Bayesian Methods for Neural Networks

Readings: Bishop, Neural Networks for Pattern Recognition*. Chapter 10.*

Aaron Courville

Bayesian Inference

We've seen Bayesian inference before, remember

- $p(\theta)$ is the *prior* probability of a parameter θ before having seen the data.
- $p(\mathcal{D}|\theta)$ is called the *likelihood*. It is the probability of the data $\mathcal{D}$ given θ

We can use bayes' rule to determine the *posterior* probability of θ given the data, $\mathcal{D}$,

$$p(\theta|\mathcal{D}) = \frac{p(D|\theta)p(\theta)}{p(\mathcal{D})}$$

In general this will provide an entire distribution over possible values of θ rather that the single most likely value of θ.

Bayesian ANNs?

We can apply this process to neural networks and come up with the probability distribution over the network weights, $\mathbf{w}$, given the training data, $p(\mathbf{w}|\mathcal{D})$.

As we will see, we can also come up with posterior distribution over:

- the network output
- a set of different sized networks
- the outputs of a set of different sized networks

Why should we bother?

Instead of considering a single answer to a question, Bayesian methods allow us to consider an entire distribution of answers. With this approach we can naturally address issues like:

- regularization (overfitting or not),
- model selection / comparison,

without the need for a separate cross-validation data set.

With these techniques we can also put error bars on the output of the network, by considering the shape of the output distribution $p(\mathbf{y}|\mathcal{D})$.

Overview

We will be looking at how, using Bayesian methods, we can explore the follow questions:

1. $p(\mathbf{w}|\mathcal{D}, \mathcal{H})$? What is the distribution over weights $\mathbf{w}$ given the data and a fixed model, $\mathcal{H}$?
2. $p(\mathbf{y}|\mathcal{D}, \mathcal{H})$? What is the distribution over network outputs $\mathbf{y}$ given the data and a model (for regression problems)?
3. $p(\mathcal{C}|\mathcal{D}, \mathcal{H})$? What is the distribution over predicted class labels $\mathcal{C}$ given the data and model (for classification problems)?
4. $p(\mathcal{H}|\mathcal{D})$? What is the distribution over models given the data?
5. $p(\mathbf{y}|\mathcal{D})$? What is the distribution over network outputs given the data (not conditioned on a particular model!)?

Overview (cont.)

We will also look briefly Monte Carlo sampling methods to deal with using Bayesian methods in the “real world”.

A good deal of current research is going into applying such methods to deal with Bayesian inference in difficult problems.

Maximum Likelihood Learning

Optimization methods focus on finding a single weight assignment that minimize some error function (typically a least squared error function).

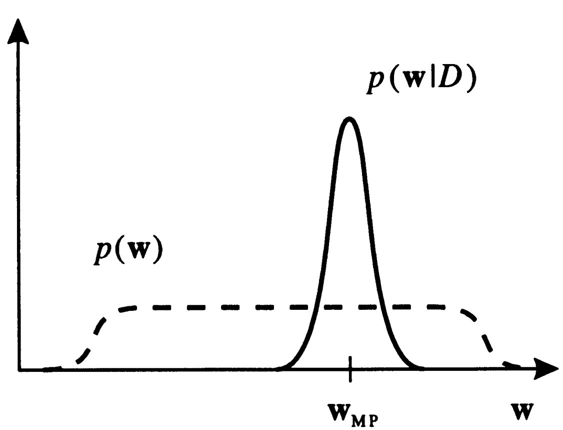
This is equivalent to finding a maximum of the likelihood function, i.e. finding a $\mathbf{w}^*$ that maximizes the probability of the data given those weights, $p(\mathcal{D}|\mathbf{w}^*)$.

1. Bayesian learning of the weights

Here we consider finding a posterior distribution over weights,

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{\int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})\, d\mathbf{w}}.$$

In the Bayesian formalism, learning the weights means changing our belief about the weights from the prior, $p(\mathbf{w})$, to the posterior, $p(\mathbf{w}|\mathcal{D})$ as a consequence of seeing the data.



Prior for the weights

Let's consider a prior for the weights of the form

$$p(\mathbf{w}) = \frac{\exp(-\alpha E_{\mathbf{w}})}{Z_{\mathbf{w}}(\alpha)}$$

where α is a hyperparameter (a parameter of a prior distribution over another parameter, for now we will assume α is known) and normalizer $Z_{\mathbf{w}}(\alpha) = \int \exp(-\alpha E_{\mathbf{w}})\, d\mathbf{w}$.

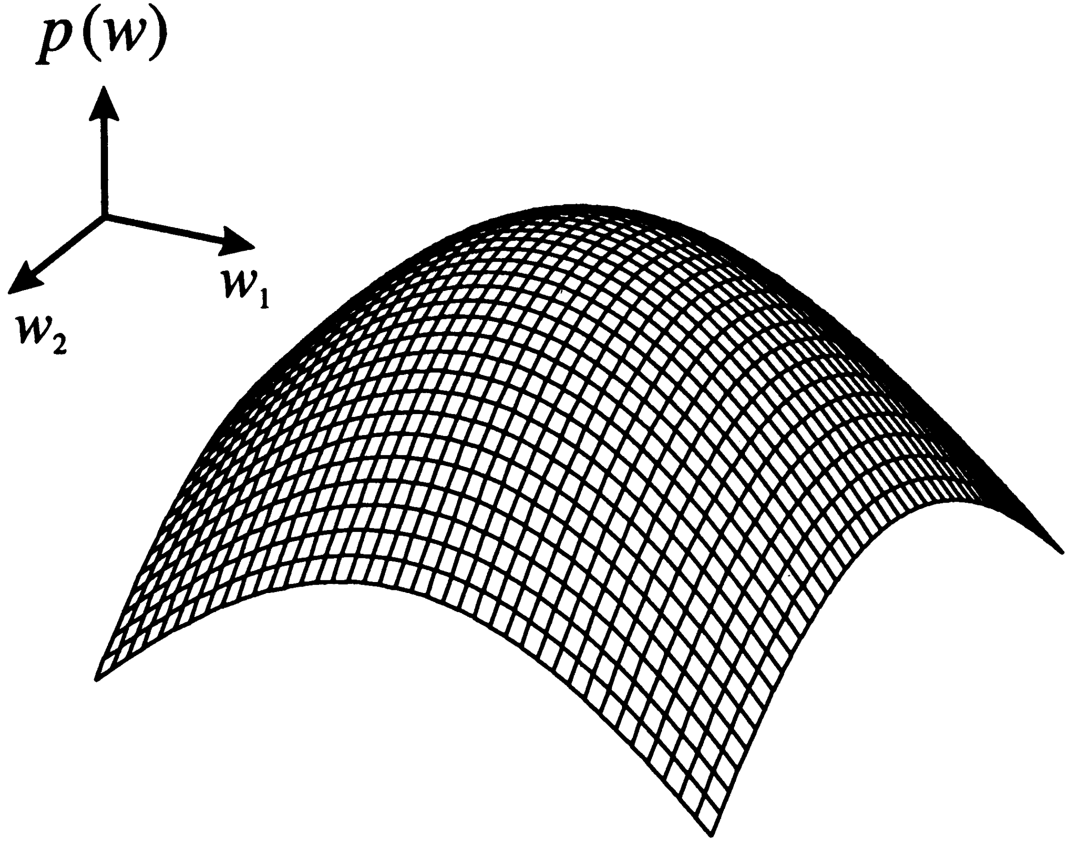
When we considered weight decay we argued that smaller weights generalize better, so we should set $E_{\mathbf{w}}$ to

$$E_{\mathbf{w}} = \frac{1}{2}||w||^2 = \frac{1}{2}\sum_{i=1}^{W} w_i^2.$$

With this $E_{\mathbf{w}}$, the prior becomes a Gaussian.

Example prior

A prior over two weights.



Likelihood of the data

Just as we did for the prior, let's consider a likelihood function of the form

$$p(\mathcal{D}|\mathbf{w}) = \frac{\exp(-\beta E_{\mathcal{D}})}{Z_{\mathcal{D}}(\beta)}$$

where β is another hyperparameter and the normalization factor $Z_{\mathcal{D}}(\beta) = \int \exp(-\beta E_{\mathcal{D}})\, d\mathcal{D}$ (where $\int d\mathcal{D} = \int dt^1 \ldots dt^N$)

If we assume that after training the target data $t \in \mathcal{D}$ obeys a Gaussian distribution with mean $y(\mathbf{x};\mathbf{w})$, then the likelihood function is given by

$$p(\mathcal{D}|\mathbf{w}) = \prod_{n=1}^{N} p(t^n|\mathbf{x}^n, \mathbf{w}) = \frac{1}{Z_{\mathcal{D}}(\beta)} \exp(-\frac{\beta}{2} \sum_{n=1}^{N} \{y(\mathbf{x};\mathbf{w}) - t^n\}^2)$$

Posterior over the weights

With $p(\mathbf{w})$ and $p(\mathcal{D}|\mathbf{w})$ defined, we can now combine them according to Bayes rule to get the posterior distribution,

$$
\begin{aligned}
p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{P(\mathcal{D})} &= \frac{1}{Z_S}\exp(-\beta E_{\mathcal{D}})\exp(-\alpha E_{\mathbf{w}}) \\
&= \frac{1}{Z_S}\exp(-S(\mathbf{w}))
\end{aligned}
$$

where

$$S(\mathbf{w}) = \beta E_{\mathcal{D}} + \alpha E_{\mathbf{w}}$$

and

$$Z_S(\alpha, \beta) = \int \exp(-\beta E_{\mathcal{D}} - \alpha E_{\mathbf{w}})\, d\mathbf{w}$$

Posterior over the weights (cont.)

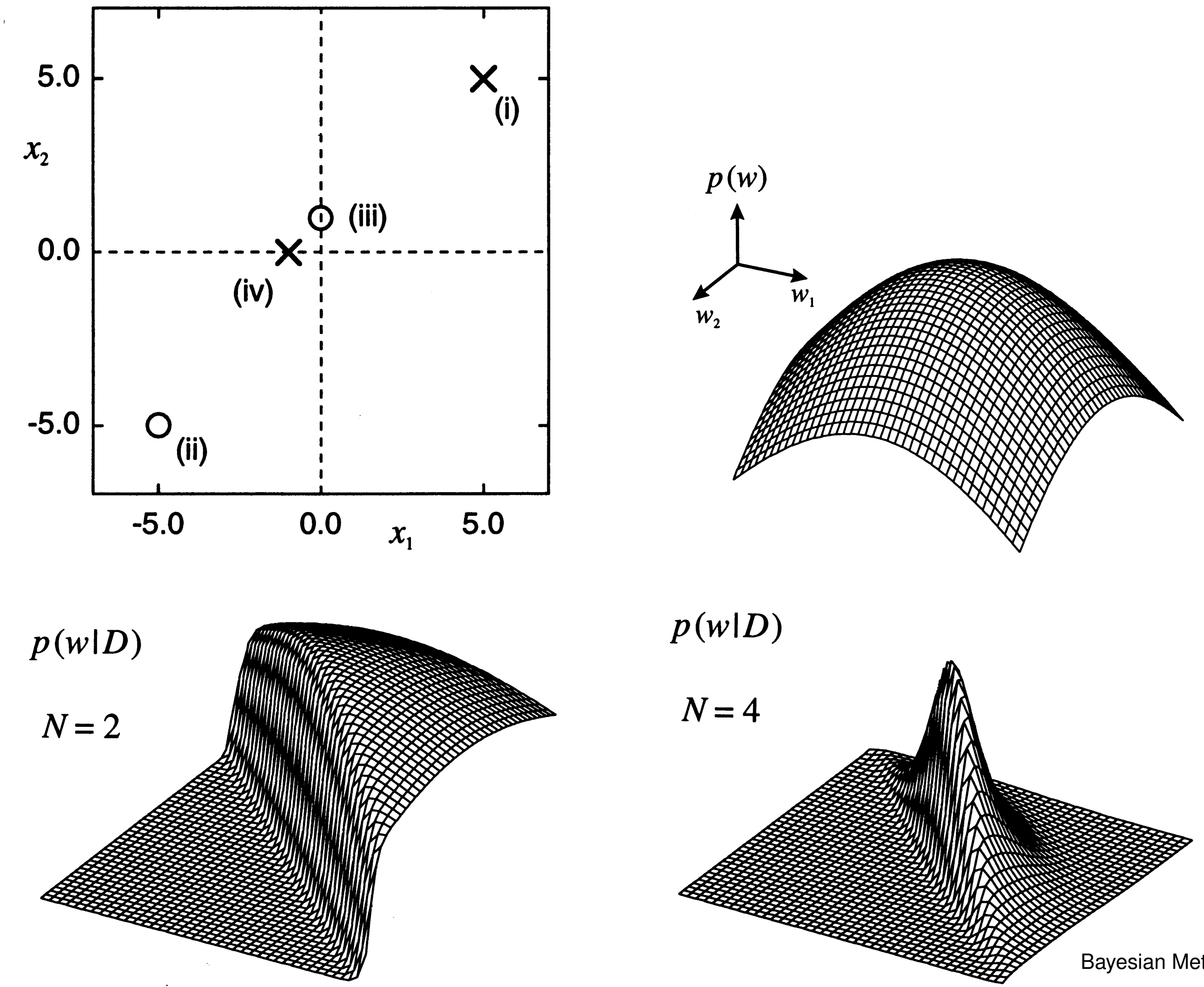
If we imagine we want to find the maximum *a posteriori* weights, $\mathbf{w}_{MP}$ (the maximum the posterior distribution), we could minimize the negative logarithm of $p(\mathbf{w}|\mathcal{D})$, which is equivalent to minimizing

$$S(\mathbf{w}) = \frac{\beta}{2} \sum_{n=1}^{N} \{y(\mathbf{x};\mathbf{w}) - t^n\}^2 + \frac{\alpha}{2} \sum_{i=1}^{W} w_i^2.$$

We've seen this before, it's the error function minimized with weight decay! The ratio α/β determines the amount we penalize large weights.

Example of Bayesian Learning

A classification problem with two inputs and one logistic output.



2. Finding a distribution over outputs

Once we have the posterior of the weights, we can consider the output of the whole distribution of weight values to produce a distribution over the network outputs.

$$p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) \, d\mathbf{w}$$

where we are *marginalizing* over the weights. In general, we require an approximation to evaluate this integral.

Distribution over outputs (cont.)

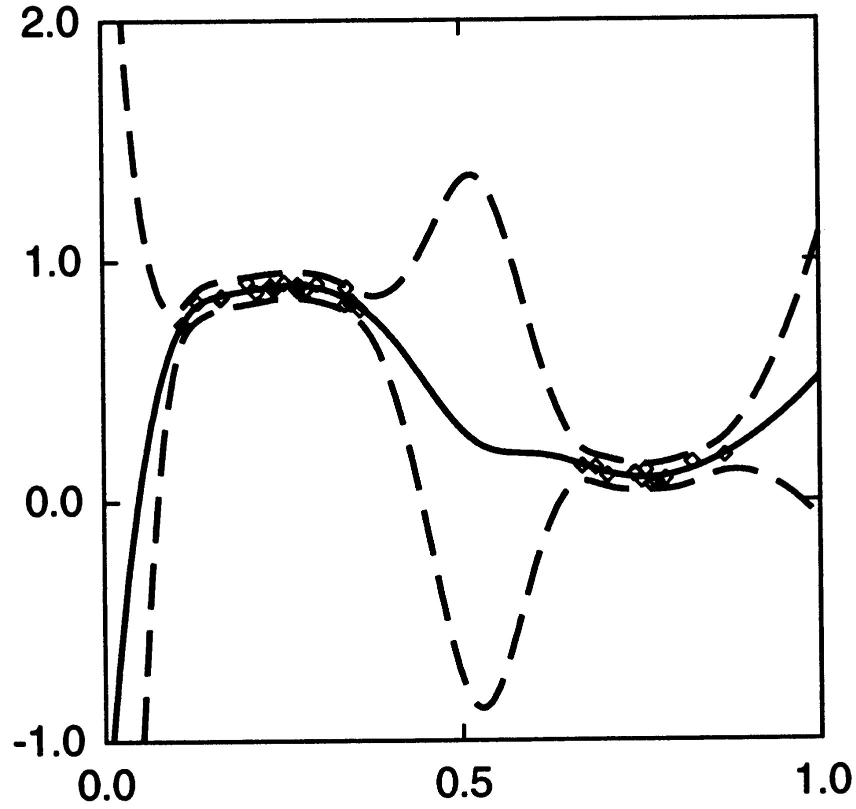
If we approximate $p(\mathbf{w}|\mathcal{D})$ as a sufficiently narrow Gaussian, we arrive at a gaussian distribution over the outputs of the network,

$$p(y|\mathbf{x}, \mathcal{D}) \approx \frac{1}{2\pi\sigma_y^{1/2}} \exp(-\frac{(y - y_{MP})^2}{2\sigma_y^2}),$$

The mean y_{MP} is the maximum *a posteriori* network output and the variance $\sigma_y^2 = \beta^{-1} + \mathbf{g}^T\mathbf{A}^{-1}\mathbf{g}$, where $\mathbf{A}$ is the Hessian of $S(\mathbf{w})$ and $\mathbf{g} \equiv \nabla_{\mathbf{w}} y|_{\mathbf{w}_{MP}}$.

Example of Bayesian Regression

The figure is an example of the application of Bayesian methods to a regression problem. The data (circles) was generated from the function, $h(x) = 0.5 + 0.4\sin(2\pi x)$.



3. Bayesian Classification with ANNs

We can apply the same techniques to classification problems where, for the two classes, the likelihood function is given by,

$$p(\mathcal{D}|\mathbf{w}) = \prod_n y(\mathbf{x}^n)^{t^n} (1 - y(\mathbf{x}^n))^{1-t^n} = \exp(-G(\mathcal{D}|\mathbf{w}))$$

where $G(\mathcal{D}|\mathbf{w})$ is the cross-entropy error function

$$G(\mathcal{D}|\mathbf{w}) = -\sum_n \{t^n \ln y(\mathbf{x^n}) + (1 - t^n) \ln(1 - y(\mathbf{x^n}))\}$$

Classification (cont.)

If we use a logistic sigmoid $y(\mathbf{x}; \mathbf{w})$ as the output activation function and interpret that as $P(\mathcal{C}_1|\mathbf{x}, \mathbf{w}))$, then the output distribution is given by

$$P(\mathcal{C}_1|\mathbf{x}, \mathcal{D}) = \int y(\mathbf{x}; \mathbf{w}) p(\mathbf{w}|\mathcal{D})\, d\mathbf{w}$$

Once again we have *marginalized* out the weights.
As we did in the case of regression, we could now apply approximations to evaluate this integral (details in the reading).

Example of Bayesian Classification

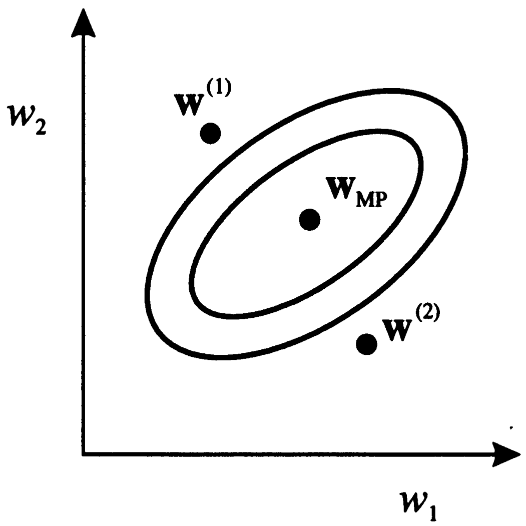


Figure 1

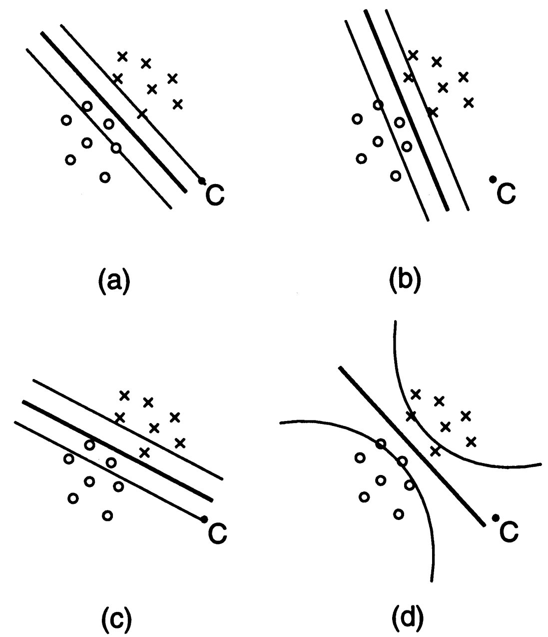


Figure 2

The three lines in Figure 2 correspond to network outputs of 0.1, 0.5, and 0.9. (a) shows the predictions made by $\mathbf{w}_{MP}$. (b) and (c) show the predictions made by the weights $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$. (d) shows $P(\mathcal{C}_1|\mathbf{x}, \mathcal{D})$, the prediction after marginalizing over the distribution of weights.

What about α and β?

Until now, we have assumed that the hyperparameters are known a priori, but in practice we will almost never know the correct form of the prior. There exist two possible alternative solutions to this problem:

1. We could find their maximum *a posteriori* values in an iterative optimization procedure where we alternate between optimizing $\mathbf{w}_{MP}$ and the hyperparameters α_{MP} and β_{MP}
2. We could be proper Bayesians and marginalize (or integrate) over the hyperparameters. For example

$$p(\mathbf{w}|\mathcal{D}) = \frac{1}{p(\mathcal{D})} \int\int p(\mathcal{D}|\mathbf{w}, \beta) p(\mathbf{w}|\alpha) p(\alpha) p(\beta) \, d\alpha \, d\beta.$$

4. Bayesian Model Comparison

Until now, we have been dealing with the application of Bayesian methods to a neural network with a fixed number of units and a fixed architecture.

With Bayesian methods, we can generalize learning to include learning the appropriate model size and even model type.

Consider a set of candidate models $\mathcal{H}_i$ that could include neural networks with different numbers of hidden units, RBF networks and other models.

Model Comparison (cont.)

We can apply Bayes' theorem to compute the posterior distribution over models, then pick the model with the largest posterior.

$$P(\mathcal{H}_i|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{H}_i)P(\mathcal{H}_i)}{p(\mathcal{D})}$$

The term $p(\mathcal{D}|\mathcal{H}_i)$ is called the evidence for $\mathcal{H}_i$ and is given by

$$p(\mathcal{D}|\mathcal{H}_i) = \int p(\mathcal{D}|\mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\mathcal{H}_i)\, d\mathbf{w}.$$

The evidence term balances between fitting the data well and avoiding overly complex models.

Model evidence $p(\mathcal{D}|\mathcal{H}_i)$

Consider a single weight, w. If we assume that the posterior is sharply peaked around the most probable value, w_{MP}, with width $\Delta w_{posterior}$ we can approximate the integral with the expression

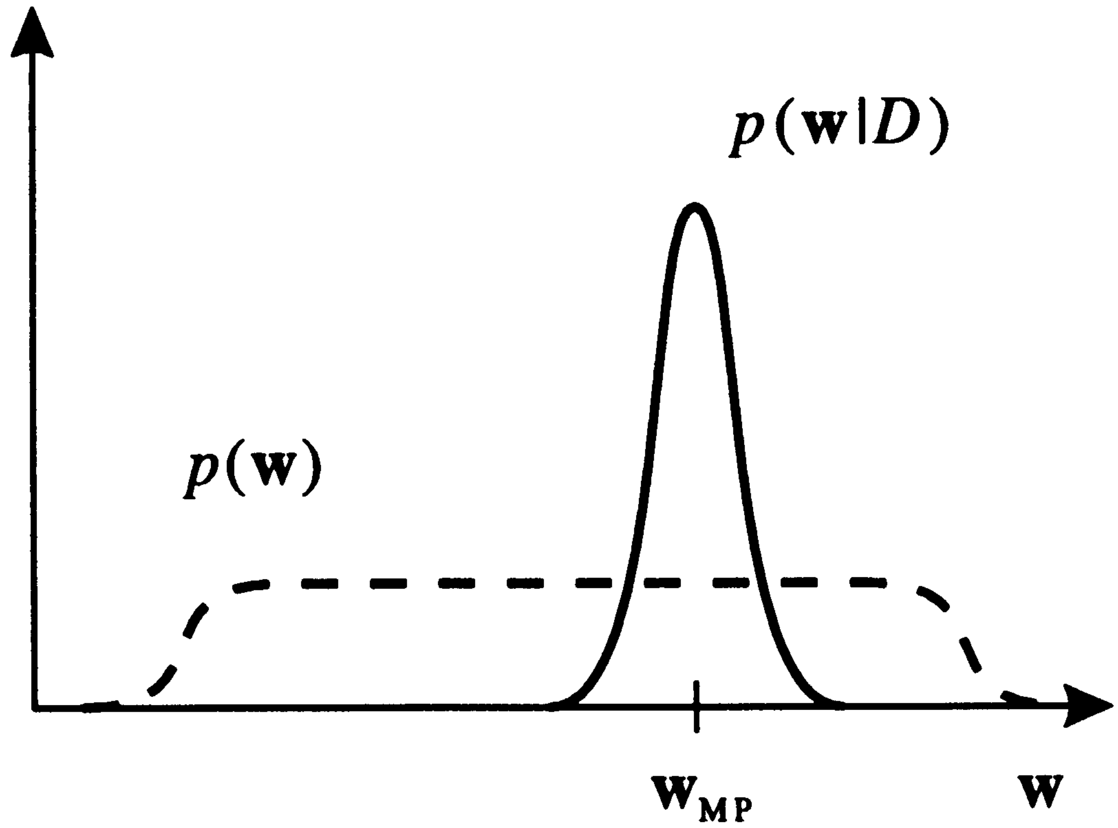
$$p(\mathcal{D}|\mathcal{H}_i) \approx p(\mathcal{D}|w_{MP}, \mathcal{H}_i) p(w_{MP}|\mathcal{H}_i)\, \Delta w_{posterior}.$$

If we also take the prior over the the weights to be uniform over a large interval Δw_{prior} then the approximation to the evidence becomes

$$p(\mathcal{D}|\mathcal{H}_i) \approx p(\mathcal{D}|w_{MP}, \mathcal{H}_i)(\frac{\Delta w_{posterior}}{\Delta w_{prior}}).$$

The ratio $\Delta w_{posterior}/\Delta w_{prior}$ is called the Occam factor and penalizes complex models.

Illustration of the Occam factor



5. Committee of models

We can go even further with Bayesian methods. Rather than picking a single model we can marginalize over a number of different models.

$$p(y|\mathbf{x}, \mathcal{D}) = \sum_i p(y|\mathbf{x}, \mathcal{H}_i) P(\mathcal{H}_i|\mathcal{D})$$

The result is a weighted average of the probability distributions over the outputs of the models in the committee.

Bayesian Methods in Practice

Bayesian methods are almost always difficult to apply directly. They involve integrals that are intractable except in the most trivial cases.

Until now, we have made assumptions about the shape of the distributions in the integrations. For a wide array of problems these assumption do not hold and may lead to very poor performance.

Typical numerical integration techniques are unsuitable for the integrations involved in applying Bayesian methods, where the integrals are over a large number of dimensions.

Monte Carlo techniques offer a way around this problem.

Monte Carlo Sampling Methods

We wish to evaluate integrals of the form:

$$I = \int F(\mathbf{w})p(\mathbf{w}|\mathcal{D})\,d\mathbf{w}$$

The idea is to approximate the integral with a finite sum,

$$I \approx \frac{1}{L}\sum_{i=L}^{L} F(\mathbf{w}_i)$$

where $\mathbf{w}_i$ is a sample of the weights generated from the distribution $p(\mathbf{w}|\mathcal{D})$. The challenge in Monte Carlo method is that it is often difficult to sample from $p(\mathbf{w}|\mathcal{D})$ directly.

Importance Sampling

If sampling from the distribution $p(\mathbf{w}|\mathcal{D})$ is impractical, we could sample from a simpler distribution $q(\mathbf{w})$, from which it is easy to sample. Then we can write

$$I = \int F(\mathbf{w}) \frac{p(\mathbf{w}|\mathcal{D})}{q(\mathbf{w})} q(\mathbf{w})\, d\mathbf{w} \approx \frac{1}{L} \sum_{i=1}^{L} F(\mathbf{w}_i) \frac{p(\mathbf{w}_i|\mathcal{D})}{q(\mathbf{w}_i)}$$

In general we cannot normalize $p(\mathbf{w}|\mathcal{D})$ so we use a modified form of the approximation with an unnormalized $\tilde{p}(\mathbf{w}_i|\mathcal{D})$,

$$I \approx \frac{\sum_{i=1}^{L} F(\mathbf{w}_i) \tilde{p}(\mathbf{w}_i|\mathcal{D}) / q(\mathbf{w}_i)}{\sum_{i=1}^{L} \tilde{p}(\mathbf{w}_i|\mathcal{D}) / q(\mathbf{w}_i)}$$