16 *Bayesian Estimation*

In the Bayesian approach, we consider parameters as random variables with a distribution allowing us to model our uncertainty in estimating them. We continue from where we left off in section 4.4 and discuss estimating both the parameters of a distribution and the parameters of a model for regression, classification, clustering, or dimensionality reduction. We also discuss nonparametric Bayesian modeling where model complexity is not fixed but depends on the data.

16.1 Introduction

BAYESIAN ESTIMATION, which we introduced in section 4.4, treats a parameter θ as a random variable with a probability distribution. The maximum likelihood approach we discussed in section 4.2 treats a parameter as an unknown constant. For example, if the parameter we want to estimate is the mean μ, its maximum likelihood estimator is the sample average $\overline{X}$. We calculate $\overline{X}$ over our training set, plug it in our model, and use it, for example, for classification. However, we know that especially with small samples, the maximum likelihood estimator can be a poor estimator and has variance—as the training set varies, we may calculate different values of $\overline{X}$, which in turn may lead to different discriminants with different generalization accuracies.

In Bayesian estimation, we make use of the fact that we have uncertainty in estimating θ and instead of a single θ_{ML}, we use all θ weighted by our estimated distribution, $p(\theta|\mathcal{X})$. That is, we average over our uncertainty in estimating θ.

While estimating $p(\theta|\mathcal{X})$, we can make use of the prior information we

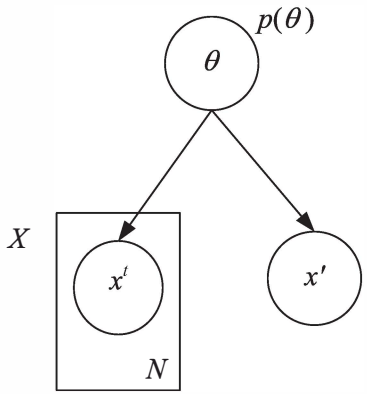

Figure 16.1 The generative graphical model (see chapter 14). The arcs are in the direction of sampling; first we pick θ from $p(\theta)$, and then we generate data by sampling from $p(x|\theta)$. The rectangular *plate* contains N independent instances drawn, and they make up the training set $\mathcal{X}$. The new x' is independently drawn given θ. This is the iid assumption. If θ is not known, they are dependent. We infer θ from the past instances using Bayes' rule, which is then used to make inference about the new x'.

may have regarding the value of the parameter. Such prior beliefs are especially important when we have a small sample (and when the variance of the maximum likelihood estimator is high). In such a case, we are interested in combining what the data tells us, namely, the value calculated from the sample, and our prior information. As we first discussed in section 4.4, we code this information using a *prior probability* distribution. For example, before looking at a sample to estimate the mean, we may have some prior belief that it is close to 2, between 1 and 3, and in such a case, we write $p(\mu)$ in such a way that the bulk of the density lies in the interval $[1, 3]$.

PRIOR PROBABILITY

Using Bayes' rule, we combine the prior and the likelihood and calculate the *posterior probability* distribution:

POSTERIOR PROBABILITY

$$p(\theta|\mathcal{X}) = \frac{p(\theta)p(\mathcal{X}|\theta)}{p(\mathcal{X})} \tag{16.1}$$

Here, $p(\theta)$ is the prior density; it is what we know regarding the possible values that θ may take *before* looking at the sample. $p(\mathcal{X}|\theta)$ is the *sample likelihood*; it tells us how likely our sample $\mathcal{X}$ is if the parameter of the distribution takes the value θ. For example, if the instances in our

sample are between 5 and 10, such a sample is likely if μ is 7 but is less likely if μ is 3 and even less likely if μ is 1. $p(\mathcal{X})$ in the denominator is a normalizer to make sure that the *posterior* $p(\theta|\mathcal{X})$ integrates to 1. It is called the posterior probability because it tells us how likely θ takes a certain value *after* looking at the sample. The Bayes' rule takes the prior distribution, combines it with what the data reveals, and generates the posterior distribution. We then use this posterior distribution later for making inference.

Let us say that we have a past sample $\mathcal{X} = \{\boldsymbol{x}^t\}_{t=1}^N$ drawn from sone distribution with unknown parameter θ. We can then draw one more instance x', and we would like to calculate its probability distribution. We can visualize this as a graphical model (chapter 14) as shown in figure 16.1. What is shown here is a *generative model* representing how the data is generated. We first sample θ from $p(\theta)$ and then sample from $p(x|\theta)$ to first generate the training instances $\boldsymbol{x}^t$ and also the new test x'.

GENERATIVE MODEL

We write the joint as

$$p(x', \mathcal{X}, \theta) = p(\theta)p(\mathcal{X}|\theta)p(x'|\theta)$$

We can estimate the probability distribution for the new x given the sample $\mathcal{X}$:

$$\begin{aligned} p(x'|\mathcal{X}) &= \frac{p(x', \mathcal{X})}{p(\mathcal{X})} = \frac{\int p(x', \mathcal{X}, \theta)d\theta}{p(\mathcal{X})} = \frac{\int p(\theta)p(\mathcal{X}|\theta)p(x'|\theta)d\theta}{p(\mathcal{X})} \\ &= \int p(x'|\theta)p(\theta|\mathcal{X})d\theta \end{aligned} \tag{16.2}$$

In calculating $p(\theta|\mathcal{X})$, Bayes' rule inverts the direction of the arc and makes a diagnostic inference. This inferred (posterior) distribution is then used to derive a predictive distribution for new x.

We see that our estimate is a weighted sum (we replace $\int d\theta$ by $\sum_\theta$ if θ is discrete valued) of estimates using all possible values of θ weighted by how likely each θ is, given the sample $\mathcal{X}$.

This is the *full Bayesian treatment* and it may not be possible if the posterior is not easy to integrate. As we saw in section 4.4, in the case of the *maximum a posteriori (MAP) estimate*, we use the mode of the posterior:

MAXIMUM A POSTERIORI (MAP) ESTIMATE

$$\theta_{MAP} = \arg\max_\theta p(\theta|\mathcal{X}) \text{ and } p_{MAP}(x'|\mathcal{X}) = p(x'|\theta_{MAP})$$

The MAP estimate corresponds to assuming that the posterior makes a very narrow peak around a single point, that is, the mode. If the prior

$p(\theta)$ is uniform over all θ, then the mode of the posterior $p(\theta|\mathcal{X})$ and the mode of the likelihood $p(\mathcal{X}|\theta)$ are at the same point, and the MAP estimate is equal to the maximum likelihood (ML) estimate:

$$\theta_{ML} = \arg\max_{\theta} p(\mathcal{X}|\theta) \text{ and } p_{ML}(x'|\mathcal{X}) = p(x'|\theta_{ML})$$

This implies that using ML corresponds to assuming no a priori distinction between different values of θ.

Basically, the Bayesian approach has two advantages:

1. The prior helps us ignore the values that θ is unlikely to take and concentrate on the region where it is likely to lie. Even a weak prior with long tails can be very helpful.

2. Instead of using a single θ estimate in prediction, we generate a set of possible θ values (as defined by the posterior) and use all of them in prediction, weighted by how likely they are.

If we use the MAP estimate instead of integrating over θ, we make use of the first advantage but not the second—if we use the ML estimate, we lose both advantages. If we use an uninformative (uniform) prior, we make use of the second advantage but not the first. Actually it is this second advantage, rather than the first, that makes the Bayesian approach interesting, and in chapter 17, we discuss combining multiple models where we see methods that are very similar, though not always Bayesian.

This approach can be used in different types of distributions and for different types of applications. The parameter θ can be the parameter of a distribution. For example, in classification, it can be the unknown class mean, for which we define a prior and get its posterior; then, we get a different discriminant for each possible value of the mean and hence the Bayesian approach will average over all possible discriminants whereas in the ML approach there is a single mean estimate and hence a single discriminant.

The unknown parameter, as we will see shortly, can also be the parameters of a fitted model. For example, in linear regression, we can define a prior distribution on the slope and the intercept parameters and calculate a posterior on them, that is, a distribution over lines. We will then be averaging over the prediction of all possible lines, weighted by how likely they are as specified by their prior weights and how well they fit the given data.

One of the most critical aspects of Bayesian estimation is evaluating the integral in equation 16.2. For some cases, we can calculate it, but mostly we cannot, and in such cases, we need to approximate it, and we will see methods for this in the next few sections, namely, Laplace and variational approximations, and Markov chain Monte Carlo (MCMC) sampling.

Now, let us see these and other applications of the Bayesian approach in more detail, starting from simple and incrementally making them more complex.

16.2 Bayesian Estimation of the Parameters of a Discrete Distribution

16.2.1 $K > 2$ States: Dirichlet Distribution

Let us say that each instance is a multinomial variable taking one of K distinct states (section 4.2.2). We say $x_i^t = 1$ if instance t is in state i and $x_j^t = 0, \forall j \neq i$. The parameters are the probabilities of states, $\boldsymbol{q} = [q_1, q_2, \ldots, q_K]^T$ with $q_i, i = 1, \ldots, K$ satisfying $q_i \geq 0, \forall i$ and $\sum_i q_i = 1$.

For example, x^t may correspond to news documents and states may correspond to K different news categories: sports, politics, arts, and so on. The probabilities q_i then correspond to the proportions of different news categories, and priors on them allow us to code our prior beliefs in these proportions; for example, we may expect to have more news related to sports than news related to arts.

The sample likelihood is

$$p(\mathcal{X}|\boldsymbol{q}) = \prod_{t=1}^{N} \prod_{i=1}^{K} q_i^{x_i^t}$$

DIRICHLET DISTRIBUTION

The prior distribution of $\boldsymbol{q}$ is the *Dirichlet distribution*:

$$\text{Dirichlet}(\boldsymbol{q}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)} \prod_{i=1}^{K} q_i^{\alpha_i - 1}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_K]^T$ and $\alpha_0 = \sum_i \alpha_i$. α_i, the parameters of the prior, are called the *hyperparameters*. $\Gamma(x)$ is the *gamma function* defined as

GAMMA FUNCTION

$$\Gamma(x) \equiv \int_0^\infty u^{x-1} e^{-u} du$$

Given the prior and the likelihood, we can derive the posterior:

$$p(\boldsymbol{q}|\mathcal{X}) \quad \propto \quad p(\mathcal{X}|\boldsymbol{q}) p(\boldsymbol{q}|\boldsymbol{\alpha})$$

$$\propto \prod_i q_i^{\alpha_i + N_i - 1} \tag{16.3}$$

where $N_i = \sum_{t=1}^{N} x_i^t$. We see that the posterior has the same form as the prior, and we call such a prior a *conjugate prior*. Both the prior and the likelihood have the form of product of powers of q_i, and we combine them to make up the posterior:

CONJUGATE PRIOR

$$\begin{aligned} p(\boldsymbol{q}|\mathcal{X}) &= \frac{\Gamma(\alpha_0 + N)}{\Gamma(\alpha_1 + N_1) \cdots \Gamma(\alpha_K + N_K)} \prod_{i=1}^{K} q_i^{\alpha_i + N_i - 1} \\ &= \text{Dirichlet}(\boldsymbol{q}|\boldsymbol{\alpha} + \boldsymbol{n}) \end{aligned} \tag{16.4}$$

where $\boldsymbol{n} = [N_1, \ldots, N_K]^T$ and $\sum_i N_i = N$.

Looking at equation 16.3, we can bring an interpretation to the hyperparameters α_i (Bishop 2006). Just as n_i are counts of occurrences of state i in a sample of N, we can view α_i as counts of occurences of state i in some imaginary sample of α_0 instances. In defining the prior, we are subjectively saying the following: In a sample of α_0, I would expect α_i of them to belong to state i. Note that larger α_0 implies that we have a higher confidence (a more peaked distribution) in our subjective proportions: Saying that I expect to have 60 out of 100 occurrences belong to state 1 has higher confidence than saying that I expect to have 6 out of 10. The posterior then is another Dirichlet that sums up the counts of the occurences of states, imagined and actual, given by the prior and the likelihood, respectively.

The conjugacy has a nice implication. In a sequential setting where we receive a sequence of instances, because the posterior and the prior have the same form, the current posterior accumulates information from all past instances and becomes the prior for the next instance.

16.2.2 $K = 2$ States: Beta Distribution

When the variable is binary, $x^t \in \{0, 1\}$, the multinomial sample becomes Bernoulli:

$$p(\mathcal{X}|q) = \prod_t q^{x^t} (1 - q)^{1 - x^t}$$

BETA DISTRIBUTION

and the Dirichlet prior reduces to the *beta distribution*:

$$\text{beta}(q|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} q^{\alpha - 1} (1 - q)^{\beta - 1}$$

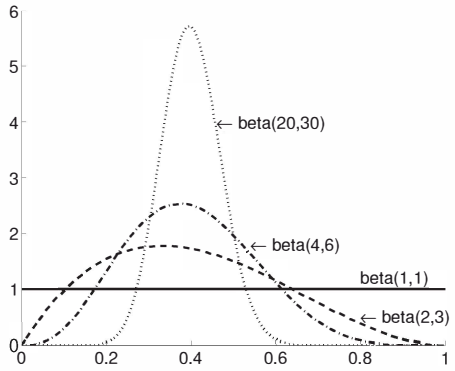

Figure 16.2 Plots of beta distributions for different sets of (α, β).

For example, x^t may be 0 or 1 depending on whether email with index t in a random sample of size N is legitimate or spam, respectively. Then defining a prior on q allows us to define a prior belief on the spam probability: I would expect, on the average, $\alpha/(\alpha+\beta)$ of my emails to be spam.

Beta is a conjugate prior, and for the posterior we get

$$p(q|A, N, \alpha, \beta) \propto q^{A+\alpha-1}(1-p)^{N-A+\beta-1}$$

where $A = \sum_t x^t$, and we see again that we combine the occurrences in the imaginary and the actual samples. Note that when $\alpha = \beta = 1$, we have a uniform prior and the posterior has the same shape as the likelihood. As the two counts, whether α and β for the prior or $\alpha+A$ and $\beta+N-A$ for the posterior, increase and their difference increases, we get a distribution that is more peaked with smaller variance (see figure 16.2). As we see more data (imagined or actual), the variance decreases.

16.3 Bayesian Estimation of the Parameters of a Gaussian Distribution

16.3.1 Univariate Case: Unknown Mean, Known Variance

We now consider the case where instances are Gaussian distributed. Let us start with the univariate case, $p(x) \sim \mathcal{N}(\mu, \sigma^2)$, where the parame-

ters are μ and σ^2; we discussed this briefly in section 4.4. The sample likelihood is

$$p(\mathcal{X}|\mu,\sigma^2) = \prod_t \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x^t-\mu)^2}{2\sigma^2}\right] \tag{16.5}$$

The conjugate prior for μ is Gaussian, $p(\mu) \sim \mathcal{N}(\mu_0^2, \sigma_0^2)$, and we write the posterior as

$$\begin{aligned} p(\mu|\mathcal{X}) &\propto p(\mu)p(\mathcal{X}|\mu) \\ &\sim \mathcal{N}(\mu_N, \sigma_N^2) \end{aligned}$$

where

$$\mu_N = \frac{\sigma^2}{N\sigma_0^2+\sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2+\sigma^2}m \tag{16.6}$$

$$\frac{1}{\sigma_N^2} = \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2} \tag{16.7}$$

where $m = \sum_t x^t/N$ is the sample average. We see that the mean of the posterior density (which is the MAP estimate), μ_N, is a weighted average of the prior mean μ_0 and the sample mean m, with weights being inversely proportional to their variances (see figure 16.3 for an example). Note that because both coefficients are between 0 and 1 and sum to 1, μ_N is always between μ_0 and m. When the sample size N or the variance of the prior σ_0^2 is large, the posterior mean is close to m, relying more on the information provided by the sample. When σ_0^2 is small—that is, when we have little prior uncertainty regarding the correct value of μ, or when we have a small sample—our prior guess μ_0 has higher effect.

σ_N gets smaller when either of σ_0 or σ gets smaller or if N is larger. Note also that σ_N is smaller than both σ_0 and $\sigma/\sqrt{N}$, that is, the posterior variance is smaller than both prior variance and that of m. Incorporating both results in a better posterior estimate than using any of the prior or sample alone.

If σ^2 is known, for new x, we can integrate over this posterior to make a prediction:

$$\begin{aligned} p(x|\mathcal{X}) &= \int p(x|\mu)p(\mu|\mathcal{X})d\mu \\ &\sim \mathcal{N}(\mu_N, \sigma_N^2+\sigma^2) \end{aligned} \tag{16.8}$$

We see that x is still Gaussian, that it is centered at the posterior mean, and that its variance now includes the uncertainty due to the estimation

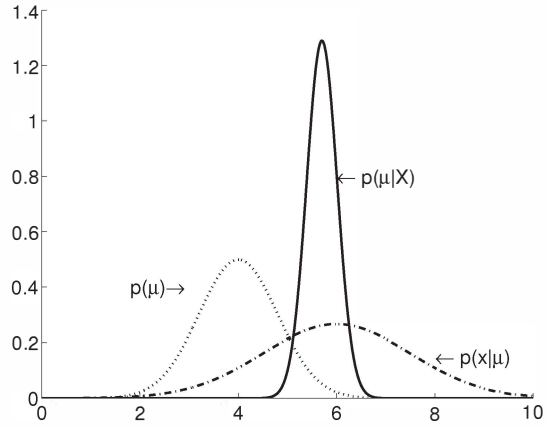

Figure 16.3 Twenty data points are drawn from $p(x) \sim \mathcal{N}(6, 1.5^2)$, prior is $p(\mu) \sim \mathcal{N}(4, 0.8^2)$, and posterior is then $p(\mu|\mathcal{X}) \sim \mathcal{N}(5.7, 0.3^2)$.

of the mean and the new sampled instance x. We can write $x = \mu + x'$ where $x' \sim \mathcal{N}(0, \sigma^2)$; then $E[x] = E[\mu] + E[x'] = \mu_N$ and $\text{Var}(x) = \text{Var}(\mu) + \text{Var}(x') = \sigma_N^2 + \sigma^2$, where this last follows from the fact that the new x' is an independent draw.

Once we get a distribution for $p(x|\mathcal{X})$, we can use it for different purposes. For example in classification, this approach corresponds to assuming Gaussian classes where means have a Gaussian prior and they are trained using $\mathcal{X}_i$, the subset of $\mathcal{X}$ labeled by class C_i. Then, $p(x|\mathcal{X}_i)$ as calculated above, corresponds to $p(x|C_i)$, which we combine with prior $P(C_i)$ to get the posterior and hence a discriminant.

16.3.2 Univariate Case: Unknown Mean, Unknown Variance

If we do not know σ^2, we also need to estimate it. For the case of variance, we work with the *precision*, the reciprocal of the variance, $\lambda \equiv 1/\sigma^2$. Using this, the sample likelihood is written as

PRECISION

$$\begin{aligned} p(\mathcal{X}|\lambda) &= \prod_t \frac{\lambda^{1/2}}{\sqrt{2\pi}} \exp\left[-\frac{\lambda}{2}(x^t - \mu)^2\right] \\ &= \lambda^{N/2}(2\pi)^{-N/2} \exp\left[-\frac{\lambda}{2}\sum_t (x^t - \mu)^2\right] \end{aligned} \tag{16.9}$$

GAMMA DISTRIBUTION

The conjugate prior for the precision is the *gamma distribution*:

$$p(\lambda) \sim \text{gamma}(a_0, b_0) = \frac{1}{\Gamma(a_0)} b_0^{a_0} \lambda^{a_0 - 1} \exp(-b_0 \lambda)$$

where we define $a_0 \equiv \nu_0/2$ and $b_0 \equiv (\nu_0/2)s_0^2$ such that s_0^2 is our prior estimate of variance and ν_0 is our confidence in this prior—it may be thought of as the size of the imaginary sample on which we believe s_0^2 is estimated.

The posterior then is also gamma:

$$\begin{aligned} p(\lambda|\mathcal{X}) &\propto p(\mathcal{X}|\lambda)p(\lambda) \\ &\sim \text{gamma}(a_N, b_N) \end{aligned}$$

where

$$\begin{aligned} a_N &= a_0 + N/2 = \frac{\nu_0 + N}{2} \\ b_N &= b_0 + \frac{N}{2}s^2 = \frac{\nu_0}{2}s_0^2 + \frac{N}{2}s^2 \end{aligned} \tag{16.10}$$

where $s^2 = \sum_t (x^t - \mu)^2/N$ is the sample variance. Again, we see that posterior estimates are weighted sum of priors and sample statistics.

To make a prediction for new x, when both μ and σ^2 are unknown, we need the joint posterior that we write as

$$p(\mu, \lambda) = p(\mu|\lambda)p(\lambda)$$

where $p(\lambda) \sim \text{gamma}(a_0, b_0)$ and $p(\mu|\lambda) \sim \mathcal{N}(\mu_0, 1/(\kappa_0\lambda))$. Here again, κ_0 may be thought of as the size of the imaginary sample and as such it defines our confidence in the prior. The conjugate prior for the joint in this case is called the *normal-gamma distribution*

NORMAL-GAMMA DISTRIBUTION

$$\begin{aligned} p(\mu, \lambda) &\sim \text{normal-gamma}(\mu_0, \kappa_0, a_0, b_0) \\ &= \mathcal{N}(\mu, 1/(\kappa_0\lambda)) \cdot \text{gamma}(a_0, b_0) \end{aligned}$$

The posterior is

$$p(\mu, \lambda|\mathcal{X}) \sim \text{normal-gamma}(\mu_N, \kappa_N, a_N, b_N) \tag{16.11}$$

where

$$\begin{aligned} \kappa_N &= \kappa_0 + N \\ \mu_N &= \frac{\kappa_0\mu_0 + Nm}{\kappa_N} \\ a_N &= a_0 + N/2 \\ b_N &= b_0 + \frac{N}{2}s^2 + \frac{\kappa_0 N}{2\kappa_N}(m - \mu_0)^2 \end{aligned} \tag{16.12}$$

To make a prediction for new x, we integrate over the posterior:

$$p(x|\mathcal{X}) = \int\int p(x|\mu,\lambda)p(\mu,\lambda|\mathcal{X})d\mu d\lambda \tag{16.13}$$

$$\sim t_{2a_N}\left(\mu_N, \frac{b_N(\kappa_N+1)}{a_N\kappa_N}\right) \tag{16.14}$$

That is, we get a (nonstandardized) t distribution having the given mean and variance values with $2a_N$ degrees of freedom. In equation 16.8, we have a Gaussian distribution; here the mean is the same but because σ^2 is unknown, its estimation adds uncertainty, and we get a t distribution with wider tails. Sometimes, equivalently, instead of modeling the precision λ, we model σ^2 and for this, we can use the inverse gamma or the inverse chi-squared distribution; see Murphy 2007.

16.3.3 Multivariate Case: Unknown Mean, Unknown Covariance

If we have multivariate $\boldsymbol{x} \in \Re^d$, we use exactly the same approach, except for the fact that we need to use the multivariate versions of the distributions (Murphy 2012). We have

$$p(\boldsymbol{x}) \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Lambda})$$

PRECISION MATRIX

where $\boldsymbol{\Lambda} \equiv \boldsymbol{\Sigma}^{-1}$ is the *precision matrix*. We use a Gaussian prior (conditioned on $\boldsymbol{\Lambda}$) for the mean:

$$p(\boldsymbol{\mu}|\boldsymbol{\Lambda}) \sim \mathcal{N}_d(\boldsymbol{\mu}_0, (1/\kappa_0)\boldsymbol{\Lambda})$$

and for the precision matrix, the multivariate version of the gamma distribution is called the *Wishart distribution*:

WISHART DISTRIBUTION

$$p(\boldsymbol{\Lambda}) \sim \text{Wishart}(\nu_0, \mathbf{V}_0)$$

where ν_0, as with κ_0, corresponds to the strength of our prior belief.

NORMAL-WISHART DISTRIBUTION

The conjugate joint prior is the *normal-Wishart distribution*:

$$p(\boldsymbol{\mu},\boldsymbol{\Lambda}) = p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda})$$

$$\sim \text{normal-Wishart}(\boldsymbol{\mu}_0, \kappa_0, \nu_0, \mathbf{V}_0) \tag{16.15}$$

and the posterior is

$$p(\mu,\boldsymbol{\Lambda}|\mathcal{X}) \sim \text{normal-Wishart}(\boldsymbol{\mu}_N, \kappa_N, \nu_N, \mathbf{V}_N)$$

where

$$
\begin{aligned}
\kappa_N &= \kappa_0 + N \\
\boldsymbol{\mu}_N &= \frac{\kappa_0 \boldsymbol{\mu}_0 + N\boldsymbol{m}}{\kappa_N} \\
\nu_N &= \nu_0 + N \\
\mathbf{V}_N &= \left(\mathbf{V}_0^{-1} + \mathbf{C} + \frac{\kappa_0 N}{\kappa_N}(\boldsymbol{m}-\boldsymbol{\mu}_0)(\boldsymbol{m}-\boldsymbol{\mu}_0)^T\right)^{-1}
\end{aligned} \tag{16.16}
$$

and $\mathbf{C} = \sum_t (\boldsymbol{x}^t - \boldsymbol{m})(\boldsymbol{x}^t - \boldsymbol{m})^T$ is the scatter matrix.

To make a prediction for new $\boldsymbol{x}$, we integrate over the joint posterior:

$$
p(\boldsymbol{x}|\mathcal{X}) = \int\int p(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Lambda}) p(\boldsymbol{\mu},\boldsymbol{\Lambda}|\mathcal{X}) d\boldsymbol{\mu} d\boldsymbol{\Lambda} \tag{16.17}
$$

$$
\sim t_{\nu_N - d + 1}\left(\boldsymbol{\mu}_N, \frac{\kappa_N + 1}{\kappa_N(\nu_N - d + 1)}(\mathbf{V}_N)^{-1}\right) \tag{16.18}
$$

That is, we get a (nonstandardized) multivariate t distribution having this mean and covariance with $\nu_N - d + 1$ degrees of freedom.

16.4 Bayesian Estimation of the Parameters of a Function

We now discuss the case where we estimate the parameters, not of a distribution, but some function of the input, for regression or classification. Again, our approach is to consider these parameters as random variables with a prior distribution and use Bayes' rule to calculate a posterior distribution. We can then either evaluate the full integral, approximate it, or use the MAP estimate.

16.4.1 Regression

Let us take the case of a linear regression model:

$$
r = \boldsymbol{w}^T \boldsymbol{x} + \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, 1/\beta) \tag{16.19}
$$

where β is the precision of the additive noise (assume that one of the d inputs is always +1).

The parameters are the weights $\boldsymbol{w}$ and we have a sample $\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_{t=1}^N$ where $\boldsymbol{x} \in \Re^d$ and $r^t \in \Re$. We can break it down into a matrix of inputs and a vector of desired outputs as $\mathcal{X} = [\mathbf{X}, \boldsymbol{r}]$ where $\mathbf{X}$ is $N \times d$ and $\boldsymbol{r}$ is $N \times 1$. From equation 16.19, we have

$$
p(r^t|\boldsymbol{x}^t, \boldsymbol{w}, \beta) \sim \mathcal{N}(\boldsymbol{w}^T \boldsymbol{x}, 1/\beta)
$$

We saw previously in section 4.6 that the log likelihood is

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}|\mathcal{X}) \equiv \log p(\mathcal{X}|\boldsymbol{w}) &= \log p(\boldsymbol{r}, \mathbf{X}|\boldsymbol{w}) \\ &= \log p(\boldsymbol{r}|\mathbf{X}, \boldsymbol{w}) + \log p(\mathbf{X})\end{aligned}$$

where the second term is a constant, independent of the parameters. We expand the first term as

$$\begin{aligned}\log p(\boldsymbol{r}|\mathbf{X}, \boldsymbol{w}, \beta) &= \log \prod_t p(r^t|\boldsymbol{x}^t, \boldsymbol{w}, \beta) \\ &= -N \log(\sqrt{2\pi}) + N \log \sqrt{\beta} - \frac{\beta}{2} \sum_t (r^t - \boldsymbol{w}^T \boldsymbol{x}^t)^2 \end{aligned} \tag{16.20}$$

For the case of the ML estimate, we find $\boldsymbol{w}$ that maximizes this, or equivalently, minimizes the last term that is the sum of the squared error. It can be rewritten as

$$\begin{aligned}E &= \sum_{t=1}^{N} (r^t - \boldsymbol{w}^T \boldsymbol{x}^t)^2 = (\boldsymbol{r} - \mathbf{X}\boldsymbol{w})^T (\boldsymbol{r} - \mathbf{X}\boldsymbol{w}) \\ &= \boldsymbol{r}^T \boldsymbol{r} - 2\boldsymbol{w}^T \mathbf{X}^T \boldsymbol{r} + \boldsymbol{w}^T \mathbf{X}^T \mathbf{X} \boldsymbol{w}\end{aligned}$$

Taking the derivative with respect to $\boldsymbol{w}$ and setting it to 0

$$-2\mathbf{X}^T \boldsymbol{r} + 2\mathbf{X}^T \mathbf{X} \boldsymbol{w} = 0 \Rightarrow \mathbf{X}^T \mathbf{X} \boldsymbol{w} = \mathbf{X}^T \boldsymbol{r}$$

we get the maximum likelihood estimator (we have previously derived this in section 5.8):

$$\boldsymbol{w}_{ML} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{r} \tag{16.21}$$

Having calculated the parameters, we can now do prediction. Given new input $\boldsymbol{x}'$, the response is calculated as

$$r' = \boldsymbol{w}_{ML}^T \boldsymbol{x}' \tag{16.22}$$

In the general case, for any model, $g(\boldsymbol{x}|\boldsymbol{w})$, for example, a multilayer perceptron where $\boldsymbol{w}$ are all the weights, we minimize, for example, using gradient descent:

$$E(\mathcal{X}|\boldsymbol{w}) = \left[r^t - g(x^t|\boldsymbol{w})\right]^2$$

and $\boldsymbol{w}_{LSQ}$ that minimizes it is called the *least squares estimator*. Then for new x', the prediction is calculated as

$$r' = g(\boldsymbol{x}'|\boldsymbol{w}_{LSQ})$$

In the case of the Bayesian approach, we define a Gaussian prior for the parameters:

$$p(\boldsymbol{w}) \sim \mathcal{N}(\mathbf{0}, (1/\alpha)\mathbf{I})$$

which is a conjugate prior, and for the posterior, we get

$$p(\boldsymbol{w}|\mathcal{X}, \boldsymbol{r}) \sim \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$

where

$$\begin{aligned} \boldsymbol{\mu}_N &= \beta \boldsymbol{\Sigma}_N \mathbf{X}^T \boldsymbol{r} \\ \boldsymbol{\Sigma}_N &= (\alpha \mathbf{I} + \beta \mathbf{X}^T \mathbf{X})^{-1} \end{aligned} \tag{16.23}$$

To calculate the output for new $\boldsymbol{x}'$, we integrate over the full posterior

$$r' = \int (\boldsymbol{w}^T \boldsymbol{x}') p(\boldsymbol{w}|\mathcal{X}, \boldsymbol{r}) d\boldsymbol{w}$$

The graphical model for this is shown in figure 14.7.

If we want to use a point estimate, the MAP estimator is

$$\boldsymbol{w}_{MAP} = \boldsymbol{\mu}_N = \beta(\alpha \mathbf{I} + \beta \mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{r} \tag{16.24}$$

and in calculating the output for input $\boldsymbol{x}'$, we replace the density with a single point, namely, the mean:

$$r' = \boldsymbol{w}_{MAP}^T \boldsymbol{x}'$$

We can also calculate the variance of our estimate:

$$\text{Var}(r') = 1/\beta + (\boldsymbol{x}')^T \boldsymbol{\Sigma}_N \boldsymbol{x}' \tag{16.25}$$

Comparing equation 16.24 with the ML estimate of equation 16.21, this can be seen as regularization—that is, we add a constant α to the diagonal to better condition the matrix to be inverted.

The prior, $p(\boldsymbol{w}) \sim \mathcal{N}(\mathbf{0}, (1/\alpha)\mathbf{I})$, says that we expect the parameters to be close to 0 with spread inversely proportional to α. When $\alpha \to 0$, we have a flat prior and the MAP estimate converges to the ML estimate.

We see in figure 16.4 that if we increase α, we force parameters to be closer to 0 and the posterior distribution moves closer to the origin and shrinks. If we decrease β, we assume noise with higher variance and the posterior also has higher variance.

If we take the log of the posterior, we have

$$\begin{aligned} \log p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{r}) &\propto \log p(\boldsymbol{r}|\boldsymbol{w}, \mathbf{X}) + \log p(\boldsymbol{w}) \\ &= -\frac{\beta}{2} \sum_t (r^t - \boldsymbol{w}^T \boldsymbol{x}^t)^2 - \frac{\alpha}{2} \boldsymbol{w}^T \boldsymbol{w} + c \end{aligned}$$

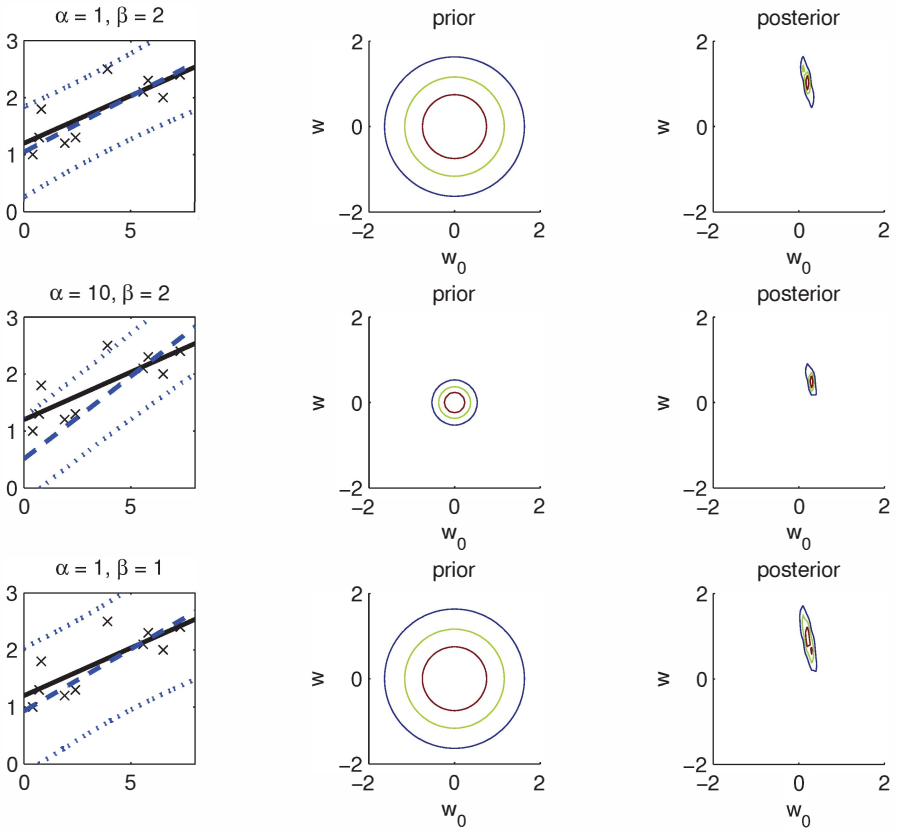

Figure 16.4 Bayesian linear regression for different values of α and β. To the left: crosses that are the data points and the straight line that is the ML solution. The MAP solution with one standard deviation error bars are also shown dashed. Center: prior density centered at 0 and variance $1/\alpha$. To the right: posterior density whose mean is the MAP solution. We see that when α is increased, the variance of the prior shrinks and the line moves closer to the flat 0 line. When β is decreased, more noise is assumed and the posterior density has higher variance.

which we maximize to find the MAP estimate. In the general case, given our model $g(\boldsymbol{x}|\boldsymbol{w})$, we can write an augmented error function

$$E_{ridge}(\boldsymbol{w}|\mathcal{X}) = \sum_t [r^t - g(\boldsymbol{x}^t|\boldsymbol{w})]^2 + \lambda \sum_i w_i^2$$

RIDGE REGRESSION

with $\lambda \equiv \alpha/\beta$. This is known as *parameter shrinkage* or *ridge regression* in statistics. In section 4.8, we called this *regularization*, and in section 11.9, we called this *weight decay* in neural networks. The first term is the negative log of the likelihood, and the second term penalizes w_i away from 0 (as dictated by α of the prior).

Though this approach reduces $\sum_i w_i^2$, it does not force individual w_i to 0; that is, it cannot be used for feature selection, namely, to determine which x_i are redundant. For this, one can use a *Laplacian prior* that uses the L_1 norm instead of the L_2 norm (Figueiredo 2003):

LAPLACIAN PRIOR

$$p(\boldsymbol{w}|\alpha) = \prod_i \frac{\alpha}{2} \exp(-\alpha|w_i|) = \left(\frac{\alpha}{2}\right)^d \exp\left(-\alpha \sum_i |w_i|\right)$$

The posterior probability is no longer Gaussian and the MAP estimate is found by minimizing

$$E_{lasso}(\boldsymbol{w}|\mathcal{X}) = \sum_t (r^t - \boldsymbol{w}^T\boldsymbol{x}^t)^2 + 2\sigma^2\alpha \sum_i |w_i|$$

where σ^2 is the variance of noise (for which we plug in our estimate). This is known as *lasso* (least absolute shrinkage and selection operator) (Tibshirani 1996). To see why L_1 induces sparseness, let us consider the case with two weights $[w_1, w_2]^T$ (Figueiredo 2003): $\|[1, 0^T\|_2 = \|[1/\sqrt{2}, 1/\sqrt{2}]^T\|_2 = 1$, whereas $\|[1, 0]^T\|_1 = 1 < \|[1/\sqrt{2}, 1/\sqrt{2}]^T\|_1 = \sqrt{2}$, and therefore L_1 prefers to set w_2 to 0 and use a large w_1, rather than having small values for both.

LASSO

16.4.2 Regression with Prior on Noise Precision

Above, we assume that β, the precision of noise, is known and $\boldsymbol{w}$ is the only parameter we integrated on. If we do not know β, we can also define a prior on it. Just as we do in section 16.3, we can define a gamma prior:

$$p(\beta) \sim \text{gamma}(a_0, b_0)$$

and a prior on $\boldsymbol{w}$ conditioned on β:

$$p(\boldsymbol{w}|\beta) \sim \mathcal{N}(\boldsymbol{\mu}_0, \beta\boldsymbol{\Sigma}_0)$$

If $\boldsymbol{\mu}_0 = \mathbf{0}$ and $\boldsymbol{\Sigma}_0 = \alpha\mathbf{I}$, we get ridge regression, as we discussed above. We now can write a conjugate normal-gamma prior on parameters $\boldsymbol{w}$ and β:

$$p(\boldsymbol{w}, \beta) = p(\beta)p(\boldsymbol{w}|\beta) \sim \text{normal-gamma}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0, a_0, b_0)$$

It can be shown (Hoff 2009) that the posterior is

$$p(\boldsymbol{w}, \beta|\mathbf{X}, \boldsymbol{r}) \sim \text{normal-gamma}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N, a_N, b_N)$$

where

$$\begin{aligned} \boldsymbol{\Sigma}_N &= (\mathbf{X}^T\mathbf{X} + \boldsymbol{\Sigma}_0)^{-1} \\ \boldsymbol{\mu}_N &= \boldsymbol{\Sigma}_N(\mathbf{X}^T\boldsymbol{r} + \boldsymbol{\Sigma}_0\boldsymbol{\mu}_0) \\ a_N &= a_0 + N/2 \\ b_N &= b_0 + \frac{1}{2}(\boldsymbol{r}^T\boldsymbol{r} + \boldsymbol{\mu}_0^T\boldsymbol{\Sigma}_0\boldsymbol{\mu}_0 - \boldsymbol{\mu}_N^T\boldsymbol{\Sigma}_N\boldsymbol{\mu}_N) \end{aligned} \tag{16.26}$$

An example is given in figure 16.5 where we fit a polynomial of different degrees on a small set of instances—$\boldsymbol{w}$ corresponds to the vector of coefficients of the polynomial. We see that the maximum likelihood starts to overfit as the degree is increased.

MARKOV CHAIN MONTE CARLO SAMPLING

We use *Markov chain Monte Carlo sampling* to get the Bayesian fit as follows: We draw a β value from $p(\beta) \sim \text{gamma}(a_N, b_N)$, and then we draw a $\boldsymbol{w}$ from $p(\boldsymbol{w}|\beta) \sim \mathcal{N}(\boldsymbol{\mu}_N, \beta\boldsymbol{\Sigma}_N)$, which gives us one sampled model from the posterior $p(\boldsymbol{w}, \beta)$. Ten such samples are drawn for each degree, as shown in figure 16.5. The thick line is the average of those ten models and is an approximation of the full integral; we see that even with ten samples, we get a reasonable and very smooth fit to the data. Note that any of the sampled models from the posterior is not necessarily any better than the maximum likelihood estimator; it is the averaging that leads to a smoother and hence better fit.

16.4.3 The Use of Basis/Kernel Functions

Using the Bayes' estimate of equation 16.23, the prediction is written as

$$\begin{aligned} r' &= (\boldsymbol{x}')^T\boldsymbol{w} \\ &= \beta(\boldsymbol{x}')^T\boldsymbol{\Sigma}_N\mathbf{X}^T\boldsymbol{r} \\ &= \sum_t \beta(\boldsymbol{x}')^T\boldsymbol{\Sigma}_N\boldsymbol{x}^t r^t \end{aligned}$$

DUAL REPRESENTATION

This is the *dual representation*. When we can write the parameter in

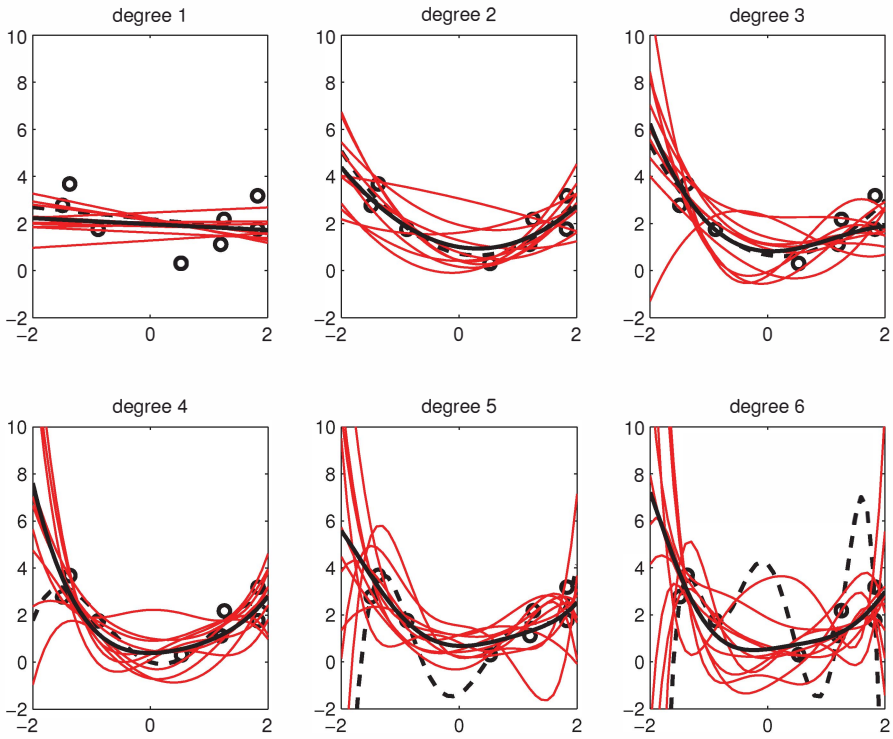

Figure 16.5 Bayesian polynomial regression example. Circles are the data points and the dashed line is the maximum likelihood fit, which overfits as the degree of the polynomial is increased. Thin lines are ten samples from the posterior $p(\boldsymbol{w}, \beta)$ and the thick line is their average.

terms of the training data, or a subset of it as in support vector machines (chapter 13), we can write the prediction as a function of the current input and past data. We can rewrite this as

$$r' = \sum_t K(\boldsymbol{x}', \boldsymbol{x}^t) r^t \tag{16.27}$$

where we define

$$K(\boldsymbol{x}', \boldsymbol{x}^t) = \beta(\boldsymbol{x}')^T \boldsymbol{\Sigma}_N \boldsymbol{x}^t \tag{16.28}$$

We know that we can generalize the linear kernel of equation 16.28 by using a nonlinear *basis function* $\phi(\boldsymbol{x})$ to map to a new space where we

BASIS FUNCTION

fit the linear model. In such a case, instead of the d-dimensional $\boldsymbol{x}$ we have the k-dimensional $\boldsymbol{\phi}(\boldsymbol{x})$ where k is the number of basis functions and instead of $N \times d$ data matrix $\mathbf{X}$, we have $N \times k$ image of the basis functions $\boldsymbol{\Phi}$.

During test, we have

$$\begin{aligned} r' &= \boldsymbol{\phi}(\boldsymbol{x}')^T \boldsymbol{w} \text{ where } \boldsymbol{w} = \beta \boldsymbol{\Sigma}_N^{\phi} \boldsymbol{\Phi}^T \boldsymbol{r} \text{ and } \boldsymbol{\Sigma}_N^{\phi} = \left(\alpha \mathbf{I} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \\ &= \beta \boldsymbol{\phi}(\boldsymbol{x}')^T \boldsymbol{\Sigma}_N^{\phi} \boldsymbol{\Phi}^T \boldsymbol{r} \\ &= \sum_t \beta \boldsymbol{\phi}(\boldsymbol{x}')^T \boldsymbol{\Sigma}_N^{\phi} \boldsymbol{\phi}(\boldsymbol{x}^t) r^t \\ &= \sum_t K(\boldsymbol{x}', \boldsymbol{x}^t) r^t \end{aligned} \tag{16.29}$$

where we define

$$K(\boldsymbol{x}', \boldsymbol{x}^t) = \beta \boldsymbol{\phi}(\boldsymbol{x}')^T \boldsymbol{\Sigma}_N^{\phi} \boldsymbol{\phi}(\boldsymbol{x}^t) \tag{16.30}$$

as the equivalent kernel. This is the dual representation in the space of $\boldsymbol{\phi}(\boldsymbol{x})$. We see that we can write our estimate as a weighted sum of the effects of instances in the training set where the effect is given by the *kernel function* $K(\boldsymbol{x}', \boldsymbol{x}^t)$; this is similar to the nonparametric kernel smoothers we discussed in chapter 8, or the kernel machines of chapter 13.

KERNEL FUNCTION

Error bars can be defined using

$$\text{Var}(r') = \beta^{-1} + \boldsymbol{\phi}(\boldsymbol{x}')^T \boldsymbol{\Sigma}_N^{\phi} \boldsymbol{\phi}(\boldsymbol{x}')$$

An example is given in figure 16.6 for the linear, quadratic, and sixth-degree kernels. This is equivalent to the polynomial regression we see in figure 16.5, except that here we use the dual representation and the polynomial coefficients $\boldsymbol{w}$ are embedded in the kernel function. We see that just as in regression proper where we can work on the original $\boldsymbol{x}$ or $\boldsymbol{\phi}(\boldsymbol{x})$, in Bayesian regression too we can work on the preprocessed $\boldsymbol{\phi}(\boldsymbol{x})$, defining parameters in that space. Later on in this chapter, we are going to see Gaussian processes where we can define and use $K(\boldsymbol{x}, \boldsymbol{x}^t)$ directly without needing to calculate $\boldsymbol{\phi}(\boldsymbol{x})$.

16.4.4 Bayesian Classification

In a two-class problem, we have a single output, and assuming a linear model, we have

$$P(C_1 | \boldsymbol{x}^t) = y^t = \text{sigmoid}(\boldsymbol{w}^T \boldsymbol{x}^t)$$

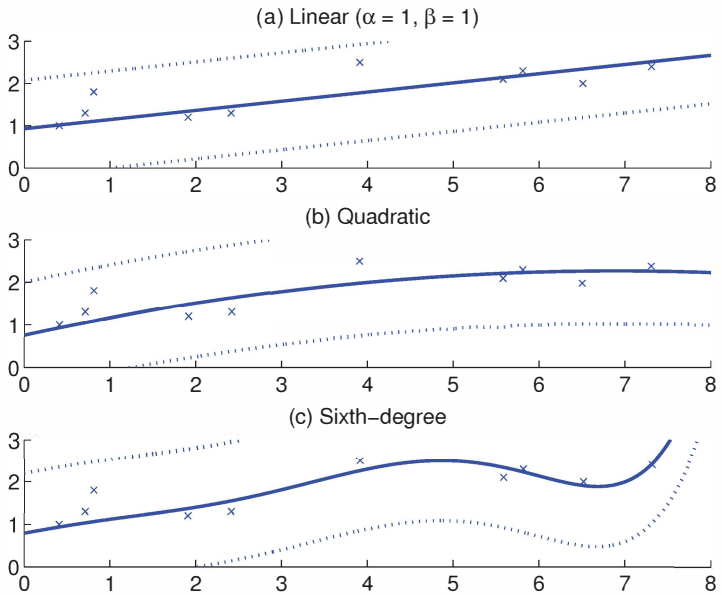

Figure 16.6 Bayesian regression using kernels with one standard deviation error bars: (a) linear: $\boldsymbol{\phi}(x) = [1, x]^T$, (b) quadratic: $\boldsymbol{\phi}(x) = [1, x, x^2]^T$, and (c) sixth degree: $\boldsymbol{\phi}(x) = [1, x, x^2, x^3, x^4, x^5, x^6]^T$.

The log likelihood of a Bernoulli sample is given as

$$\mathcal{L}(\boldsymbol{r}|\mathbf{X}) = \sum_t r^t \log y_t + (1 - r^t) \log(1 - y^t)$$

which we maximize, or minimize its negative log—the cross-entropy—to find the ML estimate, for example, using gradient descent. This is called *logistic discrimination* (section 10.7).

In the case of the Bayesian approach, we assume a Gaussian prior

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{m}_0, \mathbf{S}_0) \tag{16.31}$$

and the log of the posterior is given as

$$\begin{aligned} \log p(\boldsymbol{w}|\boldsymbol{r}, \mathbf{X}) &\propto \log p(\boldsymbol{w}) + \log p(\boldsymbol{r}|\boldsymbol{w}, \mathbf{X}) \\ &= -\frac{1}{2}(\boldsymbol{w} - \boldsymbol{m}_0)^T \mathbf{S}_0^{-1} (\boldsymbol{w} - \boldsymbol{m}_0) \\ &\quad + \sum_t r^t \log y_t + (1 - r^t) \log(1 - y^t) + c \end{aligned} \tag{16.32}$$

LAPLACE APPROXIMATION

This posterior distribution is no longer Gaussian, and we cannot integrate exactly. We can use *Laplace approximation*, which works as follows (MacKay 2003). Let us say we want to approximate some distribution $f(x)$, not necessarily normalized (to integrate to 1). In Laplace approximation, we find the mode of $f(x)$, x_0, fit a Gaussian $q(x)$ centered there with covariance given by the curvature of $f(x)$ around that mean, and then if we want to integrate, we integrate this fitted Gaussian instead.

To find the variance of the Gaussian, we consider the Taylor expansion of $f(\cdot)$ at $x = x_0$

$$\log f(x) = \log f(x_0) - \frac{1}{2}a(x - x_0)^2 + \cdots$$

where

$$a \equiv - \frac{d}{dx^2} \log f(x) \Big|_{x=x_0}$$

Note that the first, linear term disappears because the first derivative is 0 at the mode. Taking exp, we have

$$f(x) = f(x_0) \exp\left[-\frac{a}{2}(x - x_0)^2\right]$$

To normalize $f(x)$, we consider that in a Gaussian distribution

$$\int \frac{1}{\sqrt{2\pi}(1/\sqrt{a})} \exp\left[-\frac{a}{2}(x - x_0)^2\right] = 1 \Rightarrow \int \exp\left[-\frac{a}{2}(x - x_0)^2\right] = \sqrt{a/2\pi}$$

and therefore

$$q(x) = \sqrt{a/2\pi} \exp\left[-\frac{a}{2}(x - x_0)^2\right] \sim \mathcal{N}(x_0, 1/a)$$

In the multivariate setting where $\boldsymbol{x} \in \Re^d$, we have

$$\log f(\boldsymbol{x}) = \log f(\boldsymbol{x}_0) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^T \mathbf{A}(\boldsymbol{x} - \boldsymbol{x}_0) + \cdots$$

where $\mathbf{A}$ is the (Hessian) matrix of second derivatives:

$$\mathbf{A} = - \nabla\nabla \log f(\boldsymbol{x}) \big|_{\boldsymbol{x}=\boldsymbol{x}_0}$$

The Laplace approximation is then

$$f(\boldsymbol{x}) = \frac{|\mathbf{A}|^{1/2}}{(2\pi)^{d/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_0)^T \mathbf{A}(\boldsymbol{x} - \boldsymbol{x}_0)\right] \sim \mathcal{N}_d(\boldsymbol{x}_0, \mathbf{A}^{-1})$$

Having now discussed how to approximate, we can now use it for the posterior density. The MAP estimate, $\boldsymbol{w}_{MAP}$—the mode of $p(\boldsymbol{w}|\boldsymbol{r}, \mathbf{X})$—is

taken as the mean, and the covariance matrix is given by the inverse of the matrix of the second derivatives of the negative log likelihood:

$$\mathbf{S}_N = -\nabla\nabla \log p(\boldsymbol{w}|\boldsymbol{r}, \mathbf{X}) = \mathbf{S}_0^{-1} + \sum_t y^t(1-y^t)\boldsymbol{x}^t(\boldsymbol{x}^t)^T$$

We then integrate over this Gaussian to estimate the class probability:

$$P(C_1|\boldsymbol{x}) = y = \int \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x})q(\boldsymbol{w})d\boldsymbol{w}$$

where $q(\boldsymbol{w}) \sim \mathcal{N}(\boldsymbol{w}_{MAP}, \mathbf{S}_N^{-1})$. A further complication is that we cannot integrate analytically over a Gaussian convolved with a sigmoid. If we use the *probit function* instead, which has the same S-shape as the sigmoid, an analytical solution is possible (Bishop 2006).

PROBIT FUNCTION

16.5 Choosing a Prior

Defining the prior is the subjective part of Bayesian estimation and as such should be done with care. It is best to define robust priors with heavy tails so as not to limit the parameter space too much; in the extreme case of no prior preference, one can use an uninformative prior and methods have been proposed for this purpose, for example, Jeffreys prior (Murphy 2012). Sometimes our choice of a prior is also motivated by simplicity—for example, a conjugate prior makes inference quite easy.

One critical decision is when to take a parameter as a constant and when to define it as a random variable with a prior and to be integrated (averaged) out. For example, in section 16.4.1, we assume that we know the noise precision whereas in section 16.4.2, we assume we do not and define a gamma prior on it. Similarly for the spread of weights in linear regression, we assume a constant α value but can also define a prior on it and average it out if we want. Of course, this makes the prior more complicated and the whole inference more difficult but averaging over α should be preferred if we do not know what the good value for α is.

Another decision is how high to go in defining the priors. Let us say we have parameter θ and we define a posterior on it. In prediction, we have

$$\text{Level I: } p(x|\mathcal{X}) = \int p(x|\theta)p(\theta|\mathcal{X})d\theta$$

where $p(\theta|\mathcal{X}) \propto p(\mathcal{X}|\theta)p(\theta)$. If we believe that we cannot define a good

$p(\theta)$ but that it depends on some other variable, we can condition θ on a hyper parameter α and integrate it out:

$$\text{Level II: } p(x|\mathcal{X}) = \int p(x|\theta)p(\theta|\mathcal{X},\alpha)p(\alpha)d\theta d\alpha$$

This is called a *hierarchical prior*. This really makes the inference rather difficult because we need to integrate on two levels. One shortcut is to test different values α on the data, choose the best α^*, and just use that value:

$$\text{Level II ML: } p(x|\mathcal{X}) = \int p(x|\theta)p(\theta|\mathcal{X},\alpha^*)d\theta$$

This is called *level II maximum likelihood* or *empirical Bayes*.

16.6 Bayesian Model Comparison

Assume we have many models $\mathcal{M}_j$, each with its own set of parameters θ_j, and we want to compare these models. For example, in figure 16.5, we have polynomials of different degrees and let us say we want to check how well they fit the data.

For a given model $\mathcal{M}$ and parameter θ, the likelihood of data is $p(\mathcal{X}|\mathcal{M},\theta)$.

MARGINAL LIKELIHOOD

To get the Bayesian *marginal likelihood* for a given model, we average over θ:

$$p(\mathcal{X}|\mathcal{M}) = \int p(\mathcal{X}|\theta,\mathcal{M})p(\theta|\mathcal{M})d\theta \tag{16.33}$$

MODEL EVIDENCE

This is also called *model evidence*. For example, in the polynomial regression example above, for a given degree, we have

$$p(\boldsymbol{r}|\mathbf{X},\mathcal{M}) = \int\int p(\boldsymbol{r}|\mathbf{X},\boldsymbol{w},\beta,\mathcal{M})p(\boldsymbol{w},\beta|\mathcal{M})d\boldsymbol{w}d\beta$$

where $p(\boldsymbol{w},\beta|\mathcal{M})$ is the prior assumed for model $\mathcal{M}$. We can then calculate the posterior probability of a model given the data:

$$p(\mathcal{M}|\mathcal{X}) = \frac{p(\mathcal{X}|\mathcal{M})p(\mathcal{M})}{p(\mathcal{X})} \tag{16.34}$$

where $P(\mathcal{M})$ is the prior distribution defined over models. The nice property of the Bayesian approach is that even if those priors are taken uniform, the marginal likelihood, because it averages over all θ, favors simpler models. Let us assume we have models in increasing complexity, for example, polynomials with increasing degree.

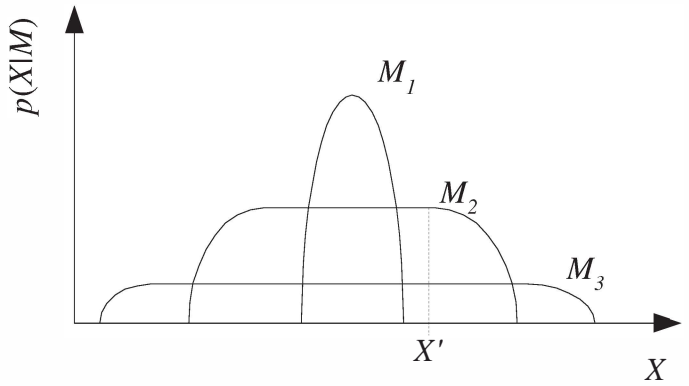

Figure 16.7 Bayesian model comparison favors simpler models. $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ are three models in increasing complexity. The x axis is the space of all datasets with N instances. A complex model can fit more datasets but spreads itself thin over the space of all possible datasets of size N; a simpler model can fit fewer datasets but each with a heavier probability. For a particular dataset $\mathcal{X}'$, if both can fit, the simpler model will have higher marginal likelihood (MacKay 2003).

Let us say we have a dataset $\mathcal{X}$ with N instances. A more complex model will be able to fit more of such datasets reasonably well compared with a simpler model—consider choosing randomly three points in a plane; the number of such triples that can be fitted by a line is much fewer than the number of triples that can be fitted by a quadratic. Given that $\sum_{\mathcal{X}} p(\mathcal{X}|\mathcal{M}) = 1$, because for a complex model there are more possible $\mathcal{X}$ where it can make a reasonable fit, if there is a fit, the value of $p(\mathcal{X}'|\mathcal{M})$ for some particular $\mathcal{X}'$ is going to be smaller—see figure 16.7. Hence for a simpler model $p(\mathcal{M}|\mathcal{X})$ will be higher (even if we assume that the priors, $p(\mathcal{M})$, are equal); this is the Bayesian interpretation of Occam's razor (MacKay 2003).

For the polynomial fitting example of figure 16.5, a comparison of likelihood and the marginal likelihood is shown in figure 16.8. We see that likelihood increases when complexity increases, which implies overfitting, but the marginal likelihood increases until the correct degree and then starts decreasing; this is because there are many more complex models that fit badly to the data and they pull the likelihood down as

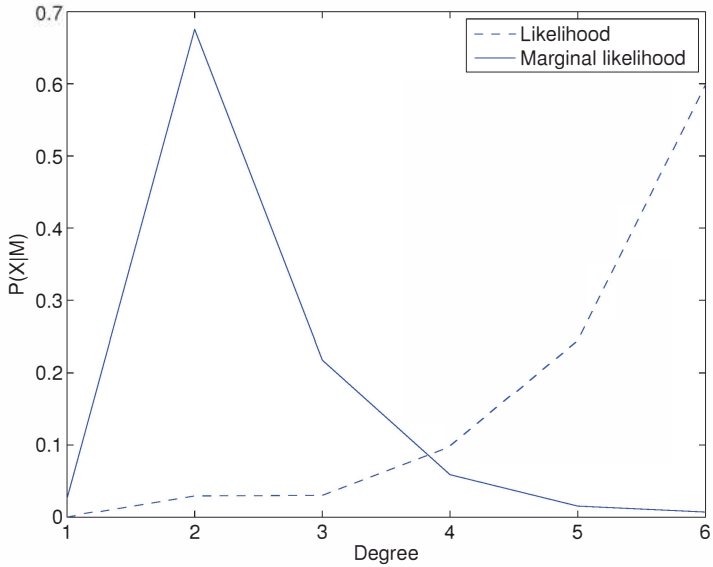

Figure 16.8 Likelihood versus marginal likelihood for the polynomial regression example. Though the likelihood increases as the degree of the polynomial increases, the marginal likelihood that averages over parameter values make a peak at the right complexity and then levels off.

we average over them.

If we have two models $\mathcal{M}_0$ and $\mathcal{M}_1$, we can compare them

$$\frac{P(\mathcal{M}_1|\mathcal{X})}{P(\mathcal{M}_0|\mathcal{X})} = \frac{P(\mathcal{X}|\mathcal{M}_1)}{P(\mathcal{X}|\mathcal{M}_0)}\frac{P(\mathcal{M}_1)}{P(\mathcal{M}_0)}$$

and we have higher belief in $\mathcal{M}_1$ if this ratio is higher than 1, and in $\mathcal{M}_0$ otherwise.

BAYES FACTOR

There are two important points here: One, the ratio of the two marginal likelihoods is called the *Bayes factor* and is enough for model selection even if the two priors are taken equal. Second, in the Bayesian approach, we do not choose among models and we do not do model selection; but in keeping with the spirit of the Bayesian approach, we average over their predictions rather than choosing one and discarding the rest. For instance, in the polynomial regression example above, rather than choosing one degree, it is best to take a weighted average over all degrees weighted by their marginal likelihoods.

BAYESIAN INFORMATION CRITERION

A related approach is the *Bayesian information criterion* (BIC) where using Laplace approximation (section 16.4.4), equation 16.33 is approximated as

$$\log p(\mathcal{X}|\mathcal{M}) \approx \text{BIC} \equiv \log p(\mathcal{X}|\theta_{ML}, \mathcal{M}) - \frac{|\mathcal{M}|}{2} \log N \tag{16.35}$$

The first term is the likelihood using the ML estimator and the second term is a penalty for complex models: $|\mathcal{M}|$ is a measure of model complexity, in other words, the degrees of freedom in the model—for example, the number of coefficients in a linear regression model. As model complexity increases, the first term may be higher but the second penalty term compensates for this.

AKAIKE'S INFORMATION CRITERION

A related, but not Bayesian, approach is *Akaike's information criterion* (AIC), which is written as

$$\text{AIC} \equiv \log p(\mathcal{X}|\theta_{ML}, \mathcal{M}) - |\mathcal{M}| \tag{16.36}$$

where again we see a penalty term that is proportional to the model complexity. It is important to note here that in such criteria, $|\mathcal{M}|$ represents the "effective" degrees of freedom and not simply the number of adjustable parameters in the model. For example in a multilayer perceptron (chapter 11), the effective degrees of freedom is much less than the number of adjustable connection weights.

One interpretation of the penalty term is as a term of "optimism" (Hastie, Tibshirani, and Friedman 2011). In a complex model, the ML estimator would overfit and hence be a very optimistic indicator of model performance; therefore, it should be cut back proportional to the model complexity.

16.7 Bayesian Estimation of a Mixture Model

In section 7.2, we discuss the mixture model where we write the density as a weighted sum of component densities. Let us remember equation 7.1:

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} P(\mathcal{G}_i) p(\boldsymbol{x}|\mathcal{G}_i)$$

where $P(\mathcal{G}_i)$ are the mixture proportions and $p(\boldsymbol{x}|\mathcal{G}_i)$ are the component densities. For example, in Gaussian mixtures, we have $p(\boldsymbol{x}|\mathcal{G}_i) \sim$

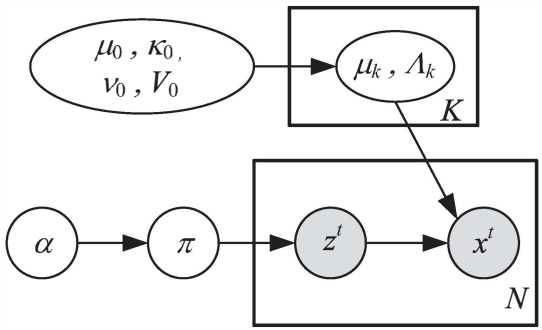

Figure 16.9 The generative graphical representation of a Gaussian mixture model.

$\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$, and defining $\pi_i \equiv P(G_i)$, we have the parameter vector $\Phi = \{\pi_i, \boldsymbol{\mu}_i, \Sigma_i\}_{i=1}^k$ that we need to learn from data $\mathcal{X} = \{\boldsymbol{x}^t\}_{t=1}^N$.

In section 7.4, we discussed the EM algorithm that is a maximum likelihood procedure:

$$\Phi_{MLE} = \arg\max_{\Phi} \log p(\mathcal{X}|\Phi)$$

If we have a prior distribution $p(\Phi)$, we can devise a Bayesian approach. For example, the MAP estimator is

$$\Phi_{MAP} = \arg\max_{\Phi} \log p(\Phi|\mathcal{X}) = \arg\max_{\Phi} \log p(\mathcal{X}|\Phi) + \log p(\Phi) \tag{16.37}$$

Let us now write down the prior. Π_i are multinomial variables and for them, we can use a Dirichlet prior as we discuss in section 16.2.1. For the Gaussian components, for the mean and precision (inverse covariance) matrix, we can use a normal-Wishart prior as we discuss in section 16.3:

$$\begin{aligned} p(\Phi) &= p(\boldsymbol{\pi}) \prod_i p(\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i) \\ &= \text{Dirichlet}(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_i \text{normal-Wishart}(\boldsymbol{\mu}_0, \kappa_0, \nu_0, \mathbf{V}_0) \end{aligned} \tag{16.38}$$

So in using EM in this case, the E-step does not change, but in the M-step we maximize the posterior with this prior (Murphy 2012). Adding log of the posterior, equation 7.10 becomes

$$\begin{aligned} \mathcal{Q}(\Phi|\Phi^l) &= \sum_t \sum_i h_i^t \log \pi_i + \sum_t \sum_i h_i^t \log p_i(\boldsymbol{x}^t|\Phi^l) + \log p(\boldsymbol{\pi}) + \\ &\quad \sum_i \log p(\boldsymbol{\mu}_i, \boldsymbol{\Lambda}_i) \end{aligned} \tag{16.39}$$

where $h_i^t \equiv E[z_i^t]$ are the soft labels estimated in the E-step using the current values of Φ. The M-step MAP estimate for the mixture proportions are as follows (based on equation 16.4):

$$\pi_i^{l+1} = \frac{\alpha_i + N_i - 1}{\sum_i \alpha_i + N - k} \tag{16.40}$$

where $N_i = \sum_i h_i^t$. The M-step MAP estimates for the Gaussian component density parameters are as follows (based on equation 16.16):

$$\begin{aligned} \boldsymbol{\mu}_i^{l+1} &= \frac{\kappa_0 \boldsymbol{\mu}_0 + N_i \boldsymbol{m}_i}{\kappa_0 + N_i} \\ \boldsymbol{\Lambda}_i^{l+1} &= \left(\frac{\mathbf{V}_0^{-1} + \mathbf{C}_i + \mathbf{S}_i}{\nu_0 + N_i + d + 2} \right)^{-1} \end{aligned} \tag{16.41}$$

where $\boldsymbol{m}_i = \sum_t h_i^t / N_i$ is the component mean, $\mathbf{C}_i = \sum_t h_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T$ is the within-scatter matrix for component i, and $\mathbf{S}_i = (\kappa_0 N_i)/(\kappa_0 + N_i)(\boldsymbol{m}_i - \boldsymbol{\mu}_0)(\boldsymbol{m}_i - \boldsymbol{\mu}_0)^T$ is the between-scatter of component i around the prior mean.

If we take $\alpha_i = 1/K$, this is a uniform prior. We can take $\kappa_0 = 0$ not to bias the mean estimates unless we do have some prior information about them. We can take $\mathbf{V}_0$ as the identity matrix and hence the MAP estimate has a regularizing effect.

The mixture density is shown as a generative graphical model in figure 16.9.

Once we know how to do basic blocks in a Bayesian manner, we can combine them to get more complicated models. For example, combining the mixture model we have here and the linear regression model we discuss in section 16.4.1, we can write the Bayesian version of the mixture of experts model (section 12.8) where we cluster the data into components and learn a separate linear regression model in each component at the same time. The posterior turns out to be rather nasty and Waterhouse et al. 1996 use variational approximation, which, roughly speaking, works as follows.

VARIATIONAL APPROXIMATION

We remember that in Laplace approximation, we approximate $p(\theta|\mathcal{X})$ by a Gaussian and integrate over the Gaussian instead. In *variational approximation*, we approximate the posterior by a density $q(\mathcal{Z}|\psi)$ whose parameters ψ are adjustable (Jordan et al. 1999; MacKay 2003; Bishop 2006). Hence, it is more general because we are not restricted to use a Gaussian density. Here, $\mathcal{Z}$ contains all the latent variables in the model

and the parameters θ, and ψ of the approximating model $q(\mathcal{Z}|\psi)$ are adjusted such that $q(\mathcal{Z}|\psi)$ is as close as possible to $p(\mathcal{Z}|\mathcal{X})$.

KULLBACK-LEIBLER DISTANCE

We define as the *Kullback-Leibler distance* between the two:

$$D_{\mathrm{KL}}(q||p) = \sum_{\mathcal{Z}} q(\mathcal{Z}|\psi) \log \frac{q(\mathcal{Z}|\psi)}{p(\mathcal{Z}|\mathcal{X})} \tag{16.42}$$

To make life easier, the set of latent variables (including the parameters) is assumed to be partitioned into subsets $\mathcal{Z}_i, i = 1, \ldots, k$, such that the variational distribution can be factorized:

$$q(\mathcal{Z}|\psi) = \prod_{i=1}^{k} q_i(\mathcal{Z}_i|\psi_i) \tag{16.43}$$

Adjustment of the parameters ψ_i in each factor is iterative, rather like the expectation-maximization algorithm we discussed in section 7.4. We start from (possibly random) initial values and while adjusting each, we use the expected values of the $\mathcal{Z}_j, j \neq i$ in a circular manner. This is called the *mean-field approximation*.

MEAN-FIELD APPROXIMATION

This factorization is an approximation. For example, in section 16.4.2 when we discuss regression, we write

$$p(\boldsymbol{w}, \beta) = p(\beta)p(\boldsymbol{w}|\beta)$$

because $\boldsymbol{w}$ is conditioned on β. A variational approximation would assume

$$p(\boldsymbol{w}, \beta) = p(\beta)p(\boldsymbol{w})$$

For example, in the mixture of experts model, the latent parameters are the component indices and the parameters are the parameters in the gating model, the regression weights in the local experts, the variance of the noise, and the hyperparameters of the priors for gating and regression weights; they are all factors (Waterhouse, MacKay, and Robinson 1996).

16.8 Nonparametric Bayesian Modeling

The models we discuss earlier in this chapter are all parametric, in the sense that we have models of fixed complexity with a set of parameters and these parameters are optimized using the data and the prior information. In chapter 8, we discussed nonparametric models where the training data makes up the model and hence model complexity depends on the

size of the data. Now we address how such a nonparametric approach can be used in the Bayesian setting.

A nonparametric model does not mean that the model has no parameters; it means that the number of parameters is not fixed and that their number can grow depending on the size of the data, or better still, depending on the complexity of the regularity that underlies the data. Such models are also sometimes called *infinite* models, in the sense that their complexity can keep on increasing with more data. In section 11.9, we discuss incremental neural network models where new hidden units are added when necessary and network is grown during training, but usually in parametric learning, adjusting model complexity is handled in an outer loop by checking performance on a separate validation set. The nonparametric Bayesian approach includes model adjustment in parameter training by using a suitable prior (Gershman and Blei 2012). This makes such models more flexible, and would have normally made them prone to overfitting if not for the Bayesian approach that alleviates this risk.

Because it is the parameters that grow, the priors on such parameters should be able to handle that growth and we will discuss three example prior distributions for three different type of machine learning applications, namely, Gaussian processes for supervised learning, Dirichlet processes for clustering, and beta processes for dimensionality reduction.

16.9 Gaussian Processes

Let us say we have the linear model $y = \mathbf{w}^T\mathbf{x}$. Then, for each $\mathbf{w}$, we have one line. Given a prior distribution $p(\mathbf{w})$, we get a distribution of lines, or to be more specific, for any $\mathbf{w}$, we get a distribution of y values calculated at x as $y(\mathbf{x}|\mathbf{w})$ when $\mathbf{w}$ is sampled from $p(\mathbf{w})$, and this is what we mean when we talk about a *Gaussian process*. We know that if $p(\mathbf{w})$ is Gaussian, each y is a linear combination of Gaussians and is also Gaussian; in particular, we are interested in the joint distribution of y values calculated at the N input data points, $\mathbf{x}^t, t = 1, \ldots, N$ (MacKay 1998).

GAUSSIAN PROCESS

We assume a zero-mean Gaussian prior

$$p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, (1/\alpha)\mathbf{I})$$

Given the $N \times d$ data points $\mathbf{X}$ and the $d \times 1$ weight vector, we write the

y outputs as

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{w} \tag{16.44}$$

which is N-variate Gaussian with

$$\begin{aligned} E[\boldsymbol{y}] &= \mathbf{X}E[\boldsymbol{w}] = \mathbf{0} \\ \text{Cov}(\boldsymbol{y}) &= E[\boldsymbol{y}\boldsymbol{y}^T] = \mathbf{X}E[\boldsymbol{w}\boldsymbol{w}^T]\mathbf{X}^T = \frac{1}{\alpha}\mathbf{X}\mathbf{X}^T \equiv \mathbf{K} \end{aligned} \tag{16.45}$$

where $\mathbf{K}$ is the (Gram) matrix with elements

$$K_{i,j} \equiv K(\boldsymbol{x}^i, \boldsymbol{x}^j) = \frac{(\boldsymbol{x}^i)^T\boldsymbol{x}^j}{\alpha}$$

COVARIANCE FUNCTION

This is known as the *covariance function* in the literature of Gaussian processes and the idea is the same as in kernel functions: If we use a set of basis functions $\boldsymbol{\phi}(\boldsymbol{x})$, we generalize from the dot product of the original inputs to the dot product of basis functions by a kernel

$$K_{i,j} = \frac{\boldsymbol{\phi}(\boldsymbol{x}^i)^T\boldsymbol{\phi}(\boldsymbol{x}^j)}{\alpha}$$

The actual observed output r is given by the line with added noise, $r = y + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \beta^{-1})$. For all N data points, we write it as

$$\boldsymbol{r} \sim \mathcal{N}_N(\mathbf{0}, \mathbf{C}_N) \text{ where } \mathbf{C}_N = \beta^{-1}\mathbf{I} + \mathbf{K} \tag{16.46}$$

To make a prediction, we consider the new data as the $(N+1)$st data point pair $(\boldsymbol{x}', r')$, and write the joint using all $N+1$ data points. We have

$$\boldsymbol{r}_{N+1} \sim \mathcal{N}_N(\mathbf{0}, \mathbf{C}_{N+1}) \tag{16.47}$$

where

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \boldsymbol{k} \\ \boldsymbol{k}^T & c \end{bmatrix}$$

with $\boldsymbol{k}$ being the $N \times 1$ dimensional vector of $K(\boldsymbol{x}', \boldsymbol{x}^t), t = 1, \ldots, N$ and $c = K(\boldsymbol{x}', \boldsymbol{x}') + \beta^{-1}$. Then to make a prediction, we calculate $p(r'|\boldsymbol{x}', \mathbf{X}, \boldsymbol{r})$, which is Gaussian with

$$\begin{aligned} E[r'|\boldsymbol{x}'] &= \boldsymbol{k}^T\mathbf{C}_N^{-1}\boldsymbol{r} \\ \text{Var}(r'|\boldsymbol{x}') &= c - \boldsymbol{k}^T\mathbf{C}_N^{-1}\boldsymbol{k} \end{aligned}$$

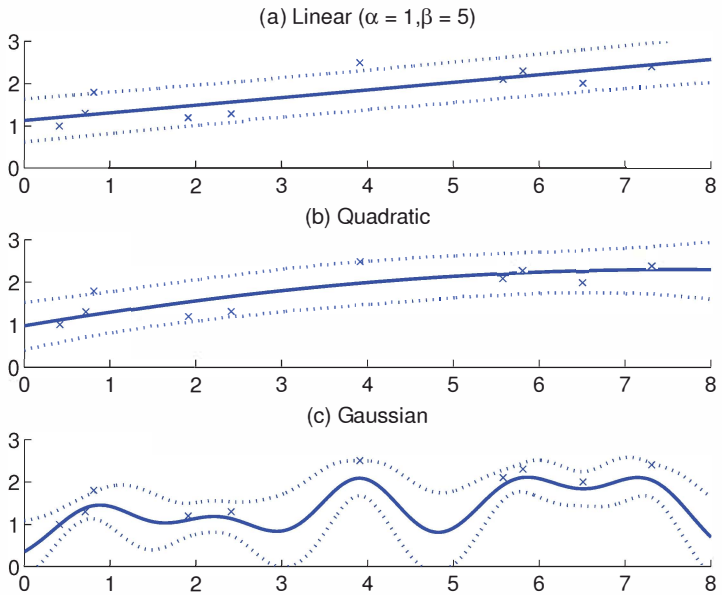

Figure 16.10 Gaussian process regression with one standard deviation error bars: (a) linear kernel, (b) quadratic kernel, and (c) Gaussian kernel with spread $s^2 = 0.5$.

An example shown in figure 16.10 uses linear, quadratic, and Gaussian kernels. The first two are defined as the dot product of their corresponding basis functions; the Gaussian kernel is defined directly as

$$K_G(\mathbf{x}^i, \mathbf{x}^j) = \exp\left[-\frac{\|\mathbf{x}^i - \mathbf{x}^j\|^2}{s^2}\right]$$

The mean, which is our point estimate (if we do not integrate over the full distribution), can also be written as a weighted sum of the kernel effects

$$E[r'|\mathbf{x}'] = \sum_t a^t K(\mathbf{x}^t, \mathbf{x}') \tag{16.48}$$

where a^t is the tth component of $\mathbf{C}_N^{-1}\mathbf{r}$. Or, we can write it as a weighted sum of the outputs of the training data points where weights are given by the kernel function

$$E[r'|\mathbf{x}'] = \sum_t r^t w^t \tag{16.49}$$

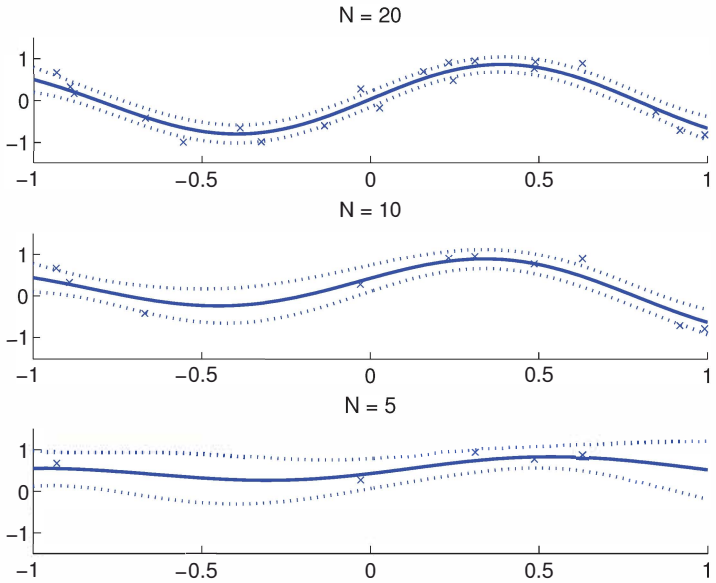

Figure 16.11 Gaussian process regression using a Gaussian kernel with $s^2 = 0.5$ and varying number of training data. We see how variance of the prediction is larger where there is few data.

where w^t is the tth component of $\boldsymbol{k}^T \mathbf{C}_N^{-1}$.

Note that we can also calculate the variance of a prediction at a point to get an idea about uncertainty in there, and it depends on the instances that affect the prediction in there. In the case of a Gaussian kernel, only instances within a locality are effective and prediction variance is high where there is little data in the vicinity (see figure 16.11).

Kernel functions can be defined and used for any application, as we have previously discussed in the context of kernel machines in chapter 13. The possibility of using kernel functions directly without needing to calculate or store the basis functions offers a great flexibility. Normally, given a training set, we first calculate the parameters, for example using equation 16.21, and then use the parameters to make predictions using equation 16.22, never needing the training set any more. This makes sense because generally the dimensionality of the parameters, which is generally $\mathcal{O}(d)$, is much lower than the size of the training set N.

When we work with basis functions, however, calculating the parameter explicitly may no longer be the case, because the dimensionality of the basis functions may be very high, even infinite. In such a case, it is cheaper to use the dual representation, taking into account the effects of training instances using kernel functions, as we do here. This idea is also used in nonparametric smoothers (chapter 8) and kernel machines (chapter 13).

The requirement here is that $\mathbf{C}_N$ be invertible and hence positive definite. For this, $\mathbf{K}$ should be semidefinite so that after adding $\beta^{-1} > 0$ to the diagonals, we get positive definiteness. We also see that the costliest operation is this inversion of the $N \times N$ matrix, which fortunately needs to be calculated only once (during training) and stored. Still, for large N, one may need an approximation.

When we use it for classification for a two-class problem, the output is filtered through a sigmoid, $y = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x})$, and the distribution of y is no longer Gaussian. The derivation is similar except that the conditional $p(r_{N+1}|\boldsymbol{x}_{N+1}, \mathbf{X}, \boldsymbol{r})$ is not Gaussian either and we need to approximate, for example, using Laplace approximation (Bishop 2006; Rasmussen and Williams 2006).

16.10 Dirichlet Processes and Chinese Restaurants

To explain a Dirichlet process, let us start with a metaphor: There is a Chinese restaurant with a lot of tables. Customers enter one by one; we start with the first customer who sits at the first table, and any subsequent customer can either sit at one of the occupied tables or go and start a new table. The probability that a customer sits at an occupied table is proportional to the number of customers already sitting at the table, and the probability that he or she sits at a new table depends on a parameter α. This is called a *Chinese restaurant process*:

CHINESE RESTAURANT PROCESS

$$\begin{aligned} \text{Join existing table } i \text{ with } P(z_i = 1) &= \frac{n_i}{\alpha + n - 1}, i = 1, \ldots, k \\ \text{Start new table with } P(z_{k+1} = 1) &= \frac{\alpha}{\alpha + n - 1} \end{aligned}$$

where n_i is the number of customers already starting at table i, $n = \sum_{i=1}^{k} n_i$ is the total number of customers. α is the propensity to start a new table and is the parameter of the process. Note that at each step,

DIRICHLET PROCESS

the sitting arrangement of customers define a partition of integers 1 to n into k subsets. This is called a *Dirichlet process* with parameter α.

We can apply this to clustering by making customer choices not only dependent on the table occupancies but also on the input. Let us say that this is not a Chinese restaurant but the dinner of a large conference, for example, NIPS. There is a large dining lounge with many tables, and in the evening, the conference participants enter the lounge one by one. They want to eat, but they also want to participate in interesting conversation. For that, they want to sit at a table where there are already many people sitting, but they also want to sit next to people having similar research interests. If they see no such table, they start a new table and expect incoming similar participants to find and join them.

Assume that instance/participant t is represented by a d-dimensional vector $\mathbf{x}^t$, and let us assume that such $\mathbf{x}^t$ are locally Gaussian distributed. This defines a Gaussian mixture over the whole space/dining lounge, and to have it Bayesian, we define priors on the parameters of the Gaussian components, as we discuss in section 16.7. To make it nonparametric, we define a Dirichlet process as the prior so a new component can be added when necessary, as follows:

$$\begin{aligned}\text{Join component } i \text{ with } P(z_i^t = 1) &\propto \frac{n_i}{\alpha + n - 1} p(\mathbf{x}^t | \mathcal{X}_i), i = 1, \ldots, k \\ \text{Start new component with } P(z_{k+1}^t) &\propto \frac{\alpha}{\alpha + n - 1} p(\mathbf{x}^t)\end{aligned}$$

$\mathcal{X}_i$ is the set of instances previously assigned to component i; using their data and the priors, we can calculate a posterior and integrating over it, we can calculate $p(\mathbf{x}^t | \mathcal{X}_i)$. Roughly speaking, the probability this new instance is assigned to component i will be high if there are already many instances in the component, that is, due to a high prior, or if $\mathbf{x}^t$ is similar to the instances already in $\mathcal{X}_i$. If none of the existing components have a high probability, a new component is added: $p(\mathbf{x}^t)$ is the marginal probability (integrated over the component parameter priors because there is no data).

Different α may lead to different numbers of clusters. To adjust α, we can use empirical Bayes, or also define a prior on it and average it out.

In chapter 7, when we talk about k-means clustering (section 7.3), we discuss leader-cluster algorithms where new clusters are added during training and as an example of that, in section 12.2.2, we discuss adaptive resonance theory where we add a new cluster if the distance to the center

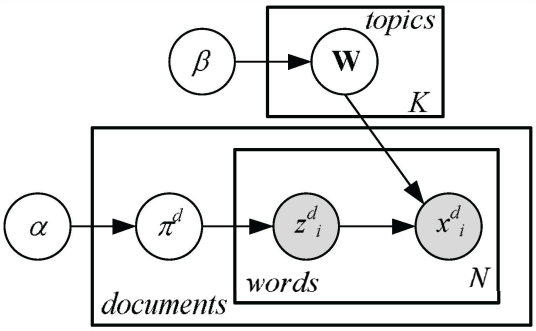

Figure 16.12 The graphical model for latent Dirichlet allocation.

of the nearest cluster is more than a vigilance value. What we have here is very similar: Assuming Gaussian components and diagonal covariance matrices, if the Euclidean distance to all clusters is high, all posteriors will be small and a new component will be added.

16.11 Latent Dirichlet Allocation

TOPIC MODELING

Let us see an application of the Bayesian approach in text processing, namely *topic modeling* (Blei 2012). In this age, there are digital repositories containing a very large number of documents—scientific articles, web pages, emails, blog posts, and so on—but finding a relevant topic for a query is very difficult, unless documents are manually annotated with topics, such as "arts," "sports," and so on. What we would like to is do this annotation automatically.

Assume we have a vocabulary with M words. Each document contains N words chosen from a number of topics in different proportions—that is, each document is a probability distribution over topics. A document may be partially "arts" and partially "politics," for example. Each topic in turn is defined as a mixture of the M words—that is, each topic corresponds to a probability distribution over words. For example, for the topic arts, the words "painting" and "sculpture" have a high probability, but the word "knee" has a low probability.

LATENT DIRICHLET ALLOCATION

In *latent Dirichlet allocation*, we define a generative process as follows (figure 16.12)—there are K topics, a vocabulary of M words, and all documents contain N words (Blei, Ng, and Jordan 2003):

To generate each document d, we first decide on the topics it will be about. These topic probabilities, $\pi_k^d, k = 1, \ldots, K$, define a multinomial distribution and are drawn from a Dirichlet prior with hyperparameter $\boldsymbol{\alpha}$ (section 16.2.1):

$$\boldsymbol{\pi}^d \sim \text{Dirichlet}_K(\boldsymbol{\alpha})$$

Once we know the topic distribution for document d, we generate its N words using it. In generating word i, first we decide on its particular topic by sampling from $\boldsymbol{\pi}$: We roll a die with K faces where face k has probability π_k. We define z_i^d as the outcome, it will be a value between 1 and K:

$$z_i^d \sim \text{Mult}_K(\boldsymbol{\pi}^d)$$

Now we know that in document d, the ith word will be about topic $z_i^d \in \{1, \ldots, K\}$. We have a $K \times M$ matrix of probabilities $\mathbf{W}$ whose row k, $\boldsymbol{w}_k \equiv [w_{k1}, \ldots, w_{KM}]^T$ gives us the probabilities of occurrences of the M words in topic k. So knowing that the topic for word i needs to come from topic z_i^d, we will sample from the multinomial distribution whose parameters are given by row z_i^d of $\mathbf{W}$ to get the word x_i^d (which is a value between 1 and M):

$$x_i^d \sim \text{Mult}_M(\boldsymbol{w}_{z_i^d})$$

This is a multinomial draw, and we define a Dirichlet prior with hyperparameter $\boldsymbol{\beta}$ on these rows of multinomial probabilities:

$$\boldsymbol{w}_k \sim \text{Dirichlet}(\boldsymbol{\beta})$$

This completes the process to generate one word. To generate the N words for the document, we do this N times; namely, for each word, we decide on a topic, then given the topic, we choose a word (inner plate in the figure). When we get to the next document, we sample another topic distribution $\boldsymbol{\pi}$ (outer plate), and then sample N words from that topic distribution.

On all the documents, we always use the same $\mathbf{W}$, and in learning, we are given a large corpus of documents, that is, only x_i^d values are observed. We can then write a posterior distribution as usual and learn $\mathbf{W}$, the word probabilities for topics shared across all documents.

Once $\mathbf{W}$ is learned, each of its rows correspond to one topic. By looking at the words with high probabilities, we can assign some meaning to these

topics. Note, however, that we will always learn some $\mathbf{W}$; whether the rows will be meaningful or not is another matter.

The model we have just discussed is parametric, and its size is fixed; we can make it nonparametric by making K, the number of topics, which is the hidden complexity parameter, increase as necessary and adapt to data using a Dirichlet process. We need to be careful though. Each document contains N words that come from some topics, but we have several documents and they all need to share the same set of topics; that is, we need to tie the Dirichlets that generate the topics. For this, we define a hierarchy; we define a higher Dirichlet process from which we draw the Dirichlets for individual documents. This is a *hierarchical Dirichlet process* (Teh et al. 2006) that allows topics learned for one document be shared by all.

HIERARCHICAL DIRICHLET PROCESS

16.12 Beta Processes and Indian Buffets

Now let us see an application of the Bayesian approach to dimensionality reduction in factor analysis. Remember that given the $N \times d$ matrix of data $\mathbf{X}$, we want to find k features or latent factors, each of which are d-dimensional such that the data can be written as a linear combination of them. That is, we want to find $\mathbf{Z}$ and $\mathbf{A}$ such that

$$\mathbf{X} = \mathbf{Z}\mathbf{A}$$

where $\mathbf{A}$ is the $k \times d$ matrix whose row j is the d-dimensional feature vector (similar to the eigenvector in PCA (section 6.3) and $\mathbf{Z}$ is $N \times k$ matrix whose row t defines instance t as a vector of features.

Let us assume that z_j^t are binary and are drawn from Bernoulli distributions with probability μ_j:

$$z_j^t = \begin{cases} 1 & \text{with probability } \mu_j \\ 0 & \text{with probability } 1 - \mu_j \end{cases} \tag{16.50}$$

So z_j^t indicates the absence/presence of hidden factor j in constructing instance t. If the corresponding factor is present, row j of $\mathbf{A}$ is chosen and the sum of all such rows chosen make up row t of $\mathbf{X}$.

We are being Bayesian so we define priors. We define a Gaussian prior on $\mathbf{A}$ and a beta conjugate prior on μ_j of Bernoulli z_j^t:

$$\mu_j \sim \text{beta}(\alpha, 1) \tag{16.51}$$

where α is the hyperparameter. We can write down the posterior and estimate the matrix $\mathbf{A}$. Looking at the rows of $\mathbf{A}$, we can get an idea about what the hidden factors represent; for example, if k is small (e.g., 2), we can plot and visualize the data.

We assume a certain k; hence this model is parametric. We can make it nonparametric and allow k increase with more data (Griffiths and Ghahramani 2011). This defines a *beta process* and the corresponding metaphor is called the *Indian buffet process*, which defines a generative model that works as follows.

BETA PROCESS

INDIAN BUFFET PROCESS

There is an Indian restaurant with a buffet that contains k dishes and each customer can take a serving of any subset of these dishes. The first customer (instance) enters and takes servings of the first m dishes; we assume m is a random variable generated from a Poisson distribution with parameter α. Then each subsequent customer n can take a serving of any existing dish j with probability n_j/n where n_j is the number of customers before who took a serving of dish j, and once he or she is done sampling the existing dishes, that customer can also ask for Poisson(α/n) additional new dishes, hence growing the model. When applied to the context of latent factor model earlier, this corresponds to a model where the number of factors need not be fixed and instead grows as the complexity inherent in data grows.

16.13 Notes

Bayesian approaches are becoming more popular recently. The use of generative graphical models corresponds quite well to the Bayesian formalism, and we are seeing interesting applications in various domains from natural language processing to computer vision to bioinformatics.

The recent field of Bayesian nonparametrics is also interesting in that adapting model complexity is now a part of training and is not an outer loop of model complexity adjustment; we expect to see more work along this direction in the near future. One example of this is the infinite hidden Markov models (Beal, Ghahramani, and Rasmussen 2002) where the number of hidden states is automatically adjusted with more data.

Due to lack of space and the need to keep the chapter to a reasonable length, the approximation and sampling methods are not discussed in detail in this chapter; see MacKay 2003, Bishop 2006, or Murphy 2012 for

more information about variational methods and Markov chain Monte Carlo sampling.

Bayesian approach is interesting and promising, and has already worked successfully in many cases, but it is far from completely supplanting the nonBayesian, or frequentist, approach. For tractability, generative models may be quite simple—for example, latent Dirichlet analysis loses the ordering or words—or the approximation methods may be hard to derive, and sampling methods slow to converge; hence frequentist shortcuts, (e.g., empirical Bayes), may be preferred in certain cases. Hence, it is best to look for an ideal compromise between the two worlds rather than fully committing to one.

16.14 Exercises

1. For the setting of figure 16.3, observe how the posterior changes as we change N, σ^2, and σ_0^2.
2. Let us denote by x the number of spam emails I receive in a random sample of n. Assume that the prior for q, the proportion of spam emails is uniform in $[0, 1]$. Find the posterior distribution for $p(q|x)$.
3. As above, except that assume that $p(q) \sim \mathcal{N}(\mu_0, \sigma_0^2)$. Also assume n is large so that you can use central limit theorem and approximate binomial by a Gaussian. Derive $p(q|x)$.
4. What is $\text{Var}(r')$ when the maximum likelihood estimator is used? Compare it with equation 16.25.
5. In figure 16.10, how does the fit change when we change s^2?

   SOLUTION: As usual, s is the smoothing parameter and we get smoother fits as we increase s.
6. Propose a filtering algorithm to choose a subset of the training set in Gaussian processes.

   SOLUTION: One nice property of Gaussian processes is that we can calculate the variance at a certain point. For any instance from the training set, we can calculate the leave-one-out estimate there and check whether the actual output is in, for example, the 95 percent prediction interval. If it is, this means that we do not need that instance and it can be left out. Those that cannot be pruned will be just like the support vectors in a kernel machine, namely, those instances that are stored and needed, to bound the total error of the fit.

ACTIVE LEARNING

7. *Active learning* is when the learner is able to generate x itself and ask a su-

pervisor to provide the corresponding r value during learning one by one, instead of passively being given a training set. How can we implement active learning using Gaussian processes? (Hint: Where do we have the largest uncertainty?)

SOLUTION: This is just like the previous exercise, except that we add instead of prune. Using the same logic, we can see that we need instances where the prediction interval is large. Given the variance as a function of $\mathbf{x}$, we search for its local maxima. In the case of a Gaussian kernel, we expect points that are distant from training data to have high variance, but this need not be the case for all kernels. While searching, we need to make sure that we do not go out of the valid input bounds.

8. Let us say we have a set of documents where for each document, we have one copy in English and one in French. How can we extend latent Dirichlet allocation for this case?

16.15 References

Beal, M. J., Z. Ghahramani, and C. E. Rasmussen. 2002. "The Infinite Hidden Markov Model." In *Advances in Neural Information Processing Systems 14*, ed. T. G. Dietterich, S. Becker, and Z. Ghahramani, 577–585. Cambridge, MA: MIT Press.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. New York: Springer.

Blei, D. M. 2012. "Probabilistic Topic Models." *Communications of the ACM* 55 (4): 77–84.

Blei, D. M., A. Y. Ng, and M. I. Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Intelligence* 3:993–1022.

Figueiredo, M. A. T. 2003. "Adaptive Sparseness for Supervised Learning." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25:1150–1159.

Gershman, S. J., and D. M. Blei. 2012. "A Tutorial on Bayesian Nonparametric Models." *Journal of Mathematical Psychology* 56:1–12.

Griffiths, T. L., and Z. Ghahramani. 2011. "The Indian Buffet Process: An Introduction and Review." *Journal of Machine Learning Research* 12:1185–1224.

Hastie, T., R. Tibshirani, and J. Friedman. 2011. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer.

Hoff, P. D. 2009. *A First Course in Bayesian Statistical Methods*. New York: Springer.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, L. K. Saul. 1999. "An Introduction to Variational Methods for Graphical Models." *Machine Learning* 37:183–233.

MacKay, D. J. C. 1998. "Introduction to Gaussian Processes." In *Neural Networks and Machine Learning*, ed. C. M. Bishop, 133–166. Berlin: Springer.

MacKay, D. J. C. 2003. *Information Theory, Inference, and Learning Algorithms*. Cambridge, UK: Cambridge University Press.

Murphy, K. P. 2007. "Conjugate Bayesian Analysis of the Gaussian Distribution." `http://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf`.

Murphy, K. P. 2012. *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press.

Rasmussen, C. E. , and C. K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press.

Teh, Y. W., M. I. Jordan, M. J. Beal, and D. M. Blei. 2006. "Hierarchical Dirichlet Processes." *Journal of Americal Statistical Association* 101: 1566–1581.

Tibshirani, R. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B* 58: 267–288.

Waterhouse, S., D. MacKay, and T. Robinson. 1996. "Bayesian Methods for Mixture of Experts." In *Advances in Neural Information Processing Systems 8*, ed. D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, 351–357. Cambridge, MA: MIT Press.