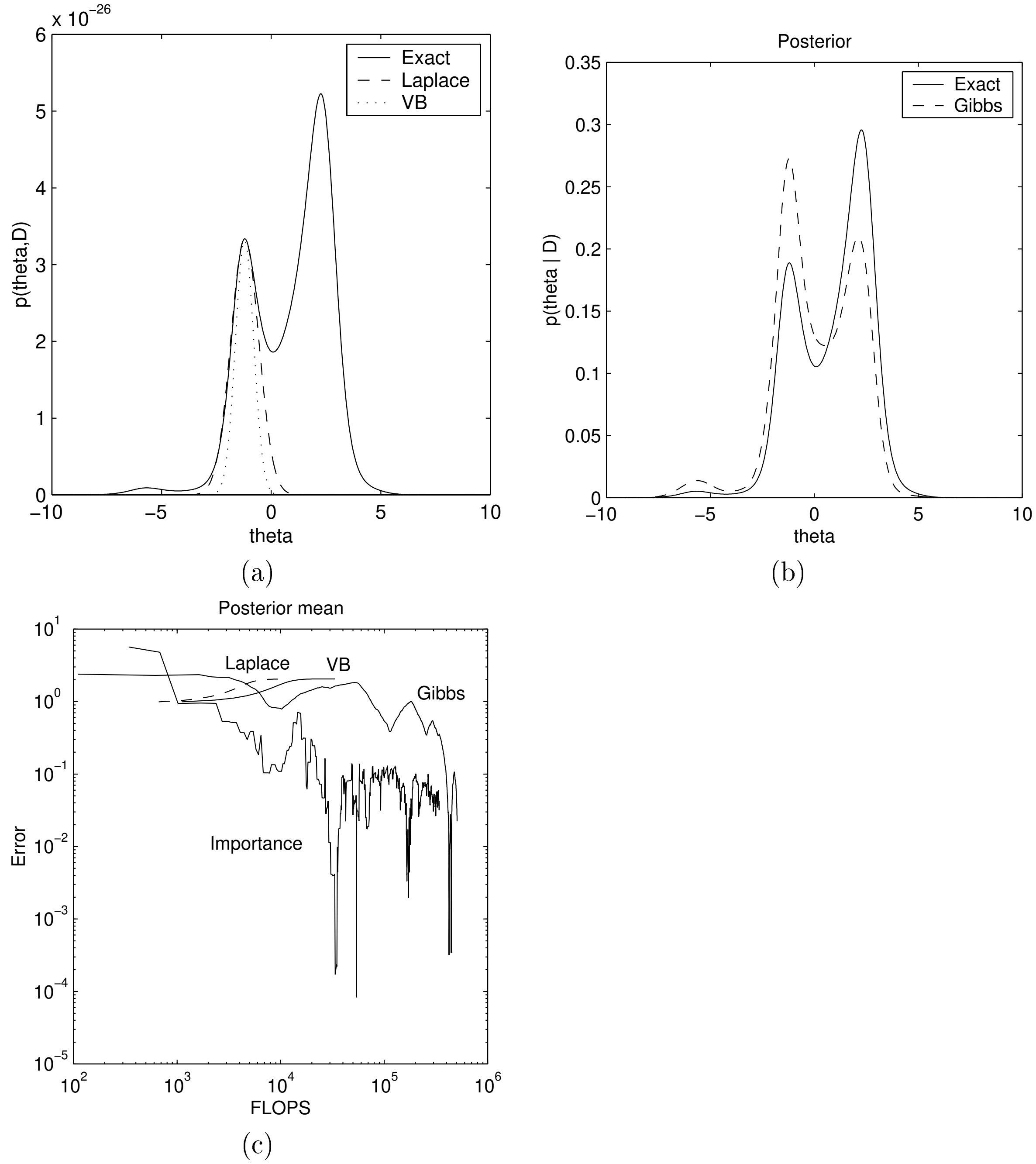



Figure 5: A complex posterior in the clutter problem. (a) Exact posterior vs. approximations obtained by Laplace's method and variational Bayes. (b) Exact posterior vs. approximation obtained by Gibbs sampling and complete conditional density averaging (Gelfand & Smith, 1990). (c) Cost vs. accuracy. EP did not converge on this dataset.

# 4 Another example: Mixture weights

This section demonstrates how the structure of ADF and EP are preserved with a different approximating distribution—a Dirichlet distribution. Let the probabilistic model be a mixture of $K$ known densities, $p_1(x), ..., p_K(x)$, with unknown mixing weights $\mathbf{w}$:

$$\begin{aligned} p(x|\mathbf{w}) &= \sum_k w_k p_k(x) && (72) \\ \sum_k w_k &= 1 && (73) \\ p(\mathbf{w}) &= 1 && (74) \\ p(D, \mathbf{w}) &= p(\mathbf{w}) \prod_i p(x_i|\mathbf{w}) && (75) \end{aligned}$$

We want to compute the posterior $p(\mathbf{w}|D)$ and the evidence $p(D)$. ADF was applied to this problem by Bernardo & Giron (1988) and Stephens (1997).

Break the joint distribution $p(D, \mathbf{w})$ into $n+1$ terms as given by (75). The parameter vector $\mathbf{w}$ must lie in the simplex $\sum_k w_k = 1$, so a Dirichlet distribution is an appropriate approximation to the posterior. The approximate posterior will have the form

$$q(\mathbf{w}) = \frac{\Gamma(\sum_k a_k)}{\prod_k \Gamma(a_k)} \prod_k w_k^{a_k - 1} \qquad (76)$$

## 4.1 ADF

For ADF, we sequence through and incorporate the terms $t_i$ into the approximate posterior. The prior is a Dirichlet distribution, so we initialize with that. To incorporate a data term $t(\mathbf{w})$, take the exact posterior

$$\hat{p}(\mathbf{w}) = \frac{t(\mathbf{w})q(\mathbf{w})}{\int_{\mathbf{w}} t(\mathbf{w})q(\mathbf{w})d\mathbf{w}} \qquad (77)$$

and minimize the KL-divergence $D(\hat{p}(\mathbf{w})||q^{new}(\mathbf{w}))$ subject to the constraint that $q^{new}(\mathbf{w})$ is a Dirichlet distribution. Zeroing the gradient with respect to $a_k^{new}$ gives the conditions

$$\Psi(a_k^{new}) - \Psi(\sum_j a_j^{new}) = \int_{\mathbf{w}} \hat{p}(\mathbf{w}) \log(w_k) d\mathbf{w} \qquad k = 1, ..., K \qquad (78)$$

which as usual are expectation constraints:

$$E_{q^{new}}[\log(w_k)] = E_{\hat{p}}[\log(w_k)] \qquad k = 1, ..., K \qquad (79)$$

For the Dirichlet distribution we see that the characteristic expectations are $E[\log(w_k)]$.

How do we solve for $a^{new}$? Given the values $E_{\hat{p}}[\log(w_k)]$, Newton's method is very effective, and EM-like algorithms are also possible (Minka, 2000a). To compute the expectations, use integration by parts to find

$$\begin{aligned} Z(\mathbf{a}) &= \int_{\mathbf{w}} t(\mathbf{w}) q(\mathbf{w}) d\mathbf{w} && (80) \\ E_{\hat{p}}[\log(w_k)] &= \Psi(a_k) - \Psi(\sum_j a_j) + \nabla_{a_k} \log Z(\mathbf{a}) && (81) \end{aligned}$$

This is a property of the Dirichlet distribution, valid for any $t(\mathbf{w})$. In the mixture weight problem, we have

$$\begin{aligned} t(\mathbf{w}) &= \sum_k p_k(x) w_k && (82) \\ Z(\mathbf{a}) &= \sum_k p_k(x) E_q[w_k] = \frac{\sum_k p_k(x) a_k}{\sum_k a_k} && (83) \\ \nabla_{a_k} \log Z(\mathbf{a}) &= \left( \frac{p_k(x)}{\sum_j a_j} - \frac{\sum_j p_j(x) a_j}{(\sum_j a_j)^2} \right) / Z(\mathbf{a}) && (84) \\ &= \frac{p_k(x)}{\sum_j p_j(x) a_j} - \frac{1}{\sum_j a_j} && (85) \end{aligned}$$

So the final ADF algorithm is

1. Initialize $\mathbf{a} = \mathbf{1}$ (the prior). Initialize $s = 1$ (the scale factor).
2. For each data point $x_i$ in turn, solve the equations

   $$\Psi(a_k^{new}) - \Psi(\sum_j a_j^{new}) = \Psi(a_k) - \Psi(\sum_j a_j) + \frac{p_k(x_i)}{\sum_j p_j(x_i) a_j} - \frac{1}{\sum_j a_j} \qquad k = 1, ..., K \quad (86)$$

   to get $\mathbf{a}^{new}$. Update $s$ via

   $$s^{new} = s \times Z_i(\mathbf{a}) = s \frac{\sum_j p_j(x_i) a_j}{\sum_j a_j} \quad (87)$$

Bernardo & Giron (1988) and Stephens (1997) show that this algorithm performs better than simpler rules like Directed Decision and Probabilistic Teacher. With a little extra work we can get an EP algorithm which is even better.

## 4.2 EP

Divide the new posterior by the old to get the termwise approximation

$$\tilde{t}(\theta) = Z \frac{q^{new}(\theta)}{q(\theta)} \quad (88)$$

$$= Z\frac{\Gamma(\sum_k a_k^{new})}{\prod_k \Gamma(a_k^{new})}\frac{\prod_k \Gamma(a_k)}{\Gamma(\sum_k a_k)}\prod_k w_k^{b_k} \tag{89}$$

$$b_k = a_k^{new} - a_k \tag{90}$$

The EP algorithm is:

1. The term approximations have the form

$$\tilde{t}_i(\theta) = s_i \prod_k w_k^{b_{ik}} \tag{91}$$

Initialize the prior term to itself:

$$\mathbf{b}_0 = \mathbf{0} \tag{92}$$

$$s_0 = 1 \tag{93}$$

Initialize the data terms to 1:

$$\mathbf{b}_i = \mathbf{0} \tag{94}$$

$$s_i = 1 \tag{95}$$

2. $a_k^{new} = 1 + \sum_{i=0}^n b_{ik} = 1$
3. Until all $(\mathbf{b}_i, s_i)$ converge:

(a) Choose the next data term in sequence, cycling back to 1

(b) Remove $\tilde{t}_i$ from the posterior to get an 'old' posterior:

$$\mathbf{a} = \mathbf{a}^{new} - \mathbf{b}_i \tag{96}$$

(c) Recompute $\mathbf{a}^{new}$ from $\mathbf{a}$ by solving

$$\Psi(a_k^{new}) - \Psi(\sum_j a_j^{new}) = \Psi(a_k) - \Psi(\sum_j a_j) + \frac{p_k(x_i)}{\sum_j p_j(x_i)a_j} - \frac{1}{\sum_j a_j} \qquad k = 1, ..., K \tag{97}$$

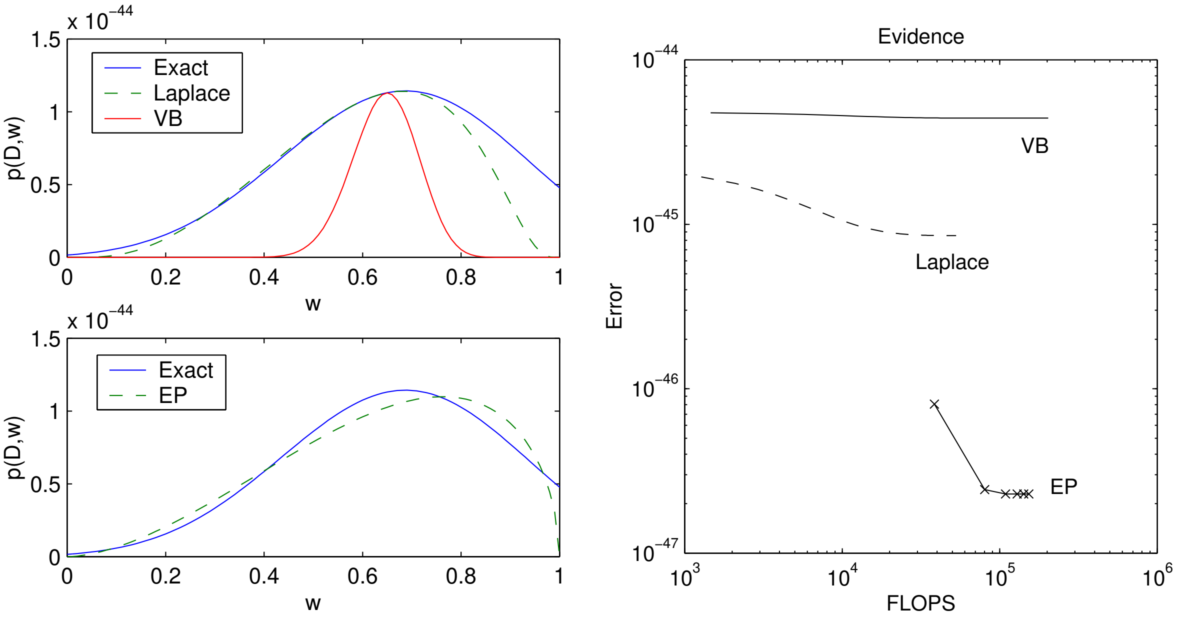
(d) Set

$$\mathbf{b}_i = \mathbf{a}^{new} - \mathbf{a} \tag{98}$$

$$s_i = Z_i(\mathbf{a})\frac{\Gamma(\sum_k a_k^{new})}{\prod_k \Gamma(a_k^{new})}\frac{\prod_k \Gamma(a_k)}{\Gamma(\sum_k a_k)} \tag{99}$$

4. Compute the scale factor:

$$p(D) \approx \frac{\prod_k \Gamma(a_k^{new})}{\Gamma(\sum_k a_k^{new})}\prod_{i=0}^n s_i \tag{100}$$



(a) (b)

Figure 6: (a) Exact posterior vs. approximations, scaled by evidence $p(D)$. The variational Bayes approximation is a lower bound while the others are not. (b) Cost vs. accuracy. ADF is the first 'x' on the EP curve.

## 4.3 Results and comparisons

EP is compared against Laplace's method and variational Bayes. For Laplace's method, we use a softmax parameterization (MacKay, 1998), which makes the approximate posterior a logistic-normal distribution (Aitchison & Shen, 1980). For variational Bayes, we use a Jensen bound as before, to get a Dirichlet approximation.

Figure 6 shows a typical example with $\mathbf{w} = [0.5, 0.5]$, $n = 50$. The component densities were two closely spaced Gaussians:

$$p_1(x) \sim \mathcal{N}(0, 3) \tag{101}$$

$$p_2(x) \sim \mathcal{N}(1, 3) \tag{102}$$

The differences in accuracy on this problem are quite dramatic, as evident from the posterior approximations. However, EP has significantly higher cost, because of the inner optimization required at each step.

# 5 Factorized approximations and belief propagation

This section shows that belief propagation, applied to arbitrary graphs, is a special case of Expectation Propagation when the posterior is approximated as completely factorized. This is interesting because belief propagation is normally considered distinct from techniques which compute approximate models. The equivalence also suggests how to improve both algorithms.

First consider the ADF case, where a completely factorized approximation was studied by Boyen & Koller (1998b). If the components of the $\theta$ vector are $\theta_1, ..., \theta_K$, then a completely factorized posterior has the form

$$q(\theta) = \prod_k q_k(\theta_k) \tag{103}$$

To incorporate a term $t(\theta)$, take the exact posterior

$$\hat{p}(\theta) = \frac{t(\theta)q(\theta)}{\int_\theta t(\theta)q(\theta)d\theta} \tag{104}$$

and minimize the KL-divergence $D(\hat{p}(\theta)||q^{new}(\theta))$ subject to the constraint that $q^{new}(\theta)$ is completely factorized. Zeroing the gradient with respect to $q_k^{new}$ gives the conditions

$$q_k^{new}(y) \quad = \quad \int_{\theta|\theta_k=y} \hat{p}(\theta)d\theta \qquad k = 1, ..., K \tag{105}$$

which as usual are expectation constraints:

$$E_{q^{new}}[\delta(\theta_k - y)] \quad = \quad E_{\hat{p}}[\delta(\theta_k - y)] \qquad k = 1, ..., K, \text{ all values of } y \tag{106}$$

This means the marginals of $q^{new}$ and $\hat{p}$ must match. Thus we arrive at the algorithm of Boyen & Koller (1998b):

1. Initialize $q_k(\theta_k) = 1$ and $s = 1$
2. For each term $t_i(\theta)$ in turn, set $q_k^{new}$ to the $k$th marginal of $\hat{p}$. Update $s$ via

$$s^{new} = s \times \int_\theta t_i(\theta)q(\theta)d\theta \tag{107}$$

While it appears that we have circumvented the exponential family, in practice we must still use it. The marginals of $\hat{p}$ are feasibly computed and stored only if $\theta$ is discrete or $\hat{p}$ is Gaussian. In the first case, $q_k(\theta_k)$ is a multinomial distribution, which is a member of the exponential family, and in the second case, $q_k(\theta_k)$ is Gaussian.

Now we turn this into an EP algorithm. From the ratio $q^{new}/q$, we see that the approximate terms $\tilde{t}_i(\theta)$ are completely factorized. The EP algorithm is thus:

1. The term approximations have the form

$$\tilde{t}_i(\theta) = s_i \prod_k \tilde{t}_{ik}(\theta_k) \tag{108}$$

   Initialize $\tilde{t}_i(\theta) = 1$.

2. Compute the posterior for $\theta$ from the product of $\tilde{t}_i$:

$$q_k^{new}(\theta_k) \propto \prod_i \tilde{t}_{ik}(\theta_k) \tag{109}$$

3. Until all $\tilde{t}_i$ converge:

   (a) Choose a $\tilde{t}_i$ to refine

   (b) Remove $\tilde{t}_i$ from the posterior. For all $k$:

$$\begin{aligned} q_k(\theta_k) &\propto \frac{q_k^{new}(\theta_k)}{\tilde{t}_{ik}(\theta_k)} && (110) \\ &\propto \prod_{j \neq i} \tilde{t}_{jk}(\theta_k) && (111) \end{aligned}$$

   (c) Recompute $q^{new}(\theta)$ from $q(\theta)$. For all $k$ and $y$:

$$\begin{aligned} q_k^{new}(y) &\propto \int_\theta \delta(\theta_k - y) t_i(\theta) q(\theta) && (112) \\ Z_i &= \int_\theta t_i(\theta) q(\theta) d\theta && (113) \end{aligned}$$

   (d) Set

$$\begin{aligned} \tilde{t}_{ik}(y) &= \frac{\int_\theta \delta(\theta_k - y) t_i(\theta) q(\theta)}{q_k(y)} && (114) \\ &= \int_\theta \delta(\theta_k - y) t_i(\theta) \prod_{j \neq k} q_j(\theta_j) && (115) \\ s_i &= Z_i && (116) \end{aligned}$$

4. Compute the scale factor:

$$p(D) \approx \prod_k \left( \int_{\theta_k} \prod_i \tilde{t}_{ik}(\theta_k) d\theta_k \right) \prod_i s_i \tag{117}$$

To make this equivalent to belief propagation (see e.g. Murphy et al. (1999)), the original terms $t_i$ should correspond to the conditional probability tables of a directed belief network. That is, we should break the joint distribution $p(D, \theta)$ into

$$p(D, \theta) = \prod_k p(\theta_k | \mathrm{pa}(\theta_k)) \prod_i p(\mathbf{x}_i | \mathrm{pa}(\mathbf{x}_i)) \tag{118}$$

where $\mathrm{pa}(X)$ is the set of parents of node $X$. The quantities in EP now have the following interpretations:

- $q_k^{new}(\theta_k)$ is the belief state of node $\theta_k$, i.e. the product of all messages into $\theta_k$.
- The 'old' posterior $q_k(\theta_k)$ for a particular term $i$ is the product of messages into $\theta_k$ except for those originating from term $i$.
- When $i \neq k$, the function $\tilde{t}_{ik}(\theta_k)$ is the message that node $i$ (either hidden or observed) sends to its parent $\theta_k$ in belief propagation. For example, suppose node $i$ is hidden. The other parents send $q_j(\theta_j)$, and the child combines them with $q_i(\theta_i)$ to get

$$\tilde{t}_{ik}(y) = \int_\theta \delta(\theta_k - y) p(\theta_i|\text{pa}(\theta_i)) q_i(\theta_i) \prod_{\text{other parents } j} q_j(\theta_j) \tag{119}$$

- When node $i$ is hidden, the function $\tilde{t}_{ii}(\theta_i)$ is a combination of messages sent to node $i$ from its parents in belief propagation. Each parent sends $q_j(\theta_j)$, and the child combines them according to

$$\tilde{t}_{ii}(\theta_i) = \int_{\text{pa}(\theta_i)} p(\theta_i|\text{pa}(\theta_i)) \prod_{\text{parents } j} q_j(\theta_j) \tag{120}$$

This proves the equivalence for directed networks. Equivalence also holds for undirected networks, if each term $t_i$ corresponds to a clique in the graph—compare e.g. the equations in Yedidia et al. (2000).

The remaining difference between belief propagation and EP is that belief propagation does not specify the order in which messages are sent. In the EP algorithm above, each node must exchange messages with all of its parents at each step. However, we can relax this condition by considering a simple variation of EP, where we do not minimize KL-divergence completely at each step, but only partially. Specifically, we only update one of the $\tilde{t}_{ik}$ functions each time. With this variation, we can pass messages in any order.

The equivalence with belief propagation tells us two things about EP:

- Even though EP is derived as an approximation, in some cases the final expectations may be exact. We know this because the marginals resulting from belief propagation are exact on tree-structured networks.
- EP may not converge for some models and/or approximating densities. We know this because belief propagation does not always converge on graphs with loops.

The non-convergence of EP can be avoided in two ways. One way is to improve the quality of the approximating density. For example, Yedidia et al. (2000) have shown that clustering nodes in a belief network can encourage belief propagation to converge (as well as improve its accuracy). Clustering can be used in any EP algorithm, by redefining the terms $t_i$. Another option with EP is to use an approximating density which is not completely factorized, such as a Markov chain or tree.

A second way to improve convergence is to devise a better iteration scheme. Analogous to belief propagation, EP can be derived by minimizing a free energy-like objective. The expression is similar to that of Yedidia et al. (2000) but involving expectation constraints. Different EP iterations can be found by applying different optimization techniques to this objective. (The same applies to belief propagation.) This derivation will appear in a separate paper.

# 6 Classification using the Bayes Point

This section applies Expectation Propagation to inference in the Bayes Point Machine. Using a full-covariance Gaussian approximation to the posterior, we get an algorithm essentially identical to what Opper & Winther (2000c) derived by statistical physics methods. However, the EP derivation yields additional benefits:

- EP provides an estimate of the evidence, $p(D)$, which is not available in Opper & Winther's method.
- EP uses a different optimization scheme which is computationally cheaper and has better convergence properties.
- EP can use a different approximation, such as a spherical Gaussian, to reduce computation, without falling back to naive mean-field theory.

The Bayes Point Machine is simply a Bayesian approach to linear classification. A linear classifier classifies a point $\mathbf{x}$ according to $y = \mathrm{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x})$ for some parameter vector $\mathbf{w}$ (the two classes are $y = \pm 1$). Given a training set $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, the likelihood for $\mathbf{w}$ can be written

$$p(D|\mathbf{w}) = \prod_i p(y_i|\mathbf{x}_i, \mathbf{w}) = \prod_i \Theta(y_i \mathbf{w}^{\mathrm{T}} \mathbf{x}_i) \tag{121}$$

$$\Theta(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases} \tag{122}$$

This is a slight abuse of notation since $p(D|\mathbf{w})$ is only a distribution for $y$, not $\mathbf{x}$. This likelihood is 1 if $\mathbf{w}$ is a perfect separator and 0 otherwise. If linear separation is not possible, then a simple solution is to enlarge the set of features, e.g. by computing squares and third powers, until the data is separable in the new feature space. This has the effect of producing a nonlinear decision boundary in the original measurement space. It can be implemented efficiently using the 'kernel trick', where we rewrite the algorithm in terms of inner products.

Another thing we can do is admit the possibility of errors in labeling or measurement. A labeling error rate of $\epsilon$ can be modeled using (Opper & Winther, 2000b)

$$\begin{aligned} p(y|\mathbf{x}, \mathbf{w}) &= \epsilon(1 - \Theta(y\mathbf{w}^{\mathrm{T}}\mathbf{x})) + (1 - \epsilon)\Theta(y\mathbf{w}^{\mathrm{T}}\mathbf{x}) \qquad (123) \\ &= \epsilon + (1 - 2\epsilon)\Theta(y\mathbf{w}^{\mathrm{T}}\mathbf{x}) \qquad (124) \end{aligned}$$

This approach to modeling errors can automatically reject outliers, whereas approaches based on adding 'slack' to the boundary remain sensitive to outliers. Now the likelihood $p(D|\mathbf{w})$ for a given $\mathbf{w}$ will be $\epsilon^r(1-\epsilon)^{n-r}$, where $r$ is the number of errors on the training set.

To complete our Bayesian model, we need to specify a prior on $\mathbf{w}$. Since the magnitude of $\mathbf{w}$ is irrelevant for classification, some authors have restricted $\mathbf{w}$ to the unit sphere in $d$ dimensions, and have given $\mathbf{w}$ a uniform distribution on the sphere (Rujan, 1997). A more tractable alternative is to leave the magnitude of $\mathbf{w}$ unconstrained, but with a prior distribution which is uniform conditional on any magnitude. One such distribution is the standard normal:

$$p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d) \qquad (125)$$

Conditional on any radius, $\mathbf{w}$ is uniformly distributed on the sphere of that radius, so no particular decision boundary is favored.

Given this model, the optimal way to classify a new data point $\mathbf{x}$ is to use the predictive distribution for $y$ given the training data:

$$p(y|\mathbf{x}, D) = \int_{\mathbf{w}} p(y|\mathbf{x}, \mathbf{w})p(\mathbf{w}|D)d\mathbf{w} \qquad (126)$$

However, this requires solving an integral each time. A practical substitute is to use a single value of $\mathbf{w}$ which approximates the predictive distribution as well as possible. Watkin (1993) and Rujan (1997) have defined the 'Bayes point' as the single best $\mathbf{w}$. However, this point is difficult to find. Thus, following Rujan (1997), we use the posterior mean for $\mathbf{w}$, $E[\mathbf{w}|D]$. Following convention, we will be sloppy and also call this the 'Bayes point'. Our strategy will be to make a Gaussian approximation to the posterior and use its mean as the estimated Bayes point. The scale factor of the posterior will also be useful because, as an estimate of $p(D)$, it allows us to choose among different models.

## 6.1 ADF

Start with the ADF updates. ADF was previously applied to the Bayes Point Machine by Csato et al. (1999). Divide the joint distribution $p(D, \mathbf{w})$ into $n$ terms, one for each data point. Since $y$ and $\mathbf{x}$ always appear multiplied together, the formulas will drop $y$ and assume that $\mathbf{x}_i$ is already scaled by $y_i$. The posterior approximation will have the form

$$q(\mathbf{w}) \sim \mathcal{N}(\mathbf{m}_w, \mathbf{V}_w) \qquad (127)$$

From minimizing the KL-divergence we get the expectation constraints

$$E_{q^{new}}[\mathbf{w}] = E_{\hat{p}}[\mathbf{w}] \tag{128}$$

$$E_{q^{new}}[\mathbf{w}\mathbf{w}^{\mathrm{T}}] = E_{\hat{p}}[\mathbf{w}\mathbf{w}^{\mathrm{T}}] \tag{129}$$

To compute these expectations, use the following relations (obtained from integration-by-parts):

$$Z(\mathbf{m}_w, \mathbf{V}_w) = \int_{\mathbf{w}} t(\mathbf{w})q(\mathbf{w})\, d\mathbf{w} \tag{130}$$

$$E_{\hat{p}}[\mathbf{w}] = \mathbf{m}_w + \mathbf{V}_w \nabla_m \log Z(\mathbf{m}_w, \mathbf{V}_w) \tag{131}$$

$$E_{\hat{p}}[\mathbf{w}\mathbf{w}^{\mathrm{T}}] - E_{\hat{p}}[\mathbf{w}]E_{\hat{p}}[\mathbf{w}]^{\mathrm{T}} = \mathbf{V}_w - \mathbf{V}_w(\nabla_m \nabla_m^{\mathrm{T}} - 2\nabla_v \log Z(\mathbf{m}_w, \mathbf{V}_w))\mathbf{V}_w \tag{132}$$

These hold for any $t(\mathbf{w})$. For the Bayes Point Machine, we have (combining $y\mathbf{x}$ into $\mathbf{x}$):

$$t(\mathbf{w}) = \epsilon + (1-2\epsilon)\Theta(\mathbf{w}^{\mathrm{T}}\mathbf{x}) \tag{133}$$

$$\phi(z) = \int_{-\infty}^{z} \mathcal{N}(z; 0, 1) dz \tag{134}$$

$$z = \frac{\mathbf{m}_w^{\mathrm{T}}\mathbf{x}}{\sqrt{\mathbf{x}^{\mathrm{T}}\mathbf{V}_w\mathbf{x}}} \tag{135}$$

$$Z(\mathbf{m}_w, \mathbf{V}_w) = \epsilon + (1-2\epsilon)\phi(z) \tag{136}$$

$$\alpha = \frac{1}{\sqrt{\mathbf{x}^{\mathrm{T}}\mathbf{V}_w\mathbf{x}}} \frac{(1-2\epsilon)\mathcal{N}(z; 0, 1)}{\epsilon + (1-2\epsilon)\phi(z)} \tag{137}$$

$$\nabla_m \log Z(\mathbf{m}_w, \mathbf{V}_w) = \alpha\mathbf{x} \tag{138}$$

$$\nabla_v \log Z(\mathbf{m}_w, \mathbf{V}_w) = -\frac{1}{2}\frac{\alpha\mathbf{m}_w^{\mathrm{T}}\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{V}_w\mathbf{x}}\mathbf{x}\mathbf{x}^{\mathrm{T}} \tag{139}$$

$$\nabla_m \nabla_m^{\mathrm{T}} - 2\nabla_v \log Z(\mathbf{m}_w, \mathbf{V}_w) = \alpha^2\mathbf{x}\mathbf{x}^{\mathrm{T}} + \frac{\alpha\mathbf{m}_w^{\mathrm{T}}\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{V}_w\mathbf{x}}\mathbf{x}\mathbf{x}^{\mathrm{T}} \tag{140}$$

$$= \frac{\alpha(\mathbf{m}_w + \mathbf{V}_w\alpha\mathbf{x})^{\mathrm{T}}\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{V}_w\mathbf{x}}\mathbf{x}\mathbf{x}^{\mathrm{T}} \tag{141}$$

The ADF algorithm is thus:

1. Initialize $\mathbf{m}_w = \mathbf{0}$, $\mathbf{V}_w = \mathbf{I}$ (the prior). Initialize $s = 1$ (the scale factor).
2. For each data point $\mathbf{x}_i$, update $(\mathbf{m}_w, \mathbf{V}_w, s)$ according to

$$\mathbf{m}_w^{new} = \mathbf{m}_w + \mathbf{V}_w\alpha_i\mathbf{x}_i \tag{142}$$

$$\mathbf{V}_w^{new} = \mathbf{V}_w - (\mathbf{V}_w\mathbf{x}_i)\left(\frac{\alpha_i\mathbf{x}_i^{\mathrm{T}}\mathbf{m}_w^{new}}{\mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i}\right)(\mathbf{V}_w\mathbf{x}_i)^{\mathrm{T}} \tag{143}$$

$$s^{new} = s \times Z_i(\mathbf{m}_w, \mathbf{V}_w) \tag{144}$$

The final $\mathbf{m}_w$ is the estimate of $\mathbf{w}$ used for classification. Note that this algorithm does not require matrix inversion. It is the same algorithm obtained by Csato et al. (1999), except they expressed it in terms of kernel inner products, which makes it more complex.

## 6.2 EP

Now we turn the ADF algorithm into an EP algorithm. By dividing two consecutive Gaussian posteriors, we find that the term approximations are also Gaussian, parameterized by $\mathbf{m}_i$ and $\mathbf{V}_i$:

$$\begin{aligned}
\tilde{t}(\mathbf{w}) &= Z\frac{q^{new}(\mathbf{w})}{q(\mathbf{w})} && (145) \\
&= \frac{Z}{\mathcal{N}(\mathbf{m}_i; \mathbf{m}_w; \mathbf{V}_i + \mathbf{V}_w)}\mathcal{N}(\mathbf{w}; \mathbf{m}_i, \mathbf{V}_i) && (146) \\
\mathbf{m}_w^{new} &= \mathbf{V}_w^{new}(\mathbf{V}_i^{-1}\mathbf{m}_i + \mathbf{V}_w^{-1}\mathbf{m}_w) && (147) \\
\mathbf{V}_w^{new} &= (\mathbf{V}_i^{-1} + \mathbf{V}_w^{-1})^{-1} && (148)
\end{aligned}$$

To simplify this, combine it with the ADF updates (142,143) to solve directly for $(\mathbf{m}_i, \mathbf{V}_i)$:

$$\begin{aligned}
\mathbf{V}_i &= \left(\frac{\alpha_i \mathbf{x}_i^{\mathrm{T}}\mathbf{m}_w^{new}}{\mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}}\right)^{-1} - \mathbf{V}_w && (149) \\
\mathbf{m}_i &= \mathbf{m}_w + (\mathbf{V}_i + \mathbf{V}_w)\alpha_i\mathbf{x}_i && (150)
\end{aligned}$$

Equation (149) involves the inverse of a singular matrix, which tells us something special about $\mathbf{V}_i$: only the projection along $\mathbf{x}_i$ will be finite. This makes perfect sense, because term $i$ only constrains the projection of $\mathbf{w}$ along $\mathbf{x}_i$. This special structure allows us to represent $\mathbf{V}_i$ with a scalar, $v_i$:

$$\begin{aligned}
\mathbf{V}_i^{-1} &= v_i^{-1}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}} && (151) \\
\mathbf{x}_i^{\mathrm{T}}\mathbf{V}_i\mathbf{x}_i &= v_i && (152)
\end{aligned}$$

From (149), we know that

$$\mathbf{x}_i^{\mathrm{T}}\mathbf{V}_i\mathbf{x}_i = \frac{\mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i}{\alpha_i\mathbf{x}_i^{\mathrm{T}}\mathbf{m}_w^{new}} - \mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i \tag{153}$$

which means the update for $v_i$ is

$$v_i = \mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i\left(\frac{1}{\alpha_i\mathbf{x}_i^{\mathrm{T}}\mathbf{m}_w^{new}} - 1\right) \tag{154}$$

Another consequence of this special structure is that instead of the full vector $\mathbf{m}_i$, we only need the projection $\mathbf{m}_i^{\mathrm{T}}\mathbf{x}_i$, which can be stored as a scalar $m_i$. Now we can write an efficient EP algorithm:

1. Except for the prior, the term approximations have the form

$$\tilde{t}_i(\mathbf{w}) = s_i \exp(-\frac{1}{2v_i}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - m_i)^2) \tag{155}$$

Initialize them to 1:

$$
\begin{aligned}
v_i &= \infty && (156)\\
m_i &= 0 && (157)\\
s_i &= 1 && (158)
\end{aligned}
$$

2. $\mathbf{m}_w^{new} = \mathbf{0}, \mathbf{V}_w^{new} = \mathbf{I}$ (the prior)
3. Until all $(m_i, v_i, s_i)$ converge:
   (a) Choose the next data term in sequence, cycling back to 1
   (b) Remove $\tilde{t}_i$ from the posterior to get an 'old' posterior:

$$
\begin{aligned}
\mathbf{V}_w &= ((\mathbf{V}_w^{new})^{-1} - v_i^{-1}\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}})^{-1} && (159)\\
&= \mathbf{V}_w^{new} + (\mathbf{V}_w^{new}\mathbf{x}_i)(v_i - \mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w^{new}\mathbf{x}_i)^{-1}(\mathbf{V}_w^{new}\mathbf{x}_i)^{\mathrm{T}} && (160)\\
\mathbf{m}_w &= \mathbf{m}_w^{new} + \mathbf{V}_w\mathbf{V}_i^{-1}(\mathbf{m}_w^{new} - \mathbf{m}_i) && (161)\\
&= \mathbf{m}_w^{new} + (\mathbf{V}_w\mathbf{x}_i)v_i^{-1}(\mathbf{x}_i^{\mathrm{T}}\mathbf{m}_w^{new} - m_i) && (162)
\end{aligned}
$$

   These expressions involving $\mathbf{V}_w$ can be computed efficiently:

$$
\begin{aligned}
\mathbf{V}_w\mathbf{x}_i &= (\mathbf{V}_w^{new}\mathbf{x}_i)\frac{v_i}{v_i - \mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w^{new}\mathbf{x}_i} && (163)\\
\mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i &= \left(\frac{1}{\mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w^{new}\mathbf{x}_i} - \frac{1}{v_i}\right)^{-1} && (164)
\end{aligned}
$$

   (c) Recompute $(\mathbf{m}_w^{new}, \mathbf{V}_w^{new}, Z_i)$ from $(\mathbf{m}_w, \mathbf{V}_w)$, as in ADF.
   (d) Set

$$
\begin{aligned}
v_i &= \mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i\left(\frac{1}{\alpha_i\mathbf{x}_i^{\mathrm{T}}\mathbf{m}_w^{new}} - 1\right) && (165)\\
m_i &= \mathbf{x}_i^{\mathrm{T}}\mathbf{m}_w + (v_i + \mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i)\alpha_i && (166)\\
s_i &= Z_i\frac{|\mathbf{V}_i + \mathbf{V}_w|^{1/2}}{|\mathbf{V}_i|^{1/2}}\exp(\frac{1}{2}(\mathbf{m}_i - \mathbf{m}_w)^{\mathrm{T}}(\mathbf{V}_i + \mathbf{V}_w)^{-1}(\mathbf{m}_i - \mathbf{m}_w)) && (167)\\
&= Z_i\sqrt{1 + v_i^{-1}\mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i}\exp(\frac{1}{2}\frac{\mathbf{x}_i^{\mathrm{T}}\mathbf{V}_w\mathbf{x}_i}{\mathbf{x}_i^{\mathrm{T}}\mathbf{m}_w^{new}}\alpha_i) && (168)
\end{aligned}
$$

4. Compute the scale factor:

$$
\begin{aligned}
B &= (\mathbf{m}_w^{new})^{\mathrm{T}}(\mathbf{V}_w^{new})^{-1}(\mathbf{m}_w^{new}) - \sum_i \frac{m_i^2}{v_i} && (169)\\
p(D) &\approx |\mathbf{V}_w^{new}|^{1/2}\exp(B/2)\prod_{i=1}^{n} s_i && (170)
\end{aligned}
$$

This algorithm processes each data point in $O(d^2)$ time. Assuming the number of iterations is constant, computing the Bayes point therefore takes $O(nd^2)$ time. Computing the scale factor at the end requires $O(d^3)$ time.

## 6.3 EP with kernels

To use the kernel trick, we need to rewrite EP in terms of inner products. Following the notation of Opper & Winther (2000c), define

$$
\begin{aligned}
\lambda_i &= \mathbf{x}_i^{\mathrm{T}} \mathbf{V}_w^{\setminus i} \mathbf{x}_i \qquad (\mathbf{V}_w^{(i)} \text{ is the } \mathbf{V}_w \text{ for term } i) && (171)\\
C_{ij} &= \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j && (172)\\
\Lambda &= \mathrm{diag}(v_1, ..., v_n) && (173)\\
h_i &= \mathbf{x}_i^{\mathrm{T}} \mathbf{m}_w^{new} && (174)\\
h_i^{\setminus i} &= \mathbf{x}_i^{\mathrm{T}} \mathbf{m}_w^{(i)} && (175)
\end{aligned}
$$

From the definition of $q^{new}(\mathbf{w})$ as the product of approximate terms $\tilde{t}_i(\mathbf{w})$, we know that

$$
\begin{aligned}
\mathbf{V}_w^{new} &= \left(\mathbf{I} + \sum_i \frac{\mathbf{x}_i \mathbf{x}_i^{\mathrm{T}}}{v_i}\right)^{-1} && (176)\\
\mathbf{m}_w^{new} &= \mathbf{V}_w^{new} \sum_j \frac{m_j \mathbf{x}_j}{v_j} && (177)\\
\mathbf{x}_i^{\mathrm{T}} \mathbf{m}_w^{new} &= \sum_j (\mathbf{x}_i^{\mathrm{T}} \mathbf{V}_w^{new} \mathbf{x}_j) \frac{m_j}{v_j} && (178)
\end{aligned}
$$

From the matrix inversion lemma we therefore have

$$
\begin{aligned}
(\mathbf{C} + \Lambda)^{-1} &= \Lambda^{-1} - \Lambda^{-1} \mathbf{X}^{\mathrm{T}} \mathbf{V}_w^{new} \mathbf{X} \Lambda^{-1} && (179)\\
\mathbf{X}^{\mathrm{T}} \mathbf{V}_w^{new} \mathbf{X} &= \Lambda - \Lambda(\mathbf{C} + \Lambda)^{-1} \Lambda && (180)\\
&= (\mathbf{C}^{-1} + \Lambda^{-1})^{-1} && (181)
\end{aligned}
$$

In this way, we can compute $\mathbf{x}_i^{\mathrm{T}} \mathbf{V}_w^{new} \mathbf{x}_j$ for any $(i, j)$ using only $\mathbf{C}$ and $\Lambda$. The EP algorithm becomes:

1. Initialize $v_i = \infty, m_i = 0, s_i = 1$.
2. Initialize $h_i = 0, \lambda_i = C_{ii}$.
3. Until all $(m_i, v_i, s_i)$ converge:
   (a) Choose the next data term in sequence, cycling back to 1

(b) Remove $\tilde{t}_i$ from the posterior to get an 'old' posterior:

$$h_i^{\setminus i} = h_i + \lambda_i v_i^{-1}(h_i - m_i) \tag{182}$$

(c) Recompute (part of) the new posterior, as in ADF:

$$z = \frac{h_i^{\setminus i}}{\sqrt{\lambda_i}} \tag{183}$$
$$Z_i = \epsilon + (1 - 2\epsilon)\phi(z) \tag{184}$$
$$\alpha_i = \frac{1}{\sqrt{\lambda_i}} \frac{(1 - 2\epsilon)\mathcal{N}(z; 0, 1)}{\epsilon + (1 - 2\epsilon)\phi(z)} \tag{185}$$
$$h_i = h_i^{\setminus i} + \lambda_i \alpha_i \tag{186}$$

(d) Set

$$v_i = \lambda_i \left( \frac{1}{\alpha_i h_i} - 1 \right) \tag{187}$$
$$m_i = h_i^{\setminus i} + (v_i + \lambda_i)\alpha_i = h_i + v_i \alpha_i \tag{188}$$
$$s_i = Z_i \sqrt{1 + v_i^{-1}\lambda_i} \exp\left(\frac{\lambda_i \alpha_i}{2h_i}\right) \tag{189}$$

(e) Now that $\Lambda$ is updated, finish recomputing the new posterior:

$$\mathbf{A} = (\mathbf{C}^{-1} + \Lambda^{-1})^{-1} \tag{190}$$

For all $i$:

$$h_i = \sum_j A_{ij} \frac{m_j}{v_j} \qquad \text{(from (178))} \tag{191}$$
$$\lambda_i = \left( \frac{1}{A_{ii}} - \frac{1}{v_i} \right)^{-1} \qquad \text{(from (164))} \tag{192}$$

4. Compute the scale factor:

$$B = \sum_{ij} A_{ij} \frac{m_i m_j}{v_i v_j} - \sum_i \frac{m_i^2}{v_i} \tag{193}$$
$$p(D) \approx \frac{|\mathbf{C} + \Lambda|^{1/2}}{|\Lambda|^{1/2}} \exp(B/2) \prod_{i=1}^{n} s_i \tag{194}$$

It would appear that we need to invert a matrix, taking $O(n^3)$ time, for each data point. However, since only one $v_i$ changes each time, $\mathbf{A}$ can be updated incrementally in $O(n^2)$ time. Assuming a constant number of iterations, the algorithm thus runs in $O(n^3)$ time.

To show the equivalence with Opper & Winther's algorithm, we make the following observations:

- At convergence, (188) will hold for all $i$, so that

$$\sum_j \frac{m_j \mathbf{x}_j}{v_j} = \sum_j \frac{h_j \mathbf{x}_j}{v_j} + \sum_j \alpha_j \mathbf{x}_j \tag{195}$$

$$\mathbf{m}_w^{new} + \sum_j \frac{m_j \mathbf{x}_j}{v_j} = \left(\mathbf{I} + \sum_j \frac{\mathbf{x}_j \mathbf{x}_j^{\mathrm{T}}}{v_j}\right) \mathbf{m}_w^{new} + \sum_j \alpha_j \mathbf{x}_j \tag{196}$$

$$\mathbf{m}_w^{new} = \sum_j \alpha_j \mathbf{x}_j \tag{197}$$

$$h_i = \sum_j \alpha_j (\mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j) = \sum_j C_{ij} \alpha_j \tag{198}$$

  This is the equation Opper & Winther use instead of (191).

- The update (192) for $\lambda_i$ can be written

$$\lambda_i = \left(\frac{1}{v_i - v_i^2 \left[(\mathbf{C} + \Lambda)^{-1}\right]_{ii}} - \frac{1}{v_i}\right)^{-1} \tag{199}$$

$$= (v_i - v_i^2 \left[(\mathbf{C} + \Lambda)^{-1}\right]_{ii})(v_i \left[(\mathbf{C} + \Lambda)^{-1}\right]_{ii})^{-1} \tag{200}$$

$$= \frac{1}{\left[(\mathbf{C} + \Lambda)^{-1}\right]_{ii}} - v_i \tag{201}$$

  This the update used by Opper & Winther instead of (192).

- EP computes a scale factor for the posterior, while Opper & Winther do not.

- All other updates are identical to those in Opper & Winther (2000c), though carried out in a different order.

## 6.4 Results and comparisons

Figure 7 demonstrates the Bayes point classifier vs. the SVM classifier on 3 training points. Each classifier is set to zero noise. The Bayes point classifier approximates a vote between all linear separators, ranging from an angle of 0° to 135°. The Bayes point chooses an angle in the middle of this range. SVM chooses the decision boundary to maximize 'margin', the distance to the nearest data point. This makes it ignore the topmost data point. In fact, the Bayes point and SVM would coincide if that point were removed. Adding the point reduces the set of linear separators, which Bayesian inference must take into account.

Figure 8 plots the situation in parameter space. For viewing convenience, we restrict $\mathbf{w}$ to the unit sphere. The exact Bayes point was computed by likelihood-weighted sampling, i.e.

sampling a random $\mathbf{w}$ on the sphere and discarding those which are not separators. EP provides an accurate estimate of the posterior mean $\mathbf{m}_w$ and posterior covariance $\mathbf{V}_w$. Compare:

$$\text{Exact } \mathbf{V}_w = \begin{bmatrix} 0.366 & 0.0551 & -0.125 \\ 0.0551 & 0.533 & -0.073 \\ -0.125 & -0.073 & 0.164 \end{bmatrix} \qquad \text{EP } \mathbf{V}_w = \begin{bmatrix} 0.358 & 0.0589 & -0.118 \\ 0.0589 & 0.542 & -0.0809 \\ -0.118 & -0.0809 & 0.163 \end{bmatrix}$$

Figure 9 plots cost vs. accuracy for EP versus three other algorithms for estimating the Bayes point: the billiard algorithm of Herbrich et al. (1999), the TAP algorithm of Opper & Winther (2000c), and the mean-field (MF) algorithm of Opper & Winther (2000c). The SVM is also plotted for reference. Its unusually long running time is due to Matlab's `quadprog` solver. TAP and MF were slower to converge than EP, even with a large initial step size of 0.5. As expected, EP and TAP converge to the same solution.

The billiard algorithm is a Monte Carlo method which bounces a ball inside the region defined by hyperplane constraints. It must be initialized with a linear separator and the SVM was used for that purpose. Since there are many other ways one could initialize the billiard, which may be much cheaper, the initialization step was not counted against the billiard's FLOP count (otherwise the billiard curve would be right of the SVM point). The accuracy axis must also be interpreted carefully. While it is tempting to use the lower envelope of the curve as the measure of error, it is actually more accurate to use the upper envelope, since this is all that the algorithm can consistently achieve as samples accumulate. Nevertheless it is clear that EP is much faster and more accurate.

Laplace's method and variational Bayes do not work at all on this problem. Laplace fails because the derivatives of the likelihood are not informative: they are either zero or infinity. Variational Bayes fails for $\epsilon = 0$ because no Gaussian can lower bound the likelihood (Gaussians never reach zero). Even with $\epsilon > 0$, the Gaussian bound must be quite narrow so we can't expect competitive performance.

For higher dimensional problems, it is more difficult to compute the exact Bayes point and make cost vs. accuracy comparisons. So instead we'll measure test performance. Figure 10 compares EP, Billiard, and SVM on discriminating handwritten digit '3' from '5'. Each data point $\mathbf{x}_i$ is an $8 \times 8$ binary image (64 dimensions). The 500-point dataset was randomly split 20 times into a training set of 70 points and a test set of 430 points. All algorithms were linear and set for zero noise. The billiard algorithm was run for 2000 iterations, which is far more computation than EP or SVM used. Nevertheless, EP is best.

The difference between EP and SVM is most dramatic when the number of samples is comparable to the dimensionality. On other standard datasets, the dimensionality is much smaller than the training set size and there is little difference between the algorithms, as observed by Opper & Winther (2000c) and Csato et al. (1999). The `sonar` dataset does have high dimensionality, but it is so easy that again there is no difference.

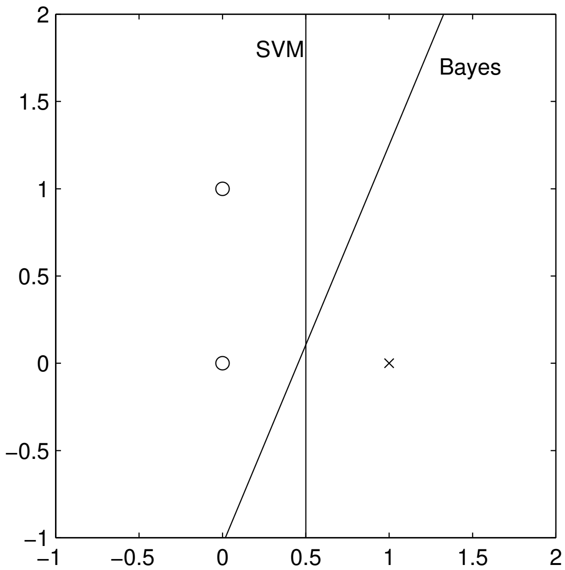



Figure 7: Bayes point machine vs. Support Vector Machine on a simple data set. The Bayes point more closely approximates a vote between all linear separators of the data.

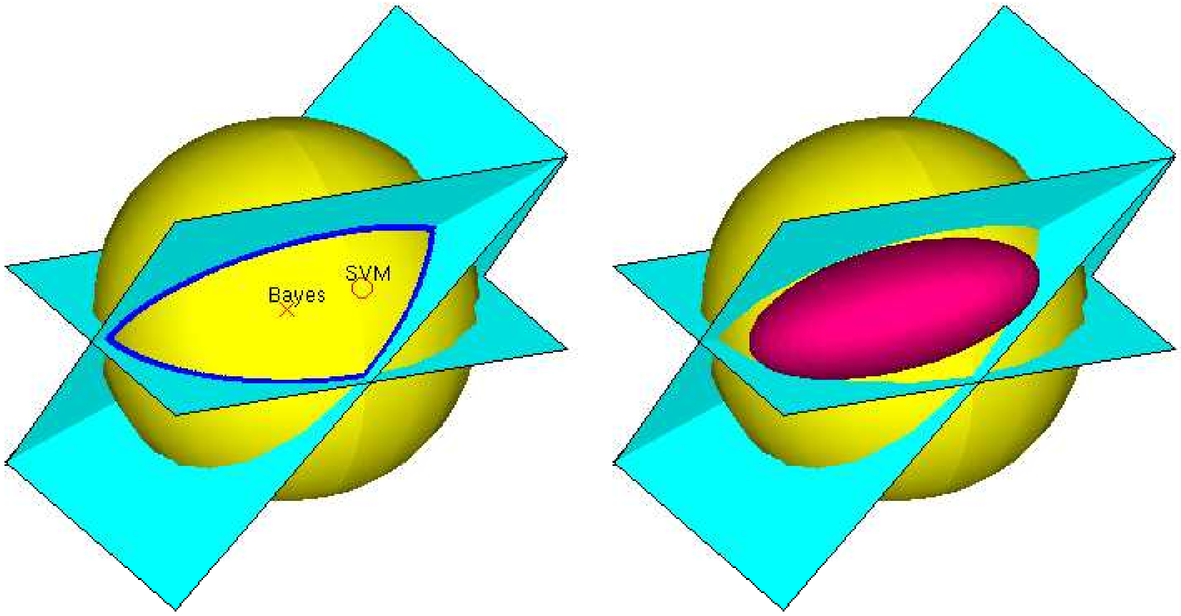



Figure 8: (a) The same problem viewed in parameter space (the unit sphere). Each data point imposes a linear constraint, giving an odd-shaped region on the sphere. The Bayes point is the centroid of the region. The EP estimate is indistinguishable from the exact Bayes point on this plot. (b) The Gaussian posterior approximation obtained by EP, rendered as a $1\sigma$ equiprobable ellipsoid.

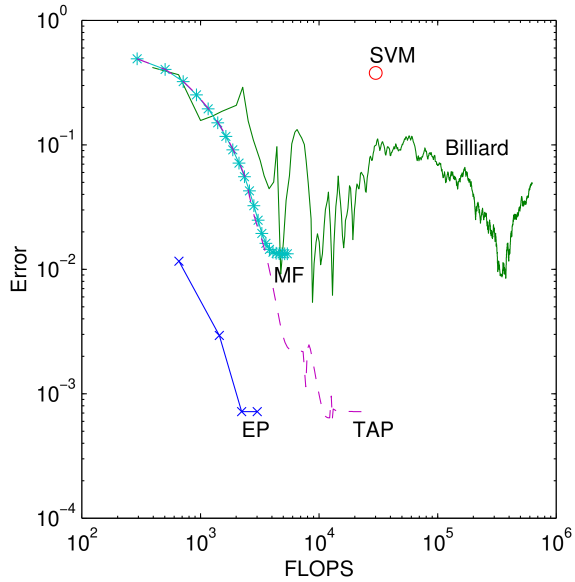



Figure 9: Cost vs. accuracy in estimating the posterior mean for the 3-point problem

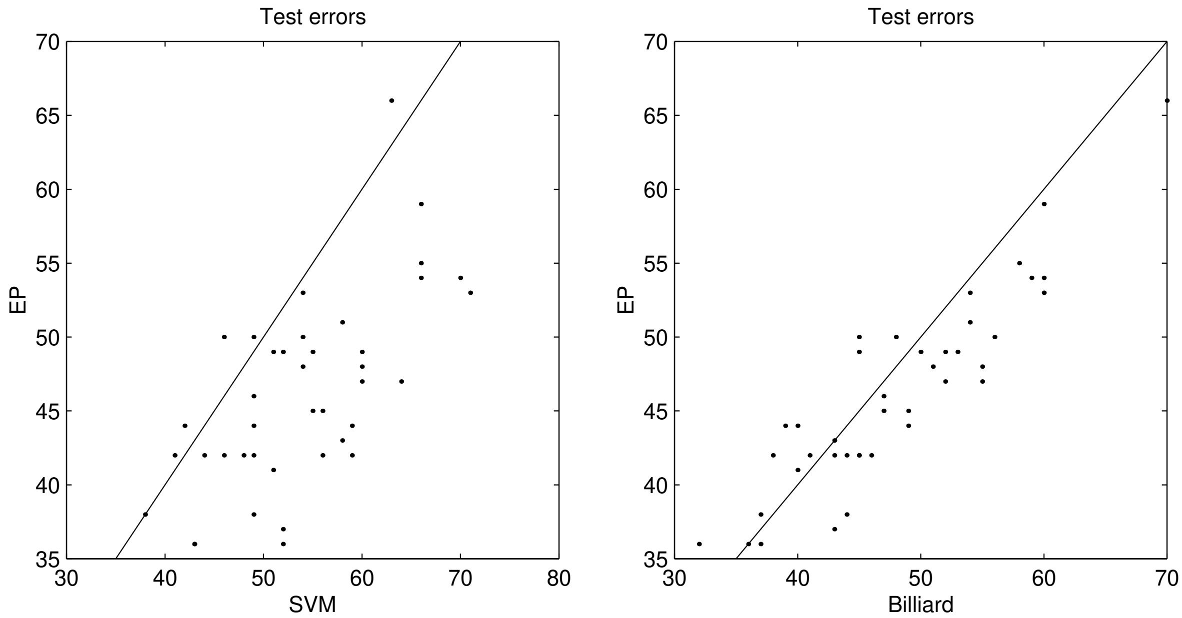



Figure 10: Test error of SVM, EP, and Billiard on digit classification, over 40 random train/test splits. EP beat SVM 34 times and beat Billiard 28 times, despite being the fastest of all three algorithms.

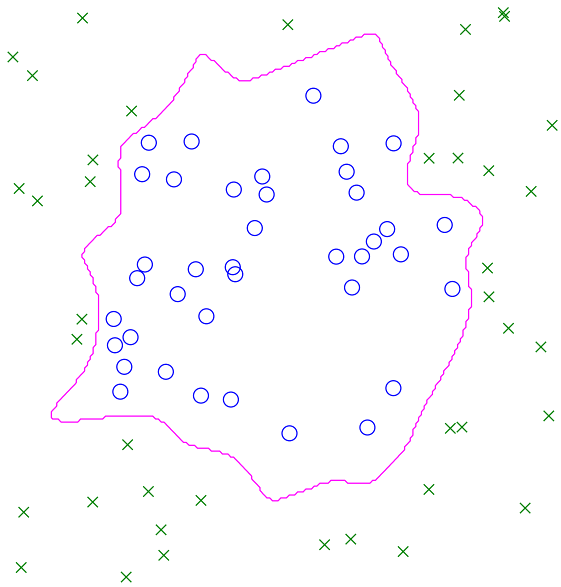
Other experiments involving nonlinear kernels and nonzero noise will be reported in the full dissertation.

## 6.5 Model selection

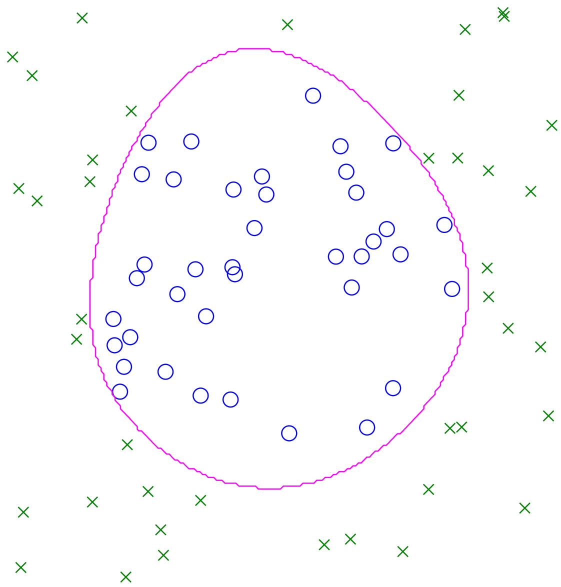
This section reports preliminary results on Bayesian model selection using the estimated evidence $p(D)$. This is a unique feature of EP because the billiard algorithm cannot provide an estimate of $p(D)$. By maximizing the evidence, we can select the appropriate feature expansion or noise parameter $\epsilon$. The SVM approach to model selection is to minimize $\frac{R^2}{\rho^2}$, where $R$ is the radius of the smallest ball containing the training set and $\rho$ is the margin (Cristianini et al., 1998).

These two criteria, evidence and margin, can lead to quite different results. Figure 11 shows a synthetic training set classified according to distance from the origin, i.e. the true decision boundary is a circle. Three different decision boundaries result from training an SVM with three different kernels (the decision boundaries from EP are nearly identical). All of these boundaries achieve zero error so we need to invoke some other rule besides training error to select among them. Over all Gaussian kernels, the margin criterion is minimized by taking $\sigma = 0.08$. This produces a decision boundary with spurious kinks and bumps. By contrast, the best Gaussian kernel according to evidence has $\sigma = 0.6$, providing a much smoother and more realistic boundary. The quadratic kernel, which is the closest to the true boundary, has even greater evidence.

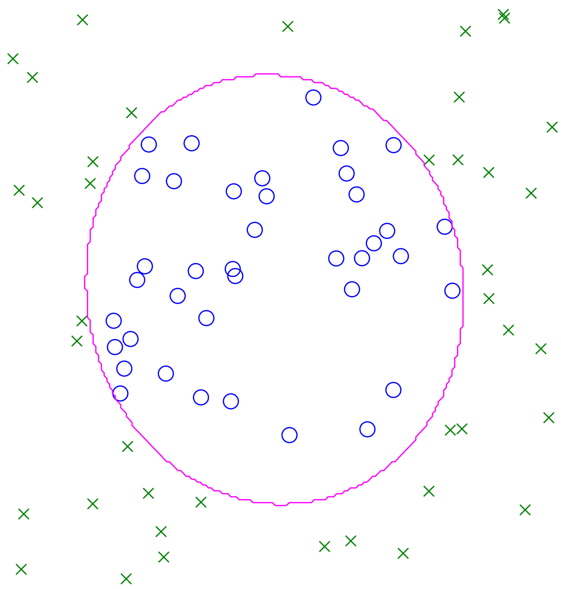
In the noise-free case, the Bayesian evidence $p(D)$ has a natural interpretation. It is the fraction of perfect separators out of all representable classifiers. If a particular set of features has a large percentage of perfect separators, then we intuitively should have more faith in that feature set. This is what Bayesian model selection does, and in choosing the Bayes point we follow the same basic argument. Maximizing the margin does not necessarily behave the same way.



Gaussian $\sigma = 0.08$



Gaussian $\sigma = 0.6$



Quadratic

| Kernel | $R^2/\rho^2$ | $\log(p(D))$ |
|---|---|---|
| $\sigma = 0.08$ | 18 | -39 |
| $\sigma = 0.6$ | 108 | -19 |
| quadratic | 656 | -16 |

Figure 11: A dataset classified with three different kernels. The margin criterion $R^2/\rho^2$ prefers the $\sigma = 0.08$ solution, while the Bayesian evidence $p(D)$ prefers the quadratic solution. The true boundary is a circle.

# References


Aitchison, J., & Shen, S. M. (1980). Logistic-normal distributions: Some properties and uses. *Biometrika*, *67*, 261–272.

Barber, D., & Sollich, P. (1999). Gaussian fields for approximate inference in layered sigmoid belief networks. *NIPS*.

Bernardo, J. M., & Giron, F. J. (1988). A Bayesian analysis of simple mixture problems. *Bayesian Statistics 3* (pp. 67–78).

Boyen, X., & Koller, D. (1998a). Approximate learning of dynamic models. *NIPS*. `http://robotics.Stanford.EDU/~koller/papers/nips98.html`.

Boyen, X., & Koller, D. (1998b). Tractable inference for complex stochastic processes. *Uncertainty in AI*. `http://robotics.Stanford.EDU/~koller/papers/uai98.html`.

Cristianini, N., Campbell, C., & Shawe-Taylor, J. (1998). Dynamically adapting kernels in support vector machines. *NIPS*.

Csato, L., Fokue, E., Opper, M., Schottky, B., & Winther, O. (1999). Efficient approaches to Gaussian process classification. *NIPS*. `http://www.ncrg.aston.ac.uk/cgi-bin/tr_avail.pl?trnumber=NCRG/99/025`.

Gelfand, A. E., & Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *J American Stat Assoc*, *85*, 398–409.

Herbrich, R., Graepel, T., & Campbell, C. (1999). Bayes point machines: Estimating the bayes point in kernel space. *IJCAI Workshop Support Vector Machines* (pp. 23–27). `http://stat.cs.tu-berlin.de/~ralfh/abstract-HerGraeCamp99a.html`.

Koller, D., Lerner, U., & Angelov, D. (1999). A general algorithm for approximate inference and its application to hybrid Bayes nets. *Uncertainty in AI* (pp. 324–333). `http://robotics.Stanford.EDU/~koller/papers/uai99kla.html`.

Kushner, H. J., & Budhiraja, A. S. (2000). A nonlinear filtering algorithm based on an approximation of the conditional distribution. *IEEE Trans Automatic Control*, *45*, 580–585.

Lauritzen, S. L. (1992). Propagation of probabilities, means and variances in mixed graphical association models. *J American Statistical Association*, *87*, 1098–1108.

MacKay, D. J. C. (1998). Choice of basis for laplace approximation. *Machine Learning*, *33*. `http://wol.ra.phy.cam.ac.uk/mackay/abstracts/laplace.html`.

Maybeck, P. S. (1982). *Stochastic models, estimation and control*, chapter 12.7. Academic Press.

Minka, T. P. (2000a). Estimating a dirichlet distribution.
http://vismod.www.media.mit.edu/~tpminka/papers/dirichlet.html.

Minka, T. P. (2000b). Variational bayes for mixture models: Reversing em.
http://vismod.www.media.mit.edu/~tpminka/papers/rem.html.

Murphy, K., Weiss, Y., & Jordan, M. (1999). Loopy-belief propagation for approximate inference: An empirical study. *Uncertainty in AI*.
http://www.cs.berkeley.edu/~murphyk/Papers/loopy.ps.gz.

Opper, M., & Winther, O. (1999). *On-line learning in neural networks*, chapter A Bayesian Approach to On-line Learning. Cambridge University Press.
http://www.ncrg.aston.ac.uk/cgi-bin/tr_avail.pl?trnumber=NCRG/99/029.

Opper, M., & Winther, O. (2000a). Adaptive TAP equations. *Advanced Mean Field Methods - Theory and Practice*. MIT Press.
http://nimis.thep.lu.se/tf2/staff/winther/opper.adaptive.ps.Z.

Opper, M., & Winther, O. (2000b). Gaussian Processes and SVM: Mean field results and leave-one-out. *Advances in Large Margin Classifiers*. MIT Press.
http://nimis.thep.lu.se/tf2/staff/winther/opper.largemargin.ps.Z.

Opper, M., & Winther, O. (2000c). Gaussian processes for classification: Mean field algorithms. *Neural Computation*.
http://nimis.thep.lu.se/tf2/staff/winther/opper.neuralcomp.ps.Z.

Rujan, P. (1997). Perceptron learning by playing billiards. *Neural Computation*, *9*, 99–122.
http://saturn.neuro.uni-oldenburg.de/~rujan/.

Shachter, R. (1990). A linear approximation method for probabilistic inference. *Uncertainty in AI*.

Stephens, M. (1997). *Bayesian methods for mixtures of normal distributions*. Doctoral dissertation, Oxford University. http://www.stats.ox.ac.uk/~stephens/papers.html.

Watkin, T. (1993). Optimal learning with a neural network. *Europhysics Letters*, *21*, 871–876.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2000). *Generalized belief propagation* (Technical Report TR2000-26). MERL. http://www.merl.com/reports/TR2000-26/index.html.