Notes of PRML

Siyu Wang

August 2018

Chapter 1

Mathematical Foundations

1.1 Probability Theory

sum rule:

$$p(X) = \sum_{Y} p(X, Y)$$

product rule:

$$p(X, Y) = p(Y|X)p(X)$$

Question: what is the most different thing between bayesian and non-bayesian?

1.1.1 probability densities

continuous variables for

$$x = g(y)$$

,

$$p_y(y) = p_x(x)|\frac{dx}{dy}| = p_x(g(y))|g'(y)|$$

Bayes' theorem still applies to probability densities or combination of discrete and continuous variables:

$$p(x) = \int p(x, y)\mathrm{d}y$$

$$p(x, y) = p(y|x)p(x)$$

1.1.2 Expectations and covariances

variance for one variable $f(x)$:

$$\mathrm{var}[f] = E[(f(x) - E[f(x)])^2]$$

covariance for two variables x, y:

$$\mathrm{cov}[x, y] = E_{x,y}[\{x - E[x]\}\{y - E[y]\}]$$

for two vectors of random variables $\mathbf{x}$ and $\mathbf{y}$, the covariance is a matrix

$$\mathrm{cov}[\mathbf{x}, \mathbf{y}] = E_{x,y}[\{\mathbf{x} - E[\mathbf{x}]\}\{\mathbf{y}^T - E[\mathbf{y}^T]\}]$$

and for one vector

$$\mathrm{cov}[\mathbf{x}] = \mathrm{cov}[\mathbf{x}, \mathbf{x}]$$

1.1.3 Bayesian probabilities

essential idea: convert a prior probability into a posterior proboability.

$$p(\mathbf{w}, \mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})} \tag{1.1}$$

note that
$p(\mathbf{w})$: before observing the data, in the form of a prior probability distribution
$\mathcal{D}$: observed data $\{t_1, \ldots, t_n\}$
$p(\mathcal{D})$: seen as a normalization constant.
$p(\mathcal{D}|\mathbf{w})$: likelihood function. a function of the parameter vector $\mathbf{w}$, expresses how probable the observed data set is for different settings of the parameter vector $\mathbf{w}$ key idea can be sumed as:

$$\text{posterior} \propto \text{likelihood} \times \text{prior} \tag{1.2}$$

Bayesian and frequentist paradigm for frequentist, $\mathbf{w}$ is considered to be fixed parameter and its value is determined by some form of 'estimator'.
one common frequentist estimator is maximum likelihood, maximize likelihood function $p(\mathcal{D}|\mathbf{w})$, in machine learning, negative log the likelihood function is called an error function.
bootstrap technique.
for Bayesian, there is only a single data set $\mathcal{D}$ and the uncertainty in the parameters is expressed through a probability distribution over $\mathbf{w}$
inclusion of prior knowledge
how to select prior distribution is a problem for Bayesian method.
Reducing the dependence on the prior. noninformative priors

1.1.4 The Gaussian distribution

For one variable:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\{-\frac{1}{2\sigma^2}(x-\mu)^2\} \tag{1.3}$$

for D-dimensional vector:

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\} \tag{1.4}$$

Problem : observations for N times $\mathbf{x} = (x_1, \ldots, x_N)^T$, we want to get parameters for the Gaussian distribution. Then the likelihood function$p(\mathcal{D}|\mathbf{w})$ is just $p(\mathbf{x}|\mu, \sigma^2) = \Pi_{n=1}^{N}\mathcal{N}(x_n|\mu, \sigma^2)$.
so using ML criterion, we will want to maximize $\ln p(\mathbf{x}|\mu, \sigma^2)$ and we can get

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n, \text{samplemean}$$

,

$$\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{ML})^2, \text{samplevariance}$$

but we can infer that:

$$E[\mu_{ML}] = \mu$$

$$E[\sigma^2_{ML}] = (\frac{N-1}{N})\sigma^2$$

rather than exactly σ^2 this can be inferred from inspiration in figure1.3.3: but there seems to be more convenient way using chi-sqaure distribution

1.1.5 Curve fitting re-visited

This time, we model curve fitting problem by : given the value of x, the corresponding value of t has a Gaussian distribution with a mean equal to the value $y(x, \mathbf{w})$ (polynomial curve).

$$p(t|x, \mathbf{w}, \beta) = \mathcal{N}(t|y(x, \mathbf{w}), \beta^{-1}) \tag{1.5}$$

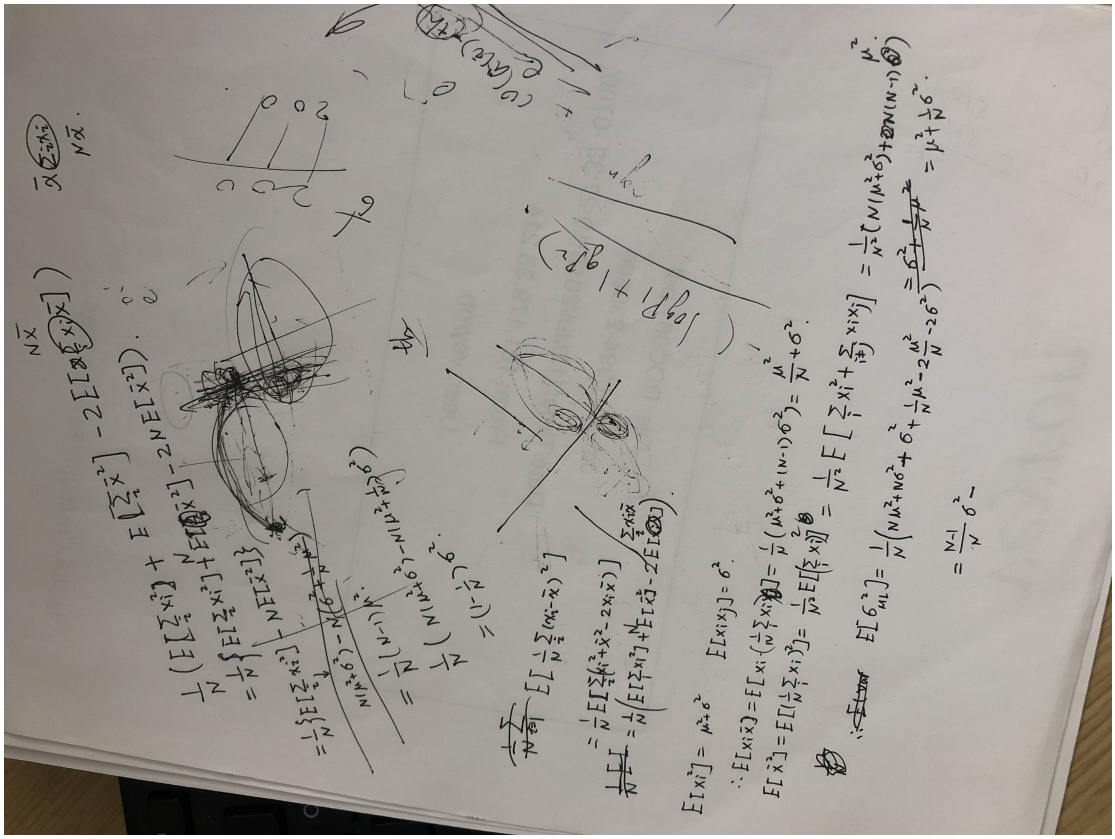


Figure 1.1: Inference

$$p(\mathbf{t}|\mathbf{x}, \mathrm{w}, \beta) = \Pi_{n=1}^{N} \mathcal{N}(t_n|y(x_n, \mathrm{w}), \beta^{-1}) \tag{1.6}$$

and log likelihood function can be derived from above. and **Maximizing likelihood is equivalent to minimizing the sum-of-squares error function defined earlier !!!**
if we use Bayesian method and assume prior distribution over $\mathbf{w}$:

$$p(\mathrm{w}|\alpha) = \mathcal{N}(\mathrm{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})) \tag{1.7}$$

α is called hyperparameters. Using Bayesian rules to get

$$p(\mathrm{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathrm{w}, \beta)p(\mathrm{w}|\alpha) \tag{1.8}$$

Using maximizing the posterior distribution, MAP and we find the maximum of the posterior is given by the minimum of:

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathrm{w}) - t_n^2 +\}\frac{\alpha}{2}\mathrm{w}^T\mathrm{w} \tag{1.9}$$

so this is similar to regularize sum-od-squares error function with $\lambda = \frac{\alpha}{\beta}$

1.1.6 Decision Theory

minimizing the misclassification rate $p(\text{mistake})$ and $p(\text{correct})$
minimizing the expected loss
can assign different loss efficient to different error

$$E[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathrm{x}, \mathcal{C}_k)\mathrm{d}\mathrm{x} \tag{1.10}$$

$p(\mathrm{x}, \mathcal{C}_k) = p(\mathcal{C}_k|\mathrm{x})p(\mathrm{x})$ to eliminate the common factor $p(\mathrm{x})$
inference and decision
Three distinct approaches:

1. First solve $p(\mathrm{x}|\mathcal{C}_k)$ as well as $p(\mathcal{C}_k$ for each class individually, then figure out $p(\mathcal{C}_k|\mathrm{x}) = \frac{p(\mathrm{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathrm{x})}$ and $p(\mathrm{x})$ can be gotten from $p(\mathrm{x}) = \sum_k p(\mathrm{x}|\mathcal{C}_k)p(\mathcal{C}_k)$. This is equivalent to model the joint distribution $p(\mathrm{x}, \mathcal{C}_k)$
2. Determine posterior class probabilities $p(\mathcal{C}_k|\mathrm{x})$ and then use decision theory. discriminative model.
3. Find $f(\mathrm{x})$ which maps x directly to a class label.

The first approach is too time-consuming and demanding, while the last one is simplest but not so robust as approach.2 because we can see a lot of information from the posterior probabilities:

- minimizing risk, by adjusting L_{kj} to get different loss function and minimize risk
- reject option
- **compensating for class priors** this is not thought of by myself but very important. sometimes we have to make training set more suitable for learning features, but this operation on training set will lead to a modification on prior probabilities so we must revise this part by divide by the class fraction in the data set and then multiply by the class fraction in the population of which we want to apply this model.
- combining models.

Loss function for regression

$$E[L] = \iint L(t, y(\mathrm{x}))p(\mathrm{x}, t)\mathrm{d}\mathrm{x}$$

If we define

$$L(t, y(\mathrm{x})) = \{y(\mathrm{x}) - t\}^2 \tag{1.11}$$

we can get $y(\mathrm{x}) = E_t[t|\mathrm{x}]$ to minimize $E[L]$.
Similarly, we also have three approaches using joint probability, posterior probability and a direct regression function $f(\mathrm{x})$
sometimes we define L by Minkowski loss as $L_q = |y(\mathrm{x}) - t|^q$

1.1.7 Information Theory

Look for a quantity $h(x)$ than is a monotonic function of the probability $p(x)$ and that expresses the information content. and should satisfy

$$h(x, y) = h(x) + h(y)$$

. so we define

$$h(x) = -\log_2 p(x)(bits)$$

or

$$h(x) = -\ln p(x), nats$$

ENTROPY:

$$H[x] = E[h(x)] = -\sum_x p(x)\log_2 p(x) \tag{1.12}$$

relative to statistical physics.
To define entropy for continuous variable:
differential entropy: $H[\mathrm{x}] = -\int p(\mathrm{x})\ln p(\mathrm{x})$
maximizing entropy by using Lagrange multipliers

- for discrete, uniform distribution
- for continuous, Gaussion distribution $H[x] = \frac{1}{2}[1 + \ln(2\pi\sigma^2)]$

conditional entropy:

$$H[y|x] = \iint p(y, x)\ln p(y|x)\mathrm{d}y\mathrm{d}x \tag{1.13}$$

so $H[x, y] = H[y|x] + H[x]$.

Relative entropy and mutual information

KL divergence (relative entropy)

$$\mathrm{KL}(p||q) = -\int p(x)\ln q(x)\mathrm{d}x - (-\int p(x)\ln p(x)\mathrm{d}x) = -\int p(x)\ln\frac{q(x)}{p(x)} \tag{1.14}$$

this is not symmetrical!
when we use convex function and Jensen's unequality,

$$f(\int xp(x)\mathrm{d}x) \leq \int f(x)p(x)\mathrm{d}x \tag{1.15}$$

also as $f(E[x]) \leq E[f(x)]$ we can get

$$\mathrm{KL}(p||q) = -\int p(x)\ln\{\frac{q(x)}{p(x)}\}\mathrm{d}x \geq -\ln\int q(x)\mathrm{d}x = 0 \tag{1.16}$$

BUT I think here, in order to get unequality for KL-divergence , Jensen's unequality should be generalized to $f(E[k(x)]) \leq E[f(k(x))]$, to be more precise the most important thing is , KL divergence can be used as a measure of the dissimilarity of two distributions $p(x)$ and $q(x)$

to approximate an unknown distribution p(x)

we have observed a finite set of training points x_n for $n = 1, 2, \ldots, N$ drawn from $p(\mathrm{x})$ here we must pay attention that we are not getting points from the distribution curve, we are getting points distributed as $p(\mathrm{x})$ says. So

$$\mathrm{KL}(p||q) = -\mathrm{E}[\ln\frac{q(\mathrm{x})}{p(\mathrm{x})}] \simeq \sum_{n=1}^{N}\{-\ln q(\mathrm{x}_n|\theta) + \ln p(\mathrm{x}_n)\} \tag{1.17}$$

consider joint distrbiution between two sets of variables x and y

$p(x, y) = p(x)p(y)$ if x and y is independent. When they are dependent, we want to use KL divergence between joint distribution and the product of marginals to see how close they are to independent.

$$I[x, y] = \mathrm{KL}(p(x, y)||p(x)p(y)) = -\int\int p(x, y)\ln(\frac{p(x)p(y)}{p(x, y)}) \tag{1.18}$$

this is mutual information between x and y and

$$I[x, y] = H[x] - H[x|y] \tag{1.19}$$

The mutual information represents the reduction in uncertainty about x as a consequence of the new observation y.

1.1.8 Some other distributions

Poisson distribution

Laplace dsitribution

1.1.9 Conclusion

The most important thought in this chapter is difference between Bayesian and non Bayesian,, e.g. ML and MAP. Sometimes ML is equivalent to minimizing error function and MAP can be equivalent to them, too.
The most important thing stepping from non Bayesian to Bayesian, is the inclusion of prior probability and we use

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

To see this problem from angel of information theory, we can find KL divergence between $p(x, y)$ and $p(x)p(y)$ will be mutual information between x and y. And this represent the reduction in uncertainty about x as a consequence of the new observation y.

1.2 Probability Distributions

In the frequentist treatment , we choose specific values for the parameters by optimizing some criterion, such as the likelihood function. But, in a Bayesian treatment we introduce prior distribution over the parameters and then use Bayes' theorem to compute the corresponding posterior distribution given the observed data.
limitation: assumes a specific functional form of the distribution.

1.2.1 Binary Variables

$$p(x=1|\mu)=\mu \tag{1.20}$$

$$\text{Bern}(x|\mu)=\mu^x(1-\mu)^{1-x} \tag{1.21}$$

$$\text{E}[x]=\mu, \text{var}[x]=\mu(1-\mu) \tag{1.22}$$

data set $\mathcal{D}=x_1,\dots,x_N$ of observed values of x, independently.
then we can construct the likelihood function

$$p(\mathcal{D}|\mu)=\Pi_{n=1}^N p(x_n|\mu)=\Pi_{n=1}^N \mu^{x_n}(1-\mu)^{1-x_n} \tag{1.23}$$

log likelihood function $\ln p(\mathcal{D}|\mu)$
maximize likelihood can lead to $\mu_{ML}=\frac{1}{N}\sum_{n=1}^N x_n$, and $\sum_{n=1}^N x_n$ is called sufficient statistic
Denote the number of observation of $x=1$ within this data set by m, then $\mu_{ML}=\frac{m}{N}N$.
But,we can get extreme result using maximum likelihood criterion! **binomial distribution**

$$\text{Bin}(m|N,\mu)=C_N^m\mu^m(1-\mu)^{N-m}$$

$$\text{E}[m]=N\mu, \text{var}[m]=N\mu(1-\mu)$$

The beta distribution

we choose an appropriate prior distribution so that the posterior distribution will have the same functional form as the prior. so beta distribution comes.

$$\text{Beta}(\mu|a,b)=\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1}\Gamma(x)=\int_0^\infty u^{x-1}e^{-u}\text{d}u \tag{1.24}$$

The distribution is normalized.

$$\text{E}[\mu]=\frac{a}{a+b}, \text{var}[\mu]=\frac{ab}{(a+b)^2(a+b+1)}$$

a and b are hyperparameters controlling the distribution of the parameter μ.
so we can get posterior distribution:

$$p(\mu|m,l,a,b)\propto\mu^{m+a-1}(1-\mu)^{l+b-1}$$

and after normalization we get:

$$p(\mu|m,l,a,b)=\frac{\Gamma(m+a+l+b)}{\Gamma(m+a)\Gamma(l+b)}\mu^{m+a-1}(1-\mu)^{l+b-1}$$

The posterior can act as the prior if we subsequently observe additional data. sequential approach: Sequential methods make use of observations one at a time, or in small batches, and then discard them before the next observations are used.
and

$$p(x=1|\mathcal{D})=\int_0^1 p(x=1|\mu)p(\mu|\mathcal{D})\text{d}\mu=\int_0^1 \mu p(\mu|\mathcal{D})\text{d}\mu=\text{E}[\mu|\mathcal{D}] \tag{1.25}$$

$=\frac{m+a}{m+a+l+b}$
so when $m,l\to\infty$, it reduce to $\frac{m}{m+l}$
when the number of observations increases, the posterior distribution becomes more sharply peaked, which can be seen from the variance. **and this is a general property of Bayesian learning!**
process of inferring can be seen at **page74** in the book.

1.2.2 Multinomial Variables

an extension to binary variables
$\mathrm{x} = (x_1, \ldots, x_K)^T$ one-hot
denote, the probability of $x_k = 1$ by the parameter μ_k and $\sum_{k=1}^K \mu_k = 1$
so, $p(\mathrm{x}|\mu) = \Pi_{k=1}^K \mu_k^{x_k}$
data set $\mathcal{D}$ of N independent observations $\mathrm{x}_1, \ldots, \mathrm{x}_N$
so likelihood function is:

$$p(\mathcal{D}|\mu) = \Pi_{n=1}^N \Pi_{k=1}^K \mu_k^{x_{nk}} = \Pi_{k=1}^K \mu_k^{\sum_n x_{nk}} = \Pi_{k=1}^K \mu_k^{m_k} \tag{1.26}$$

where $m_k = \sum_n x_{nk}$ represents the number of observations of $x_k = 1$, sufficient statistics for this distribution
to do ML, maximize $\sum_{k=1}^K m_k \ln \mu_k + \lambda(\sum_{k=1}^K \mu_k - 1)$ and we get $\mu_k^{ML} = \frac{m_k}{N}$

$$\mathrm{Mult}(m_1, \ldots, m_K|\mu, N) = \binom{N}{m_1 m_2 \ldots m_K} \Pi_{k=1}^K \mu_k^{m_k} \tag{1.27}$$

The Dirichlet distribution

a family of prior distributions for the parameters $\{\mu_k\}$ of the multinomial distribution.

$$p(\mu|\alpha) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\ldots\Gamma(\alpha_K)} \Pi_{k=1}^K \mu_k^{\alpha_k - 1} \tag{1.28}$$

where $\alpha_0 = \sum_{k=1}^K \alpha_k$. then posterior probability will be $p(\mu|\mathcal{D}, \alpha) = \mathrm{Dir}(\mu|\alpha + \mathrm{m})$

1.3 The Gaussian Distribution

the sum of multiple random variables **central limit theorem**

$$\mathcal{N}(\mathrm{x}|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(\mathrm{x} - \mu)^T \Sigma^{-1} (\mathrm{x} - \mu)\} \tag{1.29}$$

dependence on x is through a quadratic form:

$$\Delta^2 = (\mathrm{x} - \mu)^T \Sigma^{-1} (\mathrm{x} - \mu)$$

in the exponent.
We can take Σ to be symmetric because ant antisymmetric component would disappear from the exponent. consider eigenvalues (real number because of Σ's reality and symmetry) of Σ

$$\Sigma \mathrm{u}_i = \lambda_i \mathrm{u}_i, i = 1, 2, \ldots, D \tag{1.30}$$

$\mathrm{u}_i^T \mathrm{u}_j = I_{ij}$
then $\Sigma = \sum_{i=1}^D \lambda_i \mathrm{u}_i \mathrm{u}_i^T$ and $\Sigma^{-1} = \sum_{i=1}^D \frac{1}{\lambda_i} \mathrm{u}_i \mathrm{u}_i^T$
and define $y_i = \mathrm{u}_i^T (\mathrm{x} - \mu)$,
then $\Sigma^{-1} = \sum_{i=1}^D \frac{y_i^2}{\lambda_i}$
$\mathrm{y} = \mathrm{U}(\mathrm{x} - \mu)$ and $|\mathrm{J}| = 1$ so $p(\mathrm{y}) = p(\mathrm{x})|\mathrm{J}| = \Pi_{j=1}^D \frac{1}{(2\pi\lambda_j)^{1/2}} \exp\{-\frac{y_j^2}{2\lambda_j}\}$
mean and covariance $\mathrm{E}[\mathrm{x}] = \mu, \mathrm{cov}[\mathrm{x}] = \Sigma$

1.3.1 Conditional Gaussian distributions

$$\mathrm{x} = (\mathrm{x}_a^T, \mathrm{x}_b^T)^T, \mu = (\mu_a^T, \mu_b^T)^T, \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix} \tag{1.31}$$

we can get

$$p(\mathrm{x}_a|\mathrm{x}_b) = \mathcal{N}(\mathrm{x}_a|\mathrm{x}_b, \mu_{a|b}, \Sigma_{a|b}) \tag{1.32}$$

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\mathrm{x}_b - \mu_b) \tag{1.33}$$

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \tag{1.34}$$

$Cov[z] = R^{-1} = \begin{pmatrix} \Lambda^{-1} & \Lambda^{-1}A^T \\ A\Lambda^{-1} & L^{-1}+A\Lambda^{-1}A^T \end{pmatrix}$.

$E[z] = R^{-1}\begin{pmatrix} \Lambda\mu - A^TLb \\ Lb \end{pmatrix}$.

So we can use formula in marginal distribution and get p(y) and p(x|y).

$E[y] = A\mu+b \quad cov[y] = L^{-1}+A\Lambda^{-1}A^T.$

$E[x|y] = (\Lambda+A^TLA)^{-1}\{A^TL(y-b)+\Lambda\mu\} \quad cov[x|y] = (\Lambda+A^TLA)^{-1}$

∞

1.3.2 Marginal Gaussian distribution

$p(\mathrm{x}_a) = \int p(\mathrm{x}_a, \mathrm{x}_b)\mathrm{d}\mathrm{x}_b$ is still Gaussian and $p(\mathrm{x}_a) = \mathcal{N}(\mathrm{x}_a|\mu_a, \Sigma_{aa})$

1.3.3 Bayes' theorem for Gaussian variable

linear Gaussian model e.g. Gaussian conditional distribution $p(y|x)$ in which $p(y|x)$ has a mean that is a linear function of x, and a covariance which is dependent of x.
Our goal: given $p(\mathrm{x}) = \mathcal{N}(\mathrm{x}|\mu, \Lambda^{-1})$, $p(\mathrm{y}|\mathrm{x}) = \mathcal{N}(\mathrm{y}|\mathrm{A}\mathrm{x}+\mathrm{b}, \mathrm{L}^{-1})$, we want to find $p(\mathrm{y})$ and $p(\mathrm{x}|\mathrm{y})$.
$\mathrm{x} \in R^M$ and $\mathrm{y} \in R^D, \mathrm{A} \in R^{D\times M}$

$$\mathrm{z} = \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \end{pmatrix}$$

$$\ln p(\mathrm{z}) = \ln p(\mathrm{x}) + \ln p(\mathrm{y}|\mathrm{x})$$

and finally z has precision matrix as

$$\mathrm{R} = \begin{pmatrix} \Lambda + \mathrm{A}^T\mathrm{L}\mathrm{A} & -\mathrm{A}^T\mathrm{L} \\ -\mathrm{L}\mathrm{A} & \mathrm{L} \end{pmatrix}$$

and

$$\mathrm{cov}[\mathrm{z}] = \mathrm{R}^{-1}$$

1.3.4 Maximum likelihood for the Gaussian

log likelihood function:

$$\ln p(\mathrm{X}|\mu, \Sigma) = -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}(\mathrm{x}_n - \mu)^T\Sigma^{-1}(\mathrm{x}_n - \mu) \tag{1.35}$$

we should pay attention that the thing on the exponent is a number and

$$(\mathrm{x}_n - \mu)^T\Sigma^{-1}(\mathrm{x}_n - \mu) = \mathrm{x}_n^T\Sigma^{-1}\mathrm{x}_n + \mu^T\Sigma^{-1}\mu - 2\mathrm{x}_n^T\Sigma\mu \tag{1.36}$$

sufficient statistics for Gaussian distribution $\sum_{n=1}^{N}\mathrm{x}_n, \sum_{n=1}^{N}\mathrm{x}_n\mathrm{x}_n^T$ and $\frac{\partial}{\partial\mu}\ln p(\mathrm{X}|\mu, \Sigma), \therefore \mu_{ML} = \frac{1}{N}\sum_{n=1}^{N}\mathrm{x}_n$
$\Sigma_{ML} = \frac{1}{N}\sum_{n=1}^{N}(\mathrm{x}_n - \mu_{ML})(\mathrm{x}_n - \mu_{ML})^T$
and $\mathrm{E}[\mu_{ML}] = \mu, \mathrm{E}[\Sigma_{ML}] = \frac{N-1}{N}\Sigma$

1.3.5 Sequential estimation

$\mu_{ML}^{(N)} = \mu_{ML}^{(N-1)} + \frac{1}{N}(\mathrm{x}_N - \mu_{ML}^{(N-1)})$ we will not always be able to derive a sequential algorithm by this route, and so we seek a more general formulation of sequential learning, which leads us to the Robbins-Monro algorithm.

$$f(\theta) \equiv E[z|\theta] = \int zp(z|\theta)\mathrm{d}z \tag{1.37}$$

regression functions goal: to find root for $f(\theta) = 0$

$$\theta^{(N)} = \theta^{(N-1)} + \alpha_{N-1}z(\theta^{(N-1)})$$

$z(\theta^{(N)})$ is an observe value of z when θ takes the value $\theta^{(N)}$ and α_N should satisfy some conditions.
use Robbins Monro algorithm to solve maximizing log likelihood function problem
We want to get

$$\frac{\partial}{\partial\theta}\{\frac{1}{N}\sum_{n=1}^{N}\ln p(\mathrm{x}_n|\theta)\} = 0$$

exchanging summation and derivative, taking the limit $N \to \infty$ and we have

$$\mathrm{E}_x[\frac{\partial}{\partial\theta}\ln p(x|\theta)]$$

so

$$\theta^{(N)} = \theta^{(N-1)} + \alpha_{N-1}\frac{\partial}{\partial\theta^{(N-1)}}\ln p(x_N|\theta^{(N-1)})$$

1.3.6 Baysesian inference for the Gaussian

- suppose σ^2 is known, to get μ

  $$p(X|\mu) = \Pi_{n=1}^{N}p(x_n|\mu) = \frac{1}{(2\pi\sigma^2)^{N/2}}\exp\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\}$$

  we choose prior to be Gaussian, too

  $$p(\mu) = \mathcal{N}(\mu|\mu_0, \sigma_0^2)$$

  so

  $$p(\mu|X) = \mathcal{N}(\mu|\mu_N, \sigma_N^2)$$

  $$\mu_N = \frac{\sigma^2}{N\sigma_0^2+\sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2+\sigma^2}\mu_{ML}$$

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

  when N is infinite large, $\mu_N = \mu_{ML}$, only depend on observations, and σ_N^2 goes to 0 and the posterior distribution becomes infinitely peaked.
  if we take σ_0^2 to be infinite, so the prior is flat over all real numbers, so prior will be no use.
  sequential angle:

  $$p(\mu|D) \propto [p(\mu)\Pi_{n=1}^{N-1}p(\mathrm{x}_n|\mu)]p(\mathrm{x}_N|\mu)$$

  we can view the postrior distribution after N-1 observations as a prior distribution of the Nth observation.

- suppose μ is known, to get σ^2, see inference in figure1.3.6
- suppose both are not known. see inference in figure1.3.6

$\lambda \equiv 1/\sigma^2$ precision

$p(x|\lambda) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu,\lambda^{-1}) \propto \lambda^{N/2} \exp\{-\frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2\}$.

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

choose prior as Gamma: $\mathrm{Gam}(\lambda|a,b) = \frac{1}{\Gamma(a)} b^a \lambda^{a-1}\exp(-b\lambda)$.

$E[\lambda] = \frac{a}{b}$ $\mathrm{var}[\lambda] = a/b^2$.

so, if we multiply this by the likelihood function, then we get posterior distribution:

$p(\lambda|x) \propto \lambda^{a_0-1}\lambda^{N/2}\exp\{-b_0\lambda - \frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2\}$ so. $a_N = \frac{N}{2}+a_0$. $b_N = b_0+\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2$

$= b_0 + \frac{N}{2}\sigma_{ML}^2$.

$p(x|\mu,\lambda) = \prod_{n=1}^{N}(\frac{\lambda}{2\pi})^{1/2}\exp\{-\frac{\lambda}{2}(x_n-\mu)^2\} \propto [\lambda^{\frac{1}{2}}\exp(-\frac{\lambda\mu^2}{2})^{\beta}]^N \exp\{\lambda\mu\sum_{n=1}^{N}x_n - \frac{\lambda}{2}\sum_{n=1}^{N}x_n^2\}$.

prior: $p(\lambda,\mu) \propto [\lambda^{1/2}\exp(-\frac{\lambda\mu^2}{2})^{\beta}]\exp\{c\lambda\mu - d\lambda\}$ c, d, β.

in fact, $p(\lambda,\mu) = \mathcal{N}(\mu|\mu_0,(\beta\lambda)^{-1})\,\mathrm{Gam}(\lambda|a,b)$. $\mu_0 = c/\beta$. $a = 1+\beta/2$ $b = d-c^2/2\beta$.

normal-gamma or Gaussian-gamma distribution.

1.3.7 Student's t-distribution

precision λ degrees of freedom ν

adding up an infinite number of Gaussian distribution having the same mean but different presicions, we can get Student's t-distribution.

$$\mathrm{St}[x|\mu,\lambda,\nu] = \frac{\Gamma(\nu/2+1/2)}{\Gamma(\nu/2)}(\frac{\lambda}{\pi\nu})^{1/2}[1+\frac{\lambda(x-\mu)^2}{\nu}]^{-\nu/2-1/2}$$

Data set: $(x_1, x_2, ..., x_N)^T, (t_x, t_2, ..., t_N)^T$.

there is noise on the observation ! fit the data using a polynomial function:

$$y(x,\mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j \tag{1.38}$$

this is a **linear** function of coefficients $\mathbf{w}$!! so this is linear. goal is to minimize:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 \tag{1.39}$$

this is a quadratic function of w so its derivatives with respect to coefficients will be linear so the minimization of the error function has a unique solution.

1.3.8 Mixtures of Gaussians

$$p(\mathrm{x}) = \sum_{k=1}^{K}\pi_k\mathcal{N}(\mathrm{x}|\mu_k,\Sigma_k)$$

$$\sum_{k=1}^{K}\pi_k = 1$$

likelihood function:

$$\ln p(\mathrm{X}|\pi,\Sigma,\mu) = \sum_{n=1}^{N}\ln\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathrm{x}_n|\mu_k,\Sigma_k)\}$$

this is more complex than a single Guassian because the presence of summation over k inside the logarithm. we can use EM algorithm later .

1.4 The Exponential Family

definition:

$$p(\mathrm{x}|\eta) = h(\mathrm{x})g(\eta)\exp(\eta^T \mathrm{u}(\mathrm{x})) \tag{1.40}$$

$g(\eta)$ is to normalize the function $g(\eta)\int h(\mathrm{x})\exp(\eta^T \mathrm{u}(\mathrm{x}))\mathrm{dx} = 1$
Bernoulli

$$p(x|\mu) = \mu^x(1-\mu)^{1-x} = \exp\{x\ln\mu + (1-x)\ln(1-\mu)\} = (1-\mu)\exp\{\ln(\frac{\mu}{1-\mu})x\} \tag{1.41}$$

so we define $\eta = \ln\frac{\mu}{1-\mu}$ so $\mu = \sigma(\eta)\frac{1}{1+\exp(-\eta)}$ logistic sigmoid function $u(x) = x, h(x) = 1, g(\eta) = \sigma(-\eta)$

1.4.1 Maximum likelihood and sufficient statistics

$$\nabla g(\eta)\int h(\mathrm{x})\exp\{\eta^T \mathrm{u}(\mathrm{x})\}\mathrm{dx} + g(\eta)\int h(\mathrm{x})\exp\{\eta^T \mathrm{u}(\mathrm{x})\}\mathrm{u}(\mathrm{x})\mathrm{dx} = 0 \tag{1.42}$$

so $-\nabla\ln g(\eta) = \mathrm{E}[\mathrm{u}(\mathrm{x})]$
so consider

$$p(X|\eta) = (\Pi_{n=1}^N h(\mathrm{x}_n))g(\eta)^N \exp\{\eta^T \sum_{n=1}^N \mathrm{u}(\mathrm{x}_n)\}$$

set the gradient to zero and we get

$$-\nabla\ln g(\eta_{ML}) = \frac{1}{N}\sum_{n=1}^N \mathrm{u}(\mathrm{x}_n) \tag{1.43}$$

$\sum_n \mathrm{u}(\mathrm{x}_n)$ is the sufficient statistics of the distribution.

1.4.2 Conjugate priors

for any member of the exponential family, there exist a conjugate prior:

$$p(\eta|\chi,\nu) = f(\chi,\nu)g(\eta)^\nu \exp\{\nu\eta^T\chi\}$$

so the posterior distribution can be figured out as the same form:

$$p(\eta|X,\chi,\nu) \propto g(\eta)^{\nu+N}\exp\{\eta^T(\sum_{n=1}^N \mathrm{u}(\mathrm{x}_n) + \nu\chi)\}$$

1.4.3 Noninformative priors

to have little influence on the posterior distribution, we seek a form of prior distribution called a noninformative prior.

1.5 Laplace Approximation

In this chapter, we mainly want to solve the problem that when the posterior distribution is no longer Gaussian, how can we integrate exactly over the parameter vector **w**. So Laplace approximation can help us to transform a non-Gaussian distribution into a Gaussian distribution. Consider first the case of a single continuous variable z, and suppose

$$p(z) = \frac{1}{Z}f(z) \tag{1.44}$$

where $Z = \int f(z)\mathrm{d}z$ is the normalization coefficient and is unknown. Our goal is to find a Gaussian approximation $q(z)$ which is centred on a mode of the distribution $p(z)$. To find a mode of $p(z)$, we set $p'(z)|_{z=z_0} = 0$, or equivalently

$$\frac{\mathrm{d}f(z)}{\mathrm{d}z}|_{z=z_0} = 0. \tag{1.45}$$

Then due to the property of the log of Gaussian distribution, we consider a Taylor expansion of $\ln f(z)$ centred on the mode z_0 so that

$$\ln f(z) \simeq \ln f(z_0) - \frac{1}{2}A(z - z_0)^2 \tag{1.46}$$

where

$$A = -\frac{\mathrm{d}^2}{\mathrm{d}z^2}\ln f(z)|_{z=z_0}. \tag{1.47}$$

Then

$$f(z) \simeq \hat{f}(z) = f(z_0)\exp\{-\frac{A}{2}(z - z_0)^2\}. \tag{1.48}$$

Making normalization over $\hat{f}(z)$, we have

$$q(z) = \frac{A}{2\pi}^{1/2}\exp\{-\frac{A}{2}(z - z_0)^2\}. \tag{1.49}$$

Note that the Gaussian approximation is only be well defined when $A > 0$, in other words, the stationary point z_0 must be a local maximum.

Then we want to extend to M-dimensional space $\mathbf{z}$ and $p(\mathbf{z}) = \frac{1}{Z}f(\mathbf{z})$. At a stationary $\mathbf{z}_0$ the gradient $\nabla f(\mathbf{z}) = 0$. Similarly

$$\ln f(\mathbf{z}) \simeq \ln f(\mathbf{z}_0) - \frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z} - \mathbf{z}_0) \tag{1.50}$$

where the $M \times M$ Hessian matrix $\mathbf{A}$ is defined by

$$\mathbf{A} = -\nabla\nabla\ln f(\mathbf{z})|_{\mathbf{z}=\mathbf{z}_0}. \tag{1.51}$$

Taking the exponential of the both side, we obtain

$$f(\mathbf{z}) \simeq f(\mathbf{z}_0)\exp\{-\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z} - \mathbf{z}_0)\}. \tag{1.52}$$

Making normalization, we have

$$q(\mathbf{z}) = \frac{|\mathbf{A}|^{1/2}}{(2\pi)^{M/2}}\exp\{-\frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{z} - \mathbf{z}_0)\} = \mathcal{N}(\mathbf{z}|\mathbf{z}_0, \mathbf{A}^{-1}). \tag{1.53}$$

This Gaussian distribution is only well defined when its *precision matrix*, given by $\mathbf{A}$, is positive definite.

Note that the normalization constant Z of the true distribution does not need to be known in order to apply the Laplace method. As a result of the central limit theorem, the posterior distribution for a model is expected to become increasingly better when the number of data points is increased, so we would expect the Laplace approximation to be most useful when the number of data points is relatively large.

Model comparison and BIC

This section illustrates *Occam factor* and *BIC* in view of the Laplace approximation, more details can be found in section 4.4.1 of *PRML*.

1.6 Jacobian Matrix and Hessian Matrix

1.7 FOR NEW

Chapter 2

Models

2.1 Polynomial Curve Fitting

$\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ and $\mathbf{t} = (t_1, t_2, \ldots, t_N)^T$ There is noise on the observation !
Fit the data using a polynomial function:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

This is a **linear** function of coefficients $\mathbf{w}$!! So this is linear.
Goal is to minimize:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$

This is a quadratic function of $\mathbf{w}$ so its derivatives with respect to coefficients will be linear so the minimization of the error function has a unique solution.

2.1.1 the influence of order M

define error by root-mean-square:

$$E_{RMS} = \sqrt{2E(\mathbf{w}^*)/N} : \tag{2.1}$$

too small: not enough.
too large: **overfitting** and large coefficients, very sensitive to **noise**! low training error while large testing error.

2.1.2 the influence of size of training set

increasing the size of the data set reduces the over-fitting priblem.
comment: this is obvious because the when the size of data set gets larger, the order will become smaller relatively. the requirement for over-fitting is larger order. but when we choose the complexity of our model, we can not depend on the size of data set, we must choose more robust model for **specific problem**.

2.1.3 modifying error function to solve overfitting

$$\tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{M} \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2} ||\mathbf{w}||^2 \tag{2.2}$$

regularizier

$$||\mathbf{w}||^2 = w_0^2 + \cdots + w_M^2$$

somtimes w_0 is omitted to exclude the effect of origin for the tarfet variable !! in nerual networks, this approach is known as *weight decay* To optimize M and λ, we sometimes divide dataset into *training* and *validation* but this is a waste, so we need more sophisticated approaches.

2.2 Linear Models for Regression

2.2.1 Linear Basis Function Models

input $\mathrm{x} \in R^D$

$$y(\mathrm{x}, \mathrm{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D \tag{2.3}$$

extend to nonlinear function of x and $y(\mathrm{x}, \mathrm{w}) = \mathrm{w}^T \phi(\mathrm{x})$, there are M basis functions $\phi_j(\mathrm{x}), j = 1, 2 \ldots, M$ and $\phi_0(\mathrm{x}) = 1$
so $\mathrm{w} \in R^M$ **normal basis functions**

- polynomial $\phi_j(x) = x$
- Gaussian $\phi_j(x) = \exp\{-\frac{(x-\mu_j)^2}{2s^2}\}$
- sigmoidal $\phi_j(x) = \sigma(\frac{x-\mu_j}{s}), \sigma(a) = \frac{1}{1+\exp(-a)}$ sigmoid is equivalent to 'tanh' function.

Maximum likelihood and least squares

$$t = y(\mathrm{x}, \mathrm{w}) + \epsilon, \epsilon \sim \mathcal{N}(0, \beta^{-1}), \beta = \sigma^{-2}$$

so

$$p(t|\mathrm{x}, \mathrm{w}, \beta) = \mathcal{N}(t|y(\mathrm{x}, \mathrm{w}), \beta^{-1}) = \frac{1}{\sqrt{2\pi/\beta}} \exp\{-\frac{\beta(t - y(\mathrm{x}, \mathrm{w}))^2}{2}\}$$

$$\mathrm{E}[t|\mathrm{x}] = \int t p(t|\mathrm{x}) \mathrm{d}t = y(\mathrm{x}, \mathrm{w})$$

for N observations,

$$p(\mathrm{t}|\mathrm{X}, \mathrm{w}, \beta) = \Pi_{n=1}^N \mathcal{N}(t_n|\mathrm{w}^T \phi(\mathrm{x}_n), \beta^{-1})$$

we can ignore X in he condition and get likelihood function:

$$\ln p(\mathrm{t}|\mathrm{w}, \beta) = \sum_{n=1}^N \ln \mathcal{N}(t_n|\mathrm{w}^T \phi(\mathrm{x}_n), \beta^{-1}) = \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathrm{w})$$

in which,

$$E_D(\mathrm{w}) = \frac{1}{2} \sum_{n=1}^N \{t_n - \mathrm{w}^T \phi(\mathrm{x}_n)\}^2$$

use maximum likelihood to determine w and β, we get

$$\nabla \ln p(\mathrm{t}|\mathrm{w}, \beta) = \sum_{n=1}^N \{t_n - \mathrm{w}^T \phi(\mathrm{x}_n)\} \phi(\mathrm{x}_n)^T = 0 \tag{2.4}$$

and we obtain

$$\mathrm{w}_{ML} = (\Phi^T \Phi)^{-1} \Phi^T \mathrm{t} \tag{2.5}$$

here Φ is an $N \times M$ matrix called the *design matrix*, and $\Phi_{nj} = \phi_j(\mathrm{x}_n)$ so each row of Φ is dependent on the same observation and each column is dependent on the same basis function.
$\Phi^\dagger = (\Phi^T \Phi)^{-1} \Phi^T$ is known as the *Moore-Penrose pseudo-inverse* of the matrix.
For w_0, we can see $w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \bar{\phi}_j$, where $\bar{t} = \frac{1}{N} \sum_{n=1}^N t_n, \bar{\phi}_j = \frac{1}{N} \sum_{n=1}^N \phi_j(\mathrm{x}_n)$. So the bias w_0 compensates for the different between the averages of the target values and the weighted sum of the averages of the basis function values. Also we can get$1/\beta_{ML} = \frac{1}{N} \sum_{n=1}^N \{t_n - \mathrm{w}_{ML}^T \phi(\mathrm{x}_n)\}^2$

Geometry of least squares

$\mathrm{t} = (t_1, \ldots, t_N)^T$ is a vector in N-dimensional space. Define y to be an N-dimensional vector whose n^{th} element is given by $y(\mathrm{x}_n, \mathrm{w})$, so y is an arbitrary linear combination of the vectors φ_j, it can live anywhere in the M-dimensional subspace.
Thus the least-squares solution for w corresponds to that choice of y that lies in subspace S and that is closest to t. so we choose the orthogonal projection of t onto the subspace.
some problems: singularity of $\Phi^T \Phi$

Sequential learning

SGD (*stochastic gradient descent*, aka, *sequential gradient descent*)

$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \eta \nabla E_n$$

So for the case of the sum-of-squares erroe function, this gives

$$\mathrm{w}^{\tau+1} = \mathrm{w}^{(\tau)} + \eta(t_n - \mathrm{w}^{(\tau)^T}\phi_n)\phi_n$$

where $\phi_n = \phi(\mathrm{x}_n) = (\phi_1(\mathrm{x}_n), \ldots, \phi_M(\mathrm{x}_n))^T$ *least-mean-squares* algorithm (*LMS*)

Regularized least squares

Total error function to be minimized takes the from : $E_D(\mathrm{w}) + \lambda E_W(\mathrm{w})$ Usually, $E_W(\mathrm{w}) = \frac{1}{2}\mathrm{w}^T\mathrm{w}$ And we can obtain $\mathrm{w} = (\lambda\mathrm{I} + \Phi^T\Phi)^{-1}\Phi^T\mathrm{t}$ by setting the gradient of total error function with respect to w to zero.

Multiple outputs

$\mathrm{t} \in R^K, K > 1$ is the outputs.
$\mathrm{y}(\mathrm{x}, \mathrm{W}) = W^T\phi(\mathrm{x})$ $W \in R^{M\times K}, \phi(\mathrm{x})$ is a M-dimensional column vector with elements $\phi_j(\mathrm{x})$ with $\phi_0(\mathrm{x}) = 1$.
M: the number of basis functions $\phi_1(\mathrm{x}), \ldots, \phi_M(\mathrm{x})$
K: outputs are in K-dimensional space.

$$p(\mathrm{t}|\mathrm{x}, \mathrm{W}, \beta) = \mathcal{N}(\mathrm{t}|\mathrm{W}^T\phi(\mathrm{x}), \beta^{-1}\mathrm{I})$$

If we have N observations, $\mathrm{t}_1, \ldots, \mathrm{t}_N$, into one matrix $\mathrm{T} \in R^{N\times K}$, combine $\mathrm{x}_1, \ldots, \mathrm{x}_N$ into one matrix $\mathrm{X} \in R^{N\times D}$

$$\ln p(\mathrm{T}|\mathrm{X}, \mathrm{W}, \beta) = \sum_{n=1}^{N} \ln \mathcal{N}(\mathrm{t}_n|\mathrm{W}^T\phi(\mathrm{x}_n), \beta^{-1}\mathrm{I}) = \frac{NK}{2}\ln(\frac{\beta}{2\pi}) - \frac{\beta}{2}\sum_{n=1}^{N}||\mathrm{t}_n - \mathrm{W}^T\phi(\mathrm{x}_n)||^2 \quad (2.6)$$

and we can get $\mathrm{W}_{ML} = (\Phi^T\Phi)^{-1}\Phi^T\mathrm{T}$, $\mathrm{w}_k = (\Phi^T\Phi)^{-1}\Phi^T\mathrm{t}_k$

2.2.2 The Bias-Variance Decomposition

Recall Section1.5.5 in PRML, which gave us the expected loss in regression problem:

$$\mathbb{E}[L] = \iint L(t, y(\mathbf{x}))p(\mathbf{x}, t)\mathrm{d}\mathbf{x}\mathrm{d}t \quad (2.7)$$

To minimize $\mathbb{E}[L]$ we set

$$y(\mathbf{x}) = \mathbb{E}_t[t|\mathbf{x}] \quad (2.8)$$

then we can rewrite the loss function:

$$L = \{y(\mathbf{x}) - t\}^2 = \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2 \quad (2.9)$$

$$= \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + 2\{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\} + \{\mathbb{E}[t|\mathbf{x}] - t\}^2 \quad (2.10)$$

Then, the expected loss is:

$$\mathbb{E}[L] = \int \{y(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 p(\mathbf{x})\mathrm{d}\mathbf{x} + \iint \{\mathbb{E}[t|\mathbf{x}] - t\}^2 p(\mathbf{x})\mathrm{d}\mathbf{x}\mathrm{d}t. \quad (2.11)$$

Denote $h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}]$, the second term is independent of $y(\mathbf{x})$, and arises from the intrinsic noise on the data. So we can only minimize the first term. And this term is nonnegative and if we get $h(\mathbf{x})$, we can set $y(\mathbf{x}) = h(\mathbf{x})$ and we'll get a minimal loss. But because of limited training data points and limited computational sources, we can never get exact $h(\mathbf{x})$, and we want to learn an optimal $y(\mathbf{x})$ from the training data.

Now suppose we have a number of data sets of the same size N and each drawn independently from the true distribution $p(\mathbf{x}, t)$. For each data set, we can get a regression function $y(\mathbf{x}; \mathcal{D})$, and for each particular data set, the first term takes the form as:

$$\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 \tag{2.12}$$

And we denote the expectation of the function value over all the data sets as $\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D})]$, then

$$\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 = \{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D}] + \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D}] - h(\mathbf{x})\}^2 \tag{2.13}$$

$$\tag{2.14}$$

So,

$$\mathbb{E}_{\mathcal{D}}[\{y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2] \tag{2.15}$$

$$= \{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2 + \mathbb{E}_{\mathcal{D}}[\{y(\mathbf{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x}; \mathcal{D}]\}^2] \tag{2.16}$$

The first term, called squared *bias* represents the extent to which the average prediction over all data sets differ from the desired regression function. The second term, called the *variance*, measures the extent to which the solutions for individual data sets vary around their average, and hence it measures the extent to which the function $y(\mathbf{x}; \mathcal{D}$ is sensitive to the choice of data set.

To sum up, the expected squared loss is:

$$\text{expectedloss} = (\text{bias})^2 + \text{variance} + \text{noise}. \tag{2.17}$$

There is a trade-off between bias and variance. If a model is very flexible, it will lead to very low bias while high variance, and if a model is very rigid, it will lead to very low variance while high bias.

There is an example in PRML section3.2 illustrating the trade-off between bias and variance.

2.2.3 Bayesian Linear Regression

Because effective model complexity, governed by the number of basis functions, needs to be controlled according to the size of the data set, the issue of deciding the appropriate model complexity for the particular problem which cannot be decided simply maximizing the likelihood function still exists. So we turn to Bayesian treatment of linear regression and we can avoid the over-fitting problem and also lead to automatic methods of determining model complexity using the training data alone.

Parameter distribution

we assume that precision parameter β is known and the prior probability distribution is as the form:

$$p(\mathrm{w}) = \mathcal{N}(\mathrm{w}|\mathrm{m}_0, \mathrm{S}_0) \tag{2.18}$$

so the posterior probability will be Gaussian, too

$$p(\mathrm{w}|\mathrm{t}) = \mathcal{N}(\mathrm{w}|\mathrm{m}_N, \mathrm{S}_N) \tag{2.19}$$

where $\mathrm{m}_N = \mathrm{S}_N(\mathrm{S}_0^{-1}\mathrm{m}_0 + \beta\Phi^T\mathrm{t})$ and $\mathrm{S}_N^{-1} = \mathrm{S}_0^{-1} + \beta\Phi^T\Phi$
To simplify the model we assume $\mathrm{S}_0 = \alpha^{-1}\mathrm{I}$ and $\mathrm{m}_0 = 0$. Then, maximization of this posterior distribution with respect to w is therefore equivalent to minimization of the sum-of-square error function with the addition of a quadratic regularization term.

Predictive distribution

In practice, we only want to predict t for new values of x rather than w. So we evaluate the *predictive distribution* defined by

$$p(t|\mathrm{t}, \alpha, \beta) = \int p(t|\mathrm{w}, \beta)p(\mathrm{w}|\mathrm{t}, \alpha, \beta)\mathrm{dw} \tag{2.20}$$

Equivalent kernel

We see that the predictive mean can be written in the form

$$y(\mathrm{x}, \mathrm{m}_N) = \mathrm{m}_N^T \phi(\mathrm{x}) = \beta\phi(\mathrm{x})^T \mathrm{S}_N \Phi^T \mathrm{t} = \sum_{n=1}^{N} \beta\phi(\mathrm{x})^T \mathrm{S}_N \phi(\mathrm{x}_n) t_n$$

we can define kernel $k(\mathrm{x}, \mathrm{x}') = \beta\phi(\mathrm{x})^T \mathrm{S}_N \phi(\mathrm{x}')$ is known as the *smoother matrix* or *equivalent kernel*. make predictions by taking linear combinations of the training set target values.

2.2.4 Bayesian Model Comparison

Consider the problem of model selection from a Bayesian perspective by simply involving the use of probabilities to represent uncertainty in the choice of model.
Given a training seet $\mathcal{D}$ we wish to evaluate the posterior distribution $p(\mathcal{M}_i|\mathcal{D}) \propto p(\mathcal{M}_i)p(\mathcal{D}|\mathcal{M}_i)$ the prior shows our preference for different models.
If we choose $p(\mathcal{M}_i)$ to be identical, then the interesting term is the model evidence $p(\mathcal{D}|\mathcal{M}_i)$. the model evidence is sometimes called *marginal likelihood*. We can conclude that the Bayesian framework avoids the problem of over-fitting and allows models to be compared on the basis of the training data alone. However, we still needs to make assumptions about the form of the model.

2.2.5 The Evidence Approximation

If we introduce hyperpriors over α and β, the predictive distribution is obtained by marginalizing over $\mathbf{w}$, α and β:

$$p(t|\mathbf{t}) = \iiint p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \beta)p(\alpha, \beta|\mathbf{t})\mathrm{d}\mathbf{w}\mathrm{d}\alpha\mathrm{d}\beta. \tag{2.21}$$

We omitted the input variable $\mathbf{x}$ to keep the notation uncluttered. This integration can not be analytically computed so we use approximation in which we set the hyperparameters to specific values determined by maximizing the marginal likelihood function obtained by first integrating over $\mathbf{w}$. This frame work is known in the statistics literature as *empirical Bayes* or *type 2 maximum likelihood* or *generalized maximum likelihood*, and in machine learning literature is also called *evidence approximation*.

The marginal likelihood function $p(\mathbf{t}|\alpha, \beta)$ is obtained by integrating over weight parameters $\mathbf{w}$, so that

$$p(\mathbf{t}, \alpha, \beta) = \int p(\mathbf{t}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)\mathrm{d}\mathbf{w}. \tag{2.22}$$

Using conditional distribution in linear-Gaussian model, we finally can write the log of marginal likelohood in the form

$$\ln p(\mathbf{t}|\alpha, \beta) = \frac{M}{2}\ln\alpha + \frac{N}{2}\ln\beta - E(\mathbf{m}_N) - \frac{1}{2}\ln|\mathbf{A}| - \frac{N}{2}\ln(2\pi) \tag{2.23}$$

in which

$$E(\mathbf{m}_N) = \frac{\beta}{2}\|\mathbf{t} - \Phi\mathbf{m}_N\|^2 + \frac{\alpha}{2}\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N \tag{2.24}$$

$$\mathbf{A} = \nabla\nabla E(\mathbf{w}) \tag{2.25}$$

$$\mathbf{m}_N = \beta\mathbf{A}^{-1}\Phi^{\mathrm{T}}\mathbf{t}. \tag{2.26}$$

Maximizing the evidence function

First consider maximization of $p(\mathbf{t}|\alpha, \beta)$ w.r.t α. From eq.2.24, assume

$$(\beta\Phi^{\mathrm{T}}\Phi)\mathbf{u}_i = \lambda_i\mathbf{u}_i \tag{2.27}$$

then $\mathbf{A}$ has eigenvalues $\alpha + \lambda_i$.

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}\ln|\mathbf{A}| = \frac{\mathrm{d}}{\mathrm{d}\alpha}\ln\prod_i(\lambda_i + \alpha) = \frac{\mathrm{d}}{\mathrm{d}\alpha}\sum_i\ln(\lambda_i + \alpha) = \sum_i\frac{1}{\lambda_i + \alpha} \tag{2.28}$$

So the stationary points of eq.2.23 w.r.t. α satisfy:

$$0 = \frac{M}{2\alpha} - \frac{1}{2}\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N - \frac{1}{2}\sum_i \frac{1}{\lambda_i + \alpha}. \tag{2.29}$$

So

$$\alpha\mathbf{m}_N^{\mathrm{T}}\mathbf{m}_N = M - \alpha\sum_i \frac{1}{\lambda_i + \alpha} = \gamma. \tag{2.30}$$

$$\gamma = \sum_i \frac{\lambda_i}{\alpha + \lambda_i} \tag{2.31}$$

and

$$\alpha = \frac{\gamma}{\mathbf{m}_N^{\mathrm{T}}\mathbf{m}} \tag{2.32}$$

Note that this is an implicit solution for α since γ and $\mathbf{m}_N$ are both dependent of α. And we can compute *alpha* by lots of iterations until convergence of α, γ and $\mathbf{m}_N$.

Then we can similarly maximize the log likelihood function w.r.t β. Notice that the eigenvalues λ_i is defined proportional to β, and hence $\mathrm{d}\lambda_i/\mathrm{d}\beta = \lambda_i/\beta$, so

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\ln|\mathbf{A}| = \frac{\mathrm{d}}{\mathrm{d}\beta}\sum_i \ln(\lambda_i + \alpha) = \frac{1}{\beta}\sum_i \frac{\lambda_i}{\lambda_i + \alpha} = \frac{\gamma}{\beta} \tag{2.33}$$

finally

$$\frac{1}{\beta} = \frac{1}{N - \gamma}\sum_{n=1}^{N}\{t_n - \mathbf{m}_N^{\mathrm{T}}\phi(\mathbf{x}_n)\}^2. \tag{2.34}$$

This is also an implicit solution for β.

2.2.6 Limitations of Fixed Basis Functions

basis function $\phi_j(\mathrm{x})$ are fixed before the training data set is observed.
curse of dimensionality

2.3 Linear Models for Classification

This chapter aims to solve classification tasks linear models.
Three approaches:

- discriminant function, directly assigns each vector x to a specific class.
- models conditional probability distribution $p(\mathcal{C}_k|\mathrm{x})$ in an inference stage and then subsequently uses this distribution to make optimal decisions:
  - directly model the conditional probability
  - generative approach. model the class-conditional densities given by $p(\mathrm{x}|\mathcal{C}_k)$, together with the prior probabilities $p(\mathcal{C}_k)$

For classification problem, we consider to generalize a linear model by transform the linear function of $\mathbf{w}$ using a nonlinear function $f(\cdot)$ so that

$$y(\mathbf{x}) = f(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0) \tag{2.35}$$

In the machine learning literature $f(\cdot)$ is known as *activation function*, whereas its inverse is called a *link function* in statistics literature. So the model is no longer linear in the parameters due to the presence of the nonlinear activation function.

2.3.1 Discriminant Functions

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \tag{2.36}$$

For two-class situation, the normal distance from the origin to the decision surface is given by is given by

$$\frac{\mathbf{w}^T\mathbf{x}}{||\mathbf{w}||} = -\frac{w_0}{||\mathbf{w}||} \tag{2.37}$$

for multiple classes, it's kind of difficult to reduce to 2-class. a single K-class discriminant comprising K linear functions of the form: $y_k(\mathrm{x}) = \mathrm{w}_k^T\mathrm{x} + w_{k0}$ and then assigning a point x to class $\mathcal{C}_k$ if $y_k(\mathrm{x})$ is the largest.
approaches to learning the discriminant functions:

- least squares for classification
- fisher's linear discriminant
- perceptron algorithm

Least squares for classification

We have K classes and each class $\mathcal{C}$ is described by its own linear model so that

$$y_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}}\mathbf{x} + w_{k0}. \tag{2.38}$$

Using vector notation we can rewrite it in the form of:

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^{\mathrm{T}}\tilde{\mathbf{x}} \tag{2.39}$$

where $\tilde{\mathbf{W}}$ is a matrix whose k^{th} column comprises the $D+1$-dimensional vector $\tilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^{\mathrm{T}})^{\mathrm{T}}$ and $\tilde{\mathbf{x}}$ is the corresponding augmented input vector $(1, \mathbf{x}^{\mathrm{T}})^{\mathrm{T}}$. A new input $\mathbf{x}$ is then assigned to the class for which the output $y_k = \tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}$ is the largest.

We now determine $\tilde{\mathbf{W}}$ by minimizing a sum-of-square error function:

$$E_D(\tilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\{(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\tilde{\mathbf{X}}\tilde{\mathbf{W}} - \mathbf{T})\} \tag{2.40}$$

where $\mathbf{T}$ is defined as a matrix whose n^{th} row is the vector $\mathbf{t}_n^{\mathrm{T}}$ and $\tilde{\mathbf{X}}$ is a matrix whose n^{th} row is $\tilde{\mathbf{x}}_n^{\mathrm{T}}$. We then obtain the solution for optimal $\tilde{\mathbf{W}}$ in the form

$$\tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^{\mathrm{T}}\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \tilde{\mathbf{X}}^{\dagger}\mathbf{T} \tag{2.41}$$

Least square model fails obviously because when we recall that it correspond to maximum likelihood under the assumption of a Gaussian conditional distribution, whereas binary target vectors clearly have a distribution that is far from Gaussian.

Fisher's linear discriminant

Here we consider classification problem in the view of dimension reduction.

Consider two-class problem first, we can select a projection that maximizes the class separation. Assume there are N_1 points in class $\mathcal{C}_1$ and N_2 points in class $\mathcal{C}_2$, define mean vectors:

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}\mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}\mathbf{x}_n. \tag{2.42}$$

So projection class means will be:

$$m_k = \mathbf{w}^{\mathrm{T}}\mathbf{m}_k, \ k = 1, 2 \tag{2.43}$$

and we want to maximize

$$m_2 - m_1 = \mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1). \tag{2.44}$$

This can be arbitrarily large simply by increasing the magnitude of $\mathbf{w}$. So we add a constraint $\sum_i w_i^2 = 1$, and then using Lagrange multiplier we obtain $\mathbf{w} \propto (\mathbf{m}_2 - \mathbf{m}_1)$. But this approach is not

robust because some well separated samples in original two-dimensional space could overlapping after projected onto the vector $\mathbf{m}_2 - \mathbf{m}_1$. So Fisher's linear discriminant want to solve this problem.

The idea proposed by Fisher is to maximize a function that will give a large separation between the projected class means while also giving a small variance within each class, thereby minimize the class overlap. We define

$$s_k^2 = \sum_{n \epsilon \mathcal{C}_k} (y_n - m_k)^2 \tag{2.45}$$

which is the within class $\mathcal{C}_k$ variance, where $y_n = \mathbf{w}^{\mathrm{T}}\mathbf{x}_n$. And *Fisher criterion* is defined by

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}} \tag{2.46}$$

where $\mathbf{S}_{\mathrm{B}}$ is the *between-class* covariance matrix

$$\mathbf{S}_{\mathrm{B}} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}} \tag{2.47}$$

and $\mathbf{S}_{\mathrm{W}}$ is the *within-class* covariance matrix

$$\mathbf{S}_{\mathrm{W}} = \sum_{n \epsilon \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^{\mathrm{T}} + \sum_{n \epsilon \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^{\mathrm{T}}. \tag{2.48}$$

We find that J is maximized when

$$(\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w})\mathbf{S}_{\mathrm{W}}\mathbf{w} = (\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w})\mathbf{S}_{\mathrm{B}}\mathbf{w}. \tag{2.49}$$

Since do not care about the magnitude of $\mathbf{w}$, we obtain

$$\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1). \tag{2.50}$$

Now we consider the relation between Fisher's discriminant and least squares. If we adopt a slightly different (in contrast to 1-of-K coding)target encoding scheme, then the least-squares solution for the weights becomes equivalent to the Fisher solution. More details can be found in section4.1.5 of *PRML*.

As for K-classes situation, the solution is similar to 2-classes situation and more details can be found in section4.1.6 of *PRML*.

The perceptron algorithm

We choose the nonlinear activation function $f(\cdot)$ to be a step function of the form

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0. \end{cases} \tag{2.51}$$

The algorithm used to determine the parameters $\mathbf{w}$ of the perceptron can most easily be motivated by error function minimization. A natural choice for error function is the total number of misclassified patterns. However this error function is discontinuous at some points while the gradient is zero all everywhere.

We therefore consider an alternative error function as *perceptron criterion*. For right predicted input, we have

$$\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)t_n > 0. \tag{2.52}$$

The perceptron criterion is therefore given by

$$E_p(\mathbf{w}) = -\sum_{n \epsilon mcal M} \mathbf{w}^{\mathrm{T}}\phi_n t_n \tag{2.53}$$

where $\mathcal{M}$ denotes the set of all misclassified patterns. Then the change in the weight vector $\mathbf{w}$ is given by

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_p(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta\phi_n t_n. \tag{2.54}$$

2.3.2 Probabilistic Generative Models

Here we shall adopt a generative approach in which we model the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$, as well as the class prior $p(\mathcal{C}_k$, and then we want to compute posterior $p(\mathcal{C}_k|\mathbf{x})$.

Consider 2-classes case first, we have

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}. \tag{2.55}$$

Then we define

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \tag{2.56}$$

so

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1 + \exp(-a)} = \sigma(a). \tag{2.57}$$

And we call $\sigma(a) = \frac{1}{1+\exp(a)}$ *logistic sigmoid* function. The inverse of the sigmoid function is given by

$$a = \ln(\frac{\sigma}{1 - \sigma}) \tag{2.58}$$

and is known as the *logit* function. It represents the log of the ratio of probabilities $\ln[p(\mathcal{C}_1|\mathbf{x})/p(\mathcal{C}_2|\mathbf{x})]$ for the two classes, also known as the *log odds*.

Given eq.2.57, we shall shortly consider situations in which $a(\mathbf{x})$ is a linear function of $\mathbf{x}$, in which case the posterior probability is governed by a generalized linear model.

For $K > 2$ classes, we have

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j} = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \tag{2.59}$$

where a_k is defined by

$$a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k). \tag{2.60}$$

The normalized exponential is also known as the *softmax function*.

Then, we'll investigate the consequences of choosing specific forms for the class-conditional densities, looking first at continuous input variables $\mathbf{x}$ and then discussing briefly the case of discrete inputs.

For continuous input, let us assume that the class-conditional densities are Gaussian and firstly assume that all classes share the same covariance matrix.

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2})} \frac{1}{|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\} \tag{2.61}$$

Then for case of two classes, we have

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0) \tag{2.62}$$

where

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{2.63}$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}. \tag{2.64}$$

Because we have assumption of common covariance matrices, so we have cancelled the quadratic terms from Gaussian densities. And the prior $p(\mathcal{C}_k)$ only appears in the bias so when it is changed, the decision boundary will only make parallel shifts. If we relax the assumption of a shared covariance matrix and allow each class conditional density $p(\mathbf{x}|\mathcal{C}_k)$ to have its own Σ_k, then we will obtain quadratic functions of $\mathbf{x}$, giving rise to a *quadratic discriminant*.

Maximum Likelihood solution

We have data set $\{\mathbf{x}_n, t_n\}, n = 1, 2, \ldots, N$, where $t_n = 1$ denotes class $\mathcal{C}_1$ and $t_n = 0$ denotes $\mathcal{C}_2$. And we have prior class probability $p(\mathcal{C}_1) = \pi$ and $p(\mathcal{C}_2) = 1 - \pi$, then

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma) \tag{2.65}$$

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma) \tag{2.66}$$

So the likelihood function is given by

$$p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) = \prod_{n=1}^{N} [\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)]^{t_n} [(1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)]^{1-t_n} \tag{2.67}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$.

By maximizing the log of likelihood, w.r.t π we obtain

$$\pi = \frac{N_1}{N_1 + N_2} \tag{2.68}$$

where N_1 denotes the total number of data points in class $\mathcal{C}_1$ and so as N_2.

Then we pick out of the log likelihood function those terms that depend on $\boldsymbol{\mu}_1$ giving

$$\sum_{n=1}^{N} t_n \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma) = -\frac{1}{2}\sum_{n=1}^{N} t_n(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const.} \tag{2.69}$$

Setting the derivative w.r.t $\boldsymbol{\mu}_1$ to zero and we obtain

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n\mathbf{x}_n \tag{2.70}$$

which is simply the mean of all the input vectors $\mathbf{x}_n$ assigned to class $\mathcal{C}_1$. Similarly,

$$\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N}(1 - t_n)\mathbf{x}_n. \tag{2.71}$$

Finally we pick out the terms in the log likelihood function that depend on Σ, we have

$$-\frac{N}{2}\ln|\Sigma| - \frac{N}{2}\mathrm{Tr}\{\Sigma^{-1}\mathbf{S}\} \tag{2.72}$$

where

$$\mathbf{S} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2 \tag{2.73}$$

$$\mathbf{S}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \tag{2.74}$$

$$\mathbf{S}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}} \tag{2.75}$$

Using the standard result for the maximum likelihood solution for a Gaussian distribution, we see that $\Sigma = \mathbf{S}$, which represents a weighted average of the covariance matrices associated with each of the two classes separately.

This result is easily extended to K classes.

Discrete features

For simplicity, we look at binary feature values $x_i \in \{0, 1\}$, and the input is D-dimensional. Here we make the *naive Bayes* assumption in which the feature values are treated as independent, conditioned on the class $\mathcal{C}_k$. Thus, we have class-conditional distribution of the form

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{i=1}^{D}\mu_{ki}^{x_i}(1 - \ mu_{ki})^{1-x_i}. \tag{2.76}$$

For each dimension of the input feature $\mathbf{x}$, the value $x_i \sim \mathrm{Bern}(x_i|\mu_{ki})$ where k denotes the class $\mathcal{C}_k$. Substituting eq.2.76 into eq.2.60 we obtain

$$a_k(\mathbf{x}) = \sum_{i=1}^{D}\{x_i \ln \mu_{ki}\} + (1 - x_i)\ln(1 - \mu_{ki}) + \ln p(\mathcal{C}_k) \tag{2.77}$$

which is again linear functions of the input values x_i.

Exponential Family

As we have seen, for both Gaussian distributed and discrete inputs, the posterior class probabilities are given by generalized linear models with logistic sigmoid ($K = 2$ classes) or softmax ($K \geq 2$ classes) activation functions. These are particular cases of a more general result obtained by assuming that the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ are members of the exponential family of distributions.

By assuming the distribution of $\mathbf{x}$ can be written in the form

$$p(\mathbf{x}|\boldsymbol{\lambda}_k) = h(\mathbf{x})g(\boldsymbol{\lambda}_k)\exp\{\boldsymbol{\lambda}_k^{\mathrm{T}}\mathbf{x}\} \tag{2.78}$$

we can prove that for exponential family distributions, the posterior class probabilities are given by generalized linear models with logistic sigmoid ($K = 2$ classes) or softmax ($K \geq 2$ classes) activation functions.

2.3.3 Probabilistic Discriminative Models

We have discussed generative models to do classification last section, and now we have an alternative approach to do classification which is to use the functional form of the generalized linear model explicitly and to determine its parameters directly by using maximum likelihood. and there is an algorithm *IRLS*, *iterative reweighted least squares* to find solutions.

fixed basis functions

We first make a nonlinear transformation of the input $\mathbf{x}$ using a vector of basis function $\phi(\mathbf{x})$. Linear decision boundaries in the ϕ space while nonlinear in the x space.

logistic regression

The posterior probability of class $\mathcal{C}_1$ can be written as a logistic sigmoid acting on a linear function of the feature vector ϕ:

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^{\mathrm{T}}\phi) \tag{2.79}$$

with $p(\mathcal{C} - 2|\phi) = 1 - p(\mathcal{C}_1|\phi)$. Here $\sigma(\cdot)$ is the *logistic sigmoid* function. In the termninology of statistics this model is also known as *logistic regression*.

We now use maximum likelihood to determine the parameters of the logistic regression model.

For a data set $\{\phi_n, t_n\}$ where $t_n \in \{0, 1\}$ and $\phi_n = \phi(\mathbf{x}_n)$, with $n = 1, \cdots, N$ the likelihood function can be written as

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}\{1 - y_n\}^{1-t_n} \tag{2.80}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$ and $y_n = p(\mathcal{C}_1|\phi_n)$. We define an error function by taking the negative logarithm of the likelihood, which gives the *cross-entropy* error function in the form

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} \tag{2.81}$$

where $y_n = \sigma(a_n)$ and $a_n = \mathbf{w}^{\mathrm{T}}\phi_n$. Then

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n. \tag{2.82}$$

Iterative reweighted least squares

For logistic regression, there is no longer a closed-form solution, due to the nonlinearity of the logistic sigmoid function. However, the departure from a quadratic form is not substantial. To be precise, the error function is concave, as we shall see shortly, and hence has a unique minimum. The Newton-Raphson update for minimizing a function $E(\mathrm{w})$ is

$$\mathbf{w}^{(\mathrm{new})} = \mathbf{w}^{(\mathrm{old})} - \mathbf{H}^{-1}\nabla E(\mathbf{w}) \tag{2.83}$$

If we apply the Newton-Raphson method to the linear regression with sum-of -squares error function in section2.2, we'll find the formula gives the exact solution in one step.

While for logistic regression model with cross-entropy error function, we see the gradient and Hessian of this error function are given by

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n = \Phi^T(\mathbf{y} - \mathbf{t}) \tag{2.84}$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1 - y_n)\phi_n\phi_n^T = \Phi^{\mathrm{T}}\mathbf{R}\Phi \tag{2.85}$$

where R is a diagonal matrix and $R_{nn} = y_n(1 - y_n)$. So Hessian matrix is no longer constant but depends on $\mathbf{w}$ through the weighting matrix $\mathbf{R}$. So we must apply the normal equations iteratively, each time using the new weight vector $\mathbf{w}$ to compute a revised weighing matrix $\mathbf{R}$. For this reason, the algorithm is known as $IRLS$.

Multiclass logistic regression

Extending two-class case to K-class case, more details seen in section4.3.4 in *PRML*.

2.3.4 The Laplace Approximation

Refer to another section in this notes: 1.5.

2.3.5 Bayesian Logistic Regression

Exact Bayesian inference for logistic regression is intractable. So here we consider the application of the Laplace approximation to the problem.

Laplace approximation

Because we seek a Gaussian representation for the posterior distribution, it's natural to begin with a Gaussian prior:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0) \tag{2.86}$$

where $\mathbf{m}_0$ and $\mathbf{S}_0$ are fixed hyperparameters. Then the posterior over $\mathbf{w}$ is given by

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w})p(\mathbf{t}|\mathbf{w}). \tag{2.87}$$

Then

$$\ln p(\mathbf{w}|\mathbf{t}) = -\frac{1}{2}(\mathbf{w} - \mathbf{m}_0)^{\mathrm{T}}\mathbf{S}_0^{-1}(\mathbf{w} - \mathbf{m}_0) \tag{2.88}$$

$$+ \sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} + \text{const} \tag{2.89}$$

where $y_n = \sigma(\mathbf{w}^{\mathrm{T}}\phi_n)$. To obtain a Gaussian approximation to the posterior distribution, we first maximize the posterior distribution to give the MAP solution $\mathbf{w}_{\mathrm{MAP}}$, which defines the mean of the Gaussian. Then the covariance is given by

$$\mathbf{S}_N = -\nabla\nabla \ln p(\mathbf{w}|\mathbf{t}) = \mathbf{S}_0^{-1} + \sum_{n=1}^{N} y_n(1 - y_n)\phi_n\phi_n^{\mathrm{T}}. \tag{2.90}$$

The Gaussian approximation therefore takes the form

$$q(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_{\mathrm{MAP}}, \mathbf{S}_N). \tag{2.91}$$

Then the remaining task is to marginalize w.r.t this distribution in order to make predictions.

Predictive distribution

The predictive distribution for class $\mathcal{C}_1$, given a new feature vector $\phi(\mathbf{x})$, is given by

$$p(\mathcal{C}_1|\boldsymbol{\phi}, \mathbf{t}) = \int p(\mathcal{C}_1|\boldsymbol{\phi}, \mathbf{w}) p(\mathbf{w}|\mathbf{t}) \mathrm{d}\mathbf{w} \simeq \int \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) q(\mathbf{w}) \mathrm{d}\mathbf{w}. \tag{2.92}$$

and $p(\mathcal{C}_2|\boldsymbol{\phi}, \mathbf{t}) = 1 - p(\mathcal{C}_1|\boldsymbol{\phi}, \mathbf{t})$. If we use

$$\sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) = \int \delta(a - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) \sigma(a) \mathrm{d}a \tag{2.93}$$

to substitute into eq2.92 and denote $a = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}$,we will obtain

$$\int \sigma(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) q(\mathbf{w}) \mathrm{d}\mathbf{w} = \iint \delta(a - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) \sigma(a) q(\mathbf{w}) \mathrm{d}a \mathrm{d}\mathbf{w} = \int \sigma(a) p(a) \mathrm{d}a. \tag{2.94}$$

where

$$p(a) = \int \delta(a - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}) q(\mathbf{w}) \mathrm{d}\mathbf{w}. \tag{2.95}$$

Because $q(\mathbf{w})$ is Gaussian, $p(a)$ is also Gaussian, so we just need to find out the mean and covariance:

$$\mu_a = \mathbb{E}[a] = \int p(a) a \mathrm{d}a = \int q(\mathbf{w}) \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi} \mathrm{d}\mathbf{w} = \mathbf{w}_{\mathrm{MAP}}^{\mathrm{T}} \boldsymbol{\phi} \tag{2.96}$$

$$\sigma_a^2 = \mathrm{var}[a] = \int p(a) \{a^2 - \mathbb{E}[a]^2\} \mathrm{d}a = \boldsymbol{\phi}^{\mathrm{T}} \mathbf{S}_N \boldsymbol{\phi}. \tag{2.97}$$

Thus our variational approximation to the predictive distribution becomes

$$p(\mathcal{C}_1|\mathbf{t}) = \int \sigma(a) p(a) \mathrm{d}a = \int \sigma(a) \mathcal{N}(a|\mu_a, \sigma_a^2) \mathrm{d}a. \tag{2.98}$$

2.4 Neural Networks

In this section, our focus is on neural networks as efficient models for statistical pattern recognition.

2.4.1 Feed-forward Network Functions

The linear models for regression and classification discussed in Chapters 3 and 4, respectively, are based on linear combinations of fixed nonlinear basis functions $\phi_j(\mathrm{x})$ and take the form

$$y(\mathrm{x}, \mathrm{w}) = f(\sum_{j=1}^{M} w_j \phi_j(\mathrm{x})) \tag{2.99}$$

where $f(\cdot)$ is a nonlinear activation function in the case of classification and is the identity in the case of regression. Our goal is to extend this model by making the basis function $\phi_j(\mathrm{x})$ depend on parameters and then allow these parameters to be adjusted, along with the coefficients $\{w_j\}$, during training.

For the first layer,

$$a_j = \sum_{j=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.100}$$

$$z_j = h(a_j) \tag{2.101}$$

This quantities correspond to the outputs of the basis functions in eq2.99 that, in the context of neural networks, are called *hidden units*. The activation functions $h(\cdot)$ are generally chosen to be sigmoidal function or the 'tanh'. Following eq2.99 these values are again linearly combined to give *output unit activations*

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.102}$$

where $k = 1, \ldots, K$ and K is the total number of outputs. Finally, the output unit activations are transformed using an appropriate activation function to give a set of network output y_k. For standard regression problems, the activation function is the identity so that $y_k = a_k$. Similarly, for multiple binary classification problems, $y_k = \sigma(a_k)$. And for multiclass problems, a softmax function is used.

Combining these various stages to give the overall network function that, for sigmoidal output unit activation functions, we obtain

$$y_k(\mathbf{x}, \mathbf{w}) = \sigma(\sum_{j=1}^{M} w_{kj}^{(2)} h(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}) + w_{k0}^{(2)}) \tag{2.103}$$

where the set of all weight and bias parameters have been grouped together into a vector $\mathbf{w}$. The process of evaluating 2.103 can then be interpreted as a *forward propagation* of information through the network.

One property of feed-forward networks, which will play a role when we consider Bayesian model comparison, is that multiple distinct choices for the weight vector $\mathbf{w}$ can all give rise to the same mapping function from inputs to outputs.

2.4.2 Network Training

There is a simple idea to train the parameter: minimize the error function. e.g.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \|\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n\|^2 \tag{2.104}$$

However we can provide a more general view of network training by first giving a probabilistic interpretation to the network outputs.

Consider firstly **regression problem** with a single target variable t and t has a Gaussian distribution with an $\mathbf{x}$-dependent mean:

$$p(t|\mathbf{x}, \mathbf{w}) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}). \tag{2.105}$$

For the conditional distribution given by 2.105, it is sufficient to take the output unit activation function to be identity. Given a data set of N i.i.d observations $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N$ along with corresponding target values $\mathbf{t} = \{t_1, \ldots, t_N\}$, we can construct the corresponding likelihood function

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} p(t_n|\mathbf{x}_n, \mathbf{w}, \beta). \tag{2.106}$$

Taking the negative logarithm, we obtain the error function

$$\frac{\beta}{2} \sum_{n=1}^{N} \{y(\mathbf{x}_n, \mathbf{w}) - t_n\}^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln(2\pi). \tag{2.107}$$

Consider first the determination of $\mathbf{w}$, we want to minimize the error function given by

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(\mathbf{x}, \mathbf{w}) - t_n\}^2. \tag{2.108}$$

The value of $\mathbf{w}$ found by minimizing $E(\mathbf{w})$ will be denoted $\mathbf{w}_{\mathrm{ML}}$. Then

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N} \sum_{n=1}^{N} \{y(\mathbf{x}_n, \mathbf{w}_{\mathrm{ML}} - t_n\}^2. \tag{2.109}$$

Then consider the case of **binary classification problem**. The conditional distribution given inputs is then a Bernoulli distribution of the form

$$p(t|\mathbf{x}, \mathbf{w}) = y(\mathbf{x}, \mathbf{w})^t \{1 - y(\mathbf{x}, \mathbf{w})\}^{1-t}. \tag{2.110}$$

If we consider a training set, then the error function is then a *cross-entropy* error function of the form

$$E(\mathbf{w}) = \sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} \tag{2.111}$$

Finally we consider the standard multiclass classification problem in which each input is assigned to one of K mutually exclusive classes, and the networks outputs are interpreted as $y_k(\mathbf{x}, \mathbf{w}) = p(t_k = 1|\mathbf{x})$, leading to the following error function

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln y_k(\mathbf{x}_n, \mathbf{w}). \tag{2.112}$$

And the output unit activation function is given by

$$y_k(\mathbf{x}, \mathbf{w}) = \frac{\exp(a_k(\mathbf{x}, \mathbf{w})}{\sum_j \exp(a_j(\mathbf{x}_n, \mathbf{w}))} \tag{2.113}$$

Parameter optimization

Then we turn next to the task of finding a weight vector w which minimizes the chosen function $E(\mathrm{w})$. But the error function typically has a highly nonlinear dependence on the weights and bias parameters, and so there will be many points in weight space at which the gradient vanishes.

Because there is clearly no hope of finding an analytical solution to the equation $\nabla E(\mathbf{w}) = 0$ we resort to iterative numerical procedures.**The optimization of continuous nonlinear functions is a widely studied problem.** Most techniques involve choosing some initial value $\mathbf{w}^{(0)}$ for the weight vector and then moving through weight space in a succession of steps of the form

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \delta\mathbf{w}^{(\tau)}. \tag{2.114}$$

Then gradient information is important here. So we consider a local approximation of the error function based on Taylor expansion.

Local quadratic approximation

Consider the Taylor expansion of $E(\mathbf{w})$ around some point $\hat{\mathbf{w}}$ in weight space

$$E(\mathbf{w}) \simeq E(\hat{\mathbf{w}}) + (\mathbf{w} - \hat{\mathbf{w}})^{\mathrm{T}}\mathbf{b} + \frac{1}{2}(\mathbf{w} - \hat{\mathbf{w}})^{\mathrm{T}}\mathbf{H}(\mathbf{w} - \hat{\mathbf{w}}) \tag{2.115}$$

where

$$\mathbf{b} \equiv \nabla E|_{\mathbf{w}=\hat{\mathbf{w}}} \tag{2.116}$$

and the Hessian matrix $\mathbf{H} = \nabla\nabla E$.

From 2.115, we obtain

$$\nabla E \simeq \mathbf{b} + \mathbf{H}(\mathbf{w} - \hat{\mathbf{w}}). \tag{2.117}$$

Considering the minimum point $\mathbf{w}^*$, we have

$$E(\mathbf{w}) = E(\mathbf{w}^*) + \frac{1}{2}(\mathbf{w} - \mathbf{w}^*)^{\mathrm{T}}\mathbf{H}(\mathbf{w} - \mathbf{w}^*). \tag{2.118}$$

So if $\mathbf{w}^*$ is a minimum point, the Hessian matrix $\mathbf{H}$ should be *positive definite*, which means all the eigenvalues of $\mathbf{H}$ is positive.

Gradient descent optimization

The simplest approach to using gradient information is to choose the weight update in 2.114 to comprise a small step in the direction of the negative gradient, so that

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E(\mathbf{w}^{(\tau)}) \tag{2.119}$$

where η is called *learning rate*. Note that the whole data set is used and this technique was once called *batch* method. At each step the weight vector is moved in the direction of the greatest rate of decrease of the error function, and so this approach is known as *gradient descent* or *steepest descent*.

For batch optimization, there are more efficient methods, such as *conjugate gradients* and *quasi-Newton* methods.

Also, there is an online version of gradient descent called *sequential gradient descent or stochastic gradient descent*, make update to the weight vector based on one data point at a time, so that

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_n(\mathbf{w}^{(\tau)}). \tag{2.120}$$

2.4.3 Error Backpropagation

Goal: find an efficient technique for evaluating the gradient of an error function $E(\mathrm{w})$ for a feed-forward neural network.

Algorithm:

1. Apply an input vector xn to the network and forward propagate through the network to find the activations of all the hidden and output units.
2. Evaluate the δk for all the output units using.
3. Backpropagate the δ's using to obtain δj for each hidden unit in the network.
4. Evaluate the required derivatives.

2.4.4 Jacobian Matrix and Hessian Matrix

Refer to 1.6.

2.4.5 Regularization in Neural Networks

To avoid over-fitting, we can add a regularizer to the error function:

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}. \tag{2.121}$$

This regularizer can be interpreted as the negative logarithm of a zero-mean Gaussian prior distribution over the weight vector $\mathbf{w}$.

There are also many other ways to avoid over-fitting:

- Early stopping
- Invariance
- Tangent propagation
- Training with transformed data

More details can be found in section5.5 in *PRML*.

Convolutional networks

the basis of convolutional network is to build the invariance property into the structure of a neural network And it is realized through three mechanisms :

- local receptive field
- weight sharing
- subsampling

However, this approach ignores a key property of images, which is that nearby pixels are more strongly correlated than more distant pixels.

2.4.6 Mixture Density networks

2.4.7 Bayesian Neural Networks

So far, our discussion of neural networks has focussed on the use of maximum likelihood to determine the network parameters (weights and biases). Regularized maximum likelihood can be interpreted as a MAP (maximum posterior) approach in which the regularizer can be viewed as the logarithm of a prior parameter distribution. However, in a Bayesian treatment we need to marginalize over the distribution of parameters in order to make predictions.

However, the highly nonlinear dependence of the network function on the parameter values means that an exact Bayesian treatment can no longer be found. In fact, the log of the posterior distribution will be nonconvex, corresponding to the multiple local minima in the error function.

So, we will approximate the posterior distribution by a Gaussian, centred at a mode of the true posterior.

2.5 Kernel Methods

A big difference between this chapter and previous ones is that the training data points or a subset of them, are kept and used also during the prediction phase rather than are discarded and predictions for new inputs are based purely on the learned parameter vector w.

radial basis functions: depend only on the magnitude of the distance between the arguments so that $k(\mathrm{x}, \mathrm{x}') = k(||\mathrm{x} - \mathrm{x}'||)$

2.5.1 Dual Representations

Many linear models for regression and classification can be reformulated in terms of a dual representation in which the kernel function arises naturally. Here we consider a linear regression model whose parameters are determined by minimizing a regularized sum-of-squares error function given by

$$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) - t_n\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}. \tag{2.122}$$

By setting the gradient of $J(\mathbf{w})$ w.r.t $\mathbf{w}$ equal to zero, we see

$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\{\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) - t_n\}\phi(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\phi(\mathbf{x}_n) = \Phi^{\mathrm{T}}\mathbf{a} \tag{2.123}$$

where Φ is the design matrix, whose n^{th} row is given by $\phi(\mathbf{x}_n)^{\mathrm{T}}$. Here the vector $\mathbf{a} = (a_1, \ldots, a_N)^{\mathrm{T}}$, and

$$a_n = -\frac{1}{\lambda}\{\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) - t_n\}. \tag{2.124}$$

Then if we reformulate 2.122 in terms of the parameter vector $\mathbf{a}$ rather than $\mathbf{w}$ and set the gradient w.r.t. $\mathbf{a}$ to zero, we can obtain

$$\mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t} \tag{2.125}$$

and

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a} \tag{2.126}$$

where $\mathbf{K}$ is define as *Gram matrix* $\mathbf{K} = \Phi\Phi^{\mathrm{T}}$:

$$K_{nm} = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m). \tag{2.127}$$

Then, given a new input $\mathbf{x}$ we obtain the prediction as:

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\Phi\phi(\mathbf{x}) = \mathbf{k}(\mathbf{x})^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t} \tag{2.128}$$

where we define the vector $\mathbf{k}(\mathbf{x})$ with elements $k_n(\mathbf{x}) = k(\mathbf{x}_n, \mathbf{x})$. Thus we see that the dual formulation allows the solution to the least-squares problem to be expressed entirely in terms of the kernel function $k(\mathbf{x}, \mathbf{x}')$.

2.5.2 Construction of Kernels

How to construct a valid kernel is a problem. One approach is to choose a feature space mapping $\phi(\mathrm{x})$ and then use it to find the corresponding kernel.

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}') = \sum_{i=1}^{M}\phi_i(\mathbf{x})\phi_i(\mathbf{x}') \tag{2.129}$$

But, an alternative approach is to construct kernel functions directly, at the same time, we should ensure it is valid. A necessary and sufficient condition for a function $k(\mathrm{x}, \mathrm{x}')$ to be a valid kernel is that the Gram matrix K whose elements are given by $k(\mathrm{x}_n, \mathrm{x}_m)$, should be positive semidefinite for all possible choices of the set $\{\mathrm{x}_n\}$.

And there are a series techniques to construct valid kernels out of simpler kernels.

A commonly used kernel takes the form:

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/2\sigma^2)$$

called 'Gaussian' kernel. We can see that this is a valid kernel by expanding the square

$$\|\mathbf{x}' - \mathbf{x}\|^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} + (\mathbf{x}')^{\mathrm{T}}\mathbf{x}' - 2\mathbf{x}^{\mathrm{T}}\mathbf{x}' \tag{2.130}$$

to give

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\mathbf{x}^{\mathrm{T}}\mathbf{x}/2\sigma^2)\exp(\mathbf{x}^{\mathrm{T}}\mathbf{x}'/2\sigma^2)\exp(-(\mathbf{x}')^{\mathrm{T}}\mathbf{x}'/2\sigma^2) \tag{2.131}$$

then making use of some properties together with the validity of the linear kernel $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'$, we can obtain the validity of Gaussian kernel.

And another powerful approach to the construction of kernels starts from a probabilistic generative model.

$$k(\mathrm{x}, \mathrm{x}') = p(\mathrm{x})p(\mathrm{x}')$$

2.5.3 Radial Basis Function Networks

In order to find a smooth function $f(\mathbf{x})$ that fits every target value exactly, we express $f(\mathrm{x})$ as a linear combination of radial basis functions, one **centred on every data point**

$$f(\mathrm{x}) = \sum_{n=1}^{N} w_n h(\| \mathrm{x} - \mathrm{x}_n \|) \tag{2.132}$$

we can use least squares to find $\{w_n\}$. But because the number of parameters is exactly the number of data points, over-fitting will certainly become a concern.

Another motivation for radial basis functions comes from a consideration of the interpolation problem when the input (rather than the target) variable is noisy, denoting the noise as $\boldsymbol{\xi}$, then the sum-of-squares error function becomes

$$E = \frac{1}{2}\sum_{n=1}^{N}\int \{y(\mathbf{x}_n + \boldsymbol{\xi}) - t_n\}^2 \nu(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}. \tag{2.133}$$

Using the calculus of variations, we can optimize w.r.t the function $f(\mathbf{x})$ to give

$$y(\mathbf{x}_n) = \sum_{n=1}^{N} t_n h(\mathbf{x} - \mathbf{x}_n) \tag{2.134}$$

where the basis functions are given by

$$h(\mathbf{x} - \mathbf{x}_n) = \frac{\nu(\mathbf{x} - \mathbf{x}_n))}{\sum_{n=1}^{N}\nu(\mathbf{x} - \mathbf{x}_n)}. \tag{2.135}$$

This is known as *Nadaraya-Watson* model. If the noise $\boldsymbol{\xi}$ is isotropic, so that it is a function only of $\|\boldsymbol{\xi}\|$, then the basis functions will be radial.

Drawback: Because there is one basis function associated with every data point, the corresponding model can be computationally costly to evaluate when making predictions for new data points.

Nadaraya-Watson model

Assume a Parzen density estimator to model the joint distribution $p(\mathbf{x}, t)$, so that :

$$p(\mathbf{x}, t) = \frac{1}{N}\sum_{n=1}^{N} f(\mathbf{x} - \mathbf{x}_n, t - t_n) \tag{2.136}$$

$f(\cdot)$ is the component density function. Then the expression for the regression function $y(\mathbf{x})$ is given by

$$y(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int_{-\infty}^{\infty} tp(t|\mathbf{x})\mathrm{d}t \tag{2.137}$$

$$= \frac{\int tp(\mathbf{x}, t)\mathrm{d}t}{\int p(\mathbf{x}, t)\mathrm{d}t} \tag{2.138}$$

$$= \frac{\sum_n \int tf(\mathbf{x} - \mathbf{x}_n, t - t_n)\mathrm{d}t}{\sum_m \int f(\mathbf{x} - \mathbf{x}_m, t - t_m)\mathrm{d}t} \tag{2.139}$$

By assuming

$$\int_{-\infty}^{\infty} f(\mathbf{x}, t) t \mathrm{d}t = 0, \tag{2.140}$$

we can obtain

$$y(\mathbf{x}) = \sum_n k(\mathbf{x}, \mathbf{x}_n) t_n \tag{2.141}$$

where the kernel function $k(\mathbf{x}, \mathbf{x}_n)$ is given by

$$k(\mathbf{x}, \mathbf{x}_n) = \frac{g(\mathbf{x} - \mathbf{x}_n)}{\sum_m g(\mathbf{x} - \mathbf{x}_m)} \tag{2.142}$$

and we have defined

$$g(\mathbf{x}) = \int_{-\infty}^{\infty} f(\mathbf{x}, t) \mathrm{d}t. \tag{2.143}$$

2.6 Gaussian Processes

Here we want to approximate a function $y = f(\mathbf{x})$ be a linear function in the form

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \phi(\mathbf{x}) \tag{2.144}$$

where $\phi(\mathbf{x})$ is a vector composed by M fixed basis functions $\phi_j(\mathbf{x})$, $j = 1, 2, \ldots, M$. Then we assume the prior distribution over $\mathbf{w}$:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}). \tag{2.145}$$

Then given training data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with targets $\{t_1, \ldots, t_N\}$. We denote $\mathbf{y} = (y_1, \ldots, y_2)^{\mathrm{T}}$, $y_n = y(\mathbf{x}_n)$., then

$$\mathbf{y} = \Phi \mathbf{w} \tag{2.146}$$

where ϕ is a matrix with elements $\Phi_{nk} = \phi_k(\mathbf{x}_n)$. So we have $\mathbb{E}[\mathbf{y}] = \mathbf{0}$, $\mathrm{cov}[\mathbf{y}\mathbf{y}^{\mathrm{T}}] = \frac{1}{\alpha}\Phi\Phi^{\mathrm{T}} = \mathbf{K}$, where $K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\phi^{\mathrm{T}}(\mathbf{x}_n)\phi(\mathbf{x}_m)$

2.6.1 Gaussian process for regression

For regression problem, we assume our sampling target points t_n is obtained by adding a noise ϵ to the true target number y_n:

$$t_n = y_n + \epsilon. \tag{2.147}$$

So

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}) \tag{2.148}$$

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}) \tag{2.149}$$

Then we can obtain the marginal distribution of $\mathbf{t}$:

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y}) p(\mathbf{y}) \mathrm{d}\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \tag{2.150}$$

where

$$\mathbf{C}_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}. \tag{2.151}$$

Although we obtain this result via $k(\mathbf{x}_n, \mathbf{x}_m) = \frac{1}{\alpha}\phi^{\mathrm{T}}(\mathbf{x}_n)\phi(\mathbf{x}_m)$, now we can discard this to make our model more flexible. Usually we choose the kerel function $k(\cdot, \cdot))$ as

$$k(\mathbf{x}_n, \mathbf{x}_m) = \theta_0 \exp\{-\frac{\theta_1}{2}\|\mathbf{x}_n - \mathbf{x}_m\|^2\} + \theta_2 + \theta_3 \mathbf{x}_n^{\mathrm{T}} \mathbf{x}_m. \tag{2.152}$$

Then we want to find the conditional distribution $p(t_{N+1}|\mathbf{t})$. First,

$$p(\mathbf{t}_{N+1}) = \mathcal{N}(\mathbf{t}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1}) \tag{2.153}$$

where $\mathbf{C}_{N+1}$ is defined by 2.151 while N is substituted with $N+1$, and we can partition it in the form

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^{\mathrm{T}} & c \end{pmatrix} \tag{2.154}$$

where $\mathbf{C}_N$ is just defined by 2.151, $\mathbf{k}$ has elements $k(\mathbf{x}_n, \mathbf{x}_{N+1})$ and the scalar $c = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) + \beta^{-1}$. Using knowledge in Gaussian probability, we can obtain the conditional distribution $p(t_{N+1}|\mathbf{t}_N)$ is still Gaussian with mean and variance given by

$$\mathbb{E}[t_{N+1}] = m(\mathbf{x}_{N+1}) = \mathbf{k}^{\mathrm{T}}\mathbf{C}_N\mathbf{t}, \tag{2.155}$$

$$\mathrm{var}[t_{N+1}] = \sigma^2(\mathbf{x}_{N+1}) = c - \mathbf{k}^{\mathrm{T}}\mathbf{C}_N^{-1}\mathbf{k}, \tag{2.156}$$

where these functions is relied on $\mathbf{x}_{N+1}$ through $\mathbf{k}$ because $\mathbf{k}$ is defined by $k(\mathbf{x}_n, \mathbf{x}_{N+1}$.

Note that the only restriction on the kernel function is that the covariance matrix given by 2.151 is positive definite. And if kernel matrix is positive semidefinite, the requirement will certainly be satisfied.

Note that the mean given by 2.155 of the predictive distribution can be written, as a function of $\mathbf{x}_{N+1}$, in the form

$$m(\mathbf{x}_{N+1}) = \sum_{n=1}^{N} a_n k(\mathbf{x}_n, \mathbf{x}_{N+1}) \tag{2.157}$$

where a_n is the n^{th} component of $\mathbf{C}_N^{-1}\mathbf{t}$. Thus, the kernel function depends only on the distance $\|\mathbf{x}_n - \mathbf{x}_m\|$, then we obtain an expansion in radial basis functions.

2.6.2 Gaussian process for classification

Define a Gaussian process on $a(\mathbf{x})$ then transform it using a logistic sigmoid $y = \sigma(a)$, so we obtain a non-Gaussian stochastic process over $y(\mathbf{x})$ where $y \in \{0, 1\}$. Then the probability distribution over the target variable t is then given by the Bernoulli distribution

$$p(t|a) = \sigma(a)^t(1 - \sigma(a))^{1-t}. \tag{2.158}$$

As usual, we denote $\mathbf{x}_1, \ldots, \mathbf{x}_N$, $\mathbf{t}$, then we consider a single test point $\mathbf{x}_{N+1}$. Our goal is to determine the predictive distribution $p(t_{N+1}|\mathbf{t})$, where we have left the conditioning on the input variables implicit. To do this we introduce a Gaussian process prior over the vector $\mathbf{a}_{N+1}$, which has components $a(\mathbf{x}_1), \ldots, a(\mathbf{x}_{N+1})$.

$$p(\mathbf{a}_{N+1}) = \mathcal{N}(\mathbf{a}_{N+1}|\mathbf{0}, \mathbf{C}_{N+1}). \tag{2.159}$$

Unlike the regression case, the covariance matrix no longer includes a noise term because we assume that all of the training data points are correctly labelled. However, for numerical reasons it is convenient to introduce a noise-like term governed by a parameter ν that ensures that the covariance matrix is positive definite. Thus

$$\mathbf{C}(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \nu\delta_{nm} \tag{2.160}$$

where $k(\mathbf{x}_n, \mathbf{x}_m)$ is any positive semidefinite kernel function. And we assume the kernel function is governed by parameters $\boldsymbol{\theta}$.

For two-class problem, it is sufficient to predict $p(t_{N+1} = 1|\mathbf{t}_N)$:

$$p(t_{N+1} = 1|\mathbf{t}_N) = \int p(t_{N+1} = 1|a_{N+1})p(a_{N+1}|\mathbf{t}_N)\mathrm{d}a_{N+1} \tag{2.161}$$

where $p(t_{N+1} = 1|a_{N+1}) = \sigma(a_{N+1})$. The integral is analytically intractable so we want to use a Gaussian distribution to approximate the logistic function. There are generally three approaches:

- variational inference. make use of the local variational bound on the logistic sigmoid.
- expectation propagation.
- Laplace approximation.

2.7 Sparse Kernel

In this section we shall look at kernel-based algorithms that have sparse solutions, so that predictions for new inputs depend only on the kernel function evaluated at a subset of the training data points.
SVM is a decision machine and so does not provide posterior probabilities. **property**: the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum.

2.7.1 Maximum Margin Classifiers

We begin our discussion of SVM by returning to the two-class classification problem using the linear models of the form

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b. \tag{2.162}$$

The training data set comprises N input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$, with corresponding target values $t_1, \ldots, t_N$ where $t_n \in \{-1, +1\}$, and new data points $\mathbf{x}$ are classified according to the sign of $y(\mathbf{x})$. We shall assume that the data is separable in the feature space.

There may exist many solutions that separate the classes exactly, so we should try to find the one that will give the smallest generalization error. Then we define *margin* to be the smallest distance between the decision boundary and any of the samples. In SVM the decision boundary is chosen to be the one for which the margin is maximized.

The perpendicular distance of a point $\mathbf{x}$ from *a hyperplane defined by* $(y\mathbf{x}) = 0$ where $y(\mathbf{x})$ take the form $y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$ is given by $\frac{|y(\mathbf{x})|}{\|\mathbf{w}\|}$. Because we are only interested in solutions that classify all the data points correctly, so $t_n y(\mathbf{x}_n) > 0$ holds for any n. Thus the distance of a point $\mathbf{x}_n$ to the decision surface is given by

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}. \tag{2.163}$$

The margin is given by the perpendicular distance to the closest point $\mathbf{x}_n$ from the data set, and we want to optimize the parameters $\mathbf{w}$ and b in order to maximize this distance:

$$\arg\max_{\mathbf{w},b}\{\frac{1}{\|\mathbf{w}\|}\min_n[t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b)]\} \tag{2.164}$$

where we have taken the factor $\|\mathbf{w}\|$ outside the optimization over n because $\mathbf{w}$ dose not depend on n. Direct solution of this optimization problem would be very complex, and so we shall convert it into an equivalent problem that is much easier to solve. Note that if we make the rescaling $\mathbf{w} \to \kappa\mathbf{w}$ and $b \to \kappa b$, then the distance from any point $\mathbf{x}_n$ to the decision surface is unchanged. So we can set

$$t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b) = 1 \tag{2.165}$$

for the points that is closest to the surface. In this case, for all data points

$$t_n(\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) + b) \geq 1. \tag{2.166}$$

Then the optimization problem simply requires that we maximize $\|\mathbf{w}\|^{-1}$, and we have to solve the optimization problem

$$\arg\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2 \tag{2.167}$$

subject to constraints given by 2.166.

2.7.2 Relevance Vector Machines

Limitation of SVM:

1. the outputs of an SVM represent decisions rather than posterior probabilities.
2. the SVM was originally formulated for two classes, and the extension to $K > 2$ classes is problematic.

And RVM is a Bayesian sparse kernel technique for regression and classification that shares many of the characteristics of the SVM whilst avoiding its principal limitations.

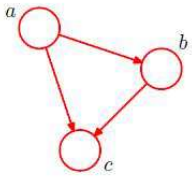


Figure 2.1:

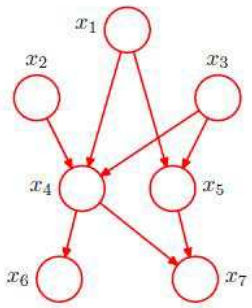


Figure 2.2:

For regression

The most important difference between RVM and SVM is that we introduce a separate hyperparameter α_i for each of the weight w_i instead of a single shared hyperparameters.

For classification

Because the prior distribution for classification problem is not the same as the regression problem, we can not integrate analytically over the parameters. So we should again use Laplace approximation technique to finish the inference.

2.8 Graphical Models

We shall find it highly advantageous to augment the analysis using diagrammatic representations of probability distributions, called *probabilistic graphical models*. These offer several useful properties:

1. They provide a simple way to visualize the structure of a probabilistic model and can be used to design and motivate new models.
2. Insights into the properties of the model, including conditional independence properties, can be obtained by inspection of the graph.
3. Complex computations, required to perform inference and learning in sophisticated models, can be expressed in terms of graphical manipulations, in which underlying mathematical expressions are carried along implicitly.

nodes: random variables. *links*: probabilistic relationships between the variables.
Bayesan networks (directed graphical models), *undirected graphical models*, *factor graph*.

2.8.1 Bayesian Networks

Decomposition of a joint probability distribution:

$$p(a,b,c) = p(c|a,b)p(b|a)p(a) \tag{8.1}$$

and the corresponding graphical model can be seen in figure2.1.
Rule: for each conditional distribution we add directed links (arrows) to the graph from the nodes corresponding to the variables on which the distribution is conditioned.
we can extend the decomposition to K variables:

$$p(x_1,...,x_K) = p(x_K|x_1,...,x_{K-1})...p(x_2|x_1)p(x_1). \tag{2.168}$$

And the corresponding directed graph is called *fully connected* because there is a link between every pair of nodes.
Case Given a directed graph as seen in 2.2, we can write the corresponding probability product:

$$p(x_1)p(x_2)p(x_3)p(x_4|x_1,x_2,x_3)p(x_5|x_2,x_3)p(x_6|x_4)p(x_7|x_4,x_5) \tag{2.169}$$

Rule:

$$p(\mathbf{x}) = \Pi_{k=1}^{K} p(x_k|\mathrm{pa}_k) \tag{2.170}$$

where pa_k denotes the set of parents of x_k, and $\mathbf{x} = \{x_1, .., x_K\}$.
The directed graphs that we are considering are subject to an important restriction namely that there must be no *directed cycles*, in other words there are no closed paths within the graph such that we can move from node to node along links following the direction of the arrows and end up back at the starting node. Such graphs are also called *directed acyclic graphs*, or *DAGs*. This is equivalent to the statement that there exists an ordering of the nodes such that there are no links that go from any node to any lower numbered node.

Example: polynomial regression

For more complex models, we shall adopt the convention that random variables will be denoted by open circles, and deterministic parameters will be denoted by smaller solid circles. some concepts: *observed variables, hidden variables, deterministic parameters.*

Generative models

sampling: given a joint distribution, we want to draw a sample $\hat{x}_1, \hat{x}_2, ..., \hat{x}_K$ from it. **ancestral sampling**
We start with the lowest-numbered node and draw a sample from the distribution $p(x_1)$, which we all $\hat{x}_1$. Then we work through each of the nodes in order, so that for node n we draw a sample from the conditional distribution p(xn—pan) in which the parent variables have been set to their sampled values. Note that at each stage, these parent values will always be available because they correspond to lower-numbered nodes that have already been sampled. The graphical model captures the *causal* process by which the observed data was generated. For this reason, such models are often called *generative* models. But the regression problem is not generative because there is no probability distribution associated with the input variable x, and it is not possible to generate synthetic data points from this model. But we can make it generative by introducing a suitable prior distribution $p(x)$, at the expense of a more complex model.

Discrete variables

parent-child pair in a directed graph. Two cases are particularly worthy of note, namely when the parent and child node each correspond to discrete variables and when they each correspond to Gaussian variables, because in these two cases the relationship can be extended hierarchically to construct arbitrarily complex directed acyclic graphs.
Case: discrete variable $\mathbf{x}$ having K possible states is given by

$$p(\mathbf{x}|\mathbf{mu}) = \mathbf{\Pi_{k=1}^{K}} \mu_{\mathbf{k}}^{\mathbf{x_k}} \tag{2.171}$$

governed by μ. Suppose we have to discrete variables $\mathbf{x_1}, \mathbf{x_2}$, joint distribution can be written as

$$p(\mathbf{x_1}, \mathbf{x_2}|\mu) = \mathbf{\Pi_{k=1}^{K}} \mathbf{\Pi_{k=1}^{K}} \mu_{\mathbf{kl}}^{\mathbf{x_{1k} x_{2l}}}. \tag{2.172}$$

x_{1k} denotes the k^{th} component of $\mathbf{x_1}$.
$K^2 - 1$ parameters. If we have M variables, there is $K^M - 1$ parameters, exponential with the num M.
Suppose $\mathbf{x_1}$ and $\mathbf{x_2}$ are independent. Each variable is then described by a separate multinomial distribution, and total number of parameters will be $2(K-1)$. If extend to M variables, there will be $M(K-1)$ variables, linear growth.
Generally case: more links than independent variables but less links than a fully connected graph.
We can turn a graph over discrete variables into a Bayesian model by introducing Dirichlet priors for the parameters.
Another way of controlling the exponential growth in the number of parameters in models of discrete variables is to use parameterized models for the conditional distributions instead of complete tables of conditional probability values.

Linear-Gaussian models

2.8.2 Conditional Independence

If

$$p(a|b,c) = p(a|c), \tag{2.173}$$

we say that a is conditionally independent of b given c, and we denote this by

$$a \perp\!\!\!\perp b|c.$$

And we will have

$$p(a,b|c) = p(a|c)p(b|c). \tag{2.174}$$

Examples

Three examples related to conditional independence: *tail-to-tail, head-to-tail, head-to-head.*

D-separation

We wish to ascertain whether a particular conditional independence statement

$$A \perp\!\!\!\perp B|C$$

is implied by a given directed acyclic graph.
i.i.d data points. Given μ, we can say the data points $x_1, x_2, ..., x_N$ are conditionally independent, but we can not say the data points are independent. because given x_1, the probability of μ will be affected and then x_2 will be affected.
naive Bayes Model Observation of $\mathbf{z}$ will block the path between x_i and x_j for $j \neq i$. So, if we are given a training set will, comprising inputs $\{x_1, ..., x_N\}$ together with their labels, then we can fit the naive Bayes model to the training data using maximum likelihood assuming that the data are drawn independently from the model.
We can view a directed graph as a filter and all probability distribution $p(\mathbf{x})$ that will be allowed through can make a set $\mathcal{DF}$.
Markov blanket.

$$p(\mathbf{x_i}|\mathbf{x_{\{j\neq i\}}}) = \frac{\mathbf{p(x_1, ...x_D)}}{\int \mathbf{p(x_1, ..., x_D)} \mathrm{d}\mathbf{x_i}} = \frac{\prod_\mathbf{k} \mathbf{p(x_k}|\mathrm{pa_k})}{\int \prod_\mathbf{k} \mathbf{p(x_k}|\mathrm{pa_k}) \mathrm{d}\mathbf{x_i}} \tag{2.175}$$

Some factors in $\prod_k p(\mathbf{x_k}|\mathrm{pa_k})$ will disappear if they are not relative to $\mathbf{x_i}$.

2.8.3 Markov Random Fields

A *Markov random field*, also known as a *Markov network* or an *undirected graphical model* , has a set of nodes each of which corresponds to a variable or group of variables, as well as a set of links each of which connects a pair of nodes. The links are undirected, that is they do not carry arrows.

Conditional independence properties

Properties in an undirected graph is much simpler than those in a directed graph because the nodes linked in a pair is symmetric, and there is no head-to-head situation. So if all paths that connect nodes in A to nodes in B pass through one or more nodes in the set C, then all such paths are 'blocked' and so the conditional independence property hold, aka.

$$A \perp\!\!\!\perp B|C$$

.

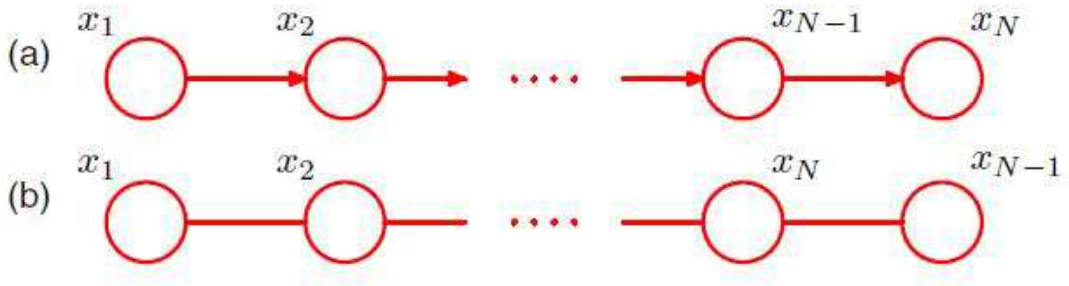


Figure 2.3:

Factorization properties

If two nodes are not linked directly, then we can obtain

$$p(x_i, x_j|\mathbf{x}_{\setminus\{\mathbf{i},\mathbf{j}\}}) = \mathbf{p}(\mathbf{x_i}|\mathbf{x}_{\setminus\{\mathbf{i},\mathbf{j}\}})\mathbf{p}(\mathbf{x_j}|\mathbf{x}_{\setminus\{\mathbf{i},\mathbf{j}\}}) \tag{2.176}$$

clique:a subset of nodes in a graph such that there exists a link between all pairs of nodes in the subset.
maximal clique:a clique such that it is not possible to include any other nodes from the graph in the set without it ceasing to be a clique.
Denote a clique by C and the set of variables in that clique by $\mathbf{x_C}$. Then the joint distribution is written as a product of *potential functions* $\phi_C(\mathbf{x_C})$ over maximal cliques of the graph

$$p(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{Z}}\prod_{\mathbf{C}}\phi_{\mathbf{C}}(\mathbf{x_C}). \tag{2.177}$$

Here the quantity Z, sometimes called the partition function, is a normalization constant and is given by

$$Z = \sum_X \prod_C \phi_C(\mathbf{x_C}) \tag{2.178}$$

energy function $E(\mathbf{x_C})$

$$\phi_C(\mathbf{x_C}) = \exp\{-\mathbf{E}(\mathbf{x_C}\} \tag{2.179}$$

Relation to directed graphs

Given a directed graph as a chain shown in figure2.3(a).

$$p(\mathbf{x}) = \mathbf{p}(\mathbf{x_1})\mathbf{p}(\mathbf{x_2}|\mathbf{x_1})\mathbf{p}(\mathbf{x_3}|\mathbf{x_2})...\mathbf{p}(\mathbf{x_N}|\mathbf{x_{N-1}}) \tag{2.180}$$

Also, for undirected graph shown in figure2.3(b).

$$p(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{Z}}\psi_{\mathbf{1,2}}(\mathbf{x_1}, \mathbf{x_2})\psi_{\mathbf{2,3}}(\mathbf{x_2}, \mathbf{x_3})...\psi_{\mathbf{N-1,N}}(\mathbf{x_{N-1}}, \mathbf{x_N}). \tag{2.181}$$

And we define

$$\psi_{1,2}(x_1, x_2) = p(x_1)p(x_2|x_1) \tag{2.182}$$
$$\psi_{2,3}(x_2, x_3) = p(x_3|x_2) \tag{2.183}$$
$$\vdots \tag{2.184}$$
$$\psi_{N-1,N}(x_{N-1}, x_N) = p(x_N|x_{N-1}) \tag{2.185}$$

2.9 Continuous Latent Variables

The simplest continuous latent variable model assumes Gaussian distributions for both the latent and observed variables and makes use of a linear,Gaussian dependence of the observed variables on Ihe slate of the latent variables. This leads to a probabilistic formulation of the well-known technique of principal component analysis (PCA), as well as to a related model called factor analysis.

2.9.1 Principal Component Analysis

Two commonly used definitions of PCA:

- the orthogonal projection of the data onto a lower dimensional linear space.
- the linear projection that minimizes the average projection cost.

Maximum variance formulation

Consider a data set of observations $\{\mathrm{x}_n\}$ and x_n is a Euclidean variable with dimensionality D. Our goal is to project the data onto a space having dimensionality $M < D$ while maximizing the variance of the projected data.

To begin with, we consider $M = 1$ and we define the one-dimensional space using a D–dimensional vector u_1, which is a unit vector. So the mean of the projected data is $\mathrm{u}_1\bar{\mathrm{x}}$ where

$$\bar{\mathrm{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathrm{x}_n \tag{2.186}$$

And the variance of the projected data is given by

$$\frac{1}{N}\sum_{n=1}^{N}\{\mathrm{u}_1^T\mathrm{x}_n - \mathrm{u}_1^T\bar{\mathrm{x}}\}^2 = (u_1^T S u_1) \tag{2.187}$$

where S is the data covariance matrix2de defined by

$$J(\mathrm{u}_1) = \mathrm{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathrm{x}_n - \bar{\mathrm{x}})(\mathrm{x}_n - \bar{\mathrm{x}})^T \tag{2.188}$$

Then we maximize the $J(\mathrm{u}_1)$ with respect to u_1 under the constraint $\mathrm{u}_1^T\mathrm{u}_1 = 1$ and we obtain

$$\mathrm{S}\mathrm{u}_1 = \lambda_1\mathrm{u}_1 \tag{2.189}$$

which says that u_1 must be an eigenvector of S. And

$$\mathrm{u}_1^{\mathrm{T}}\mathrm{S}\mathrm{u} = \lambda_1, \tag{2.190}$$

so we will choose u_1 to be the eigenvector corresponding to the largest eigenvalue.

Minimum-error formulation

Now we discuss another formulation of PCA based on projection error minimization. To do this, we introduce a complete orthogonal set of D–dimensional basis vectors $\{\mathrm{u}_1\}$ that satisfies

$$\mathrm{u}_{\mathrm{i}}^{\mathrm{T}}\mathrm{u}_{\mathrm{j}} = \delta_{ij}. \tag{2.191}$$

And each data point can be represented by a linear combination of the basis vectors

$$\mathrm{x}_n = \sum_{i=1}^{D}\alpha_{ni}\mathrm{u}_i. \tag{2.192}$$

where $\alpha_{ni} = \mathrm{x}_n^T u_j$. And if we want to reduce the dimensionality to $M < D$, we can approximate each data point x_n by

$$\tilde{\mathrm{x}_n} = \sum_{i=1}^{M} z_{ni}\mathrm{u}_i + \sum_{i=M+1}^{D} bi\mathrm{u}_i \tag{2.193}$$

where z_{ni} depend on the particular point while b_i is fixed for all points. Then we define the loss by

$$J = \frac{1}{N}\sum_{n=1}^{N}\|\mathrm{x}_n - \tilde{\mathrm{x}_n}\|^2. \tag{2.194}$$

Then we minimize J by setting the derivatives with respect to $\{z_{nj}\}$ and b_i to zero, and we obtain

$$z_{nj} = \mathrm{x}_n^T \mathrm{u}_j \tag{2.195}$$

$$b_j = \bar{\mathrm{x}}^T \mathrm{u}_j. \tag{2.196}$$

So the distortion is

$$\mathrm{x}_n - \tilde{\mathrm{x}}_n = \sum_{i=M+1}^{D} \{(\mathrm{x}_n - \bar{\mathrm{x}}_n)^T \mathrm{u}_i\}\mathrm{u}_i \tag{2.197}$$

and

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}(\mathrm{x}_n^T \mathrm{u}_i - \bar{\mathrm{x}}^T \mathrm{u}_i)^2 = \sum_{i=M+1}^{D} \mathrm{u}_i^T \mathrm{S}\mathrm{u}_i. \tag{2.198}$$

Then we need to minimize J with respect to $\{\mathrm{u}_i\}$ under constraint $\mathrm{u}_i^T \mathrm{u}_i = 1$ and we can obtain

$$\mathrm{S}\mathrm{u}_i = \lambda_i \mathrm{u}_i \tag{2.199}$$

and

$$J = \sum_{i=M+1}^{D} \lambda_i. \tag{2.200}$$

Application of PCA

PCA for high-dimensional data

In some situations, $N < D$, a set of N points in D-dimensional space defines a linear subspace whose dimensionality is at most $N-1$. So if we perform PCA we will find that at least $D-N+1$ of the eigenvalues are zero, and the time cost is very large. We can solve this problem as follows. Define X to be the $N \times D$-dimensional centred data matrix, whose n^{th} row is given by $\mathrm{x}_n - \bar{\mathrm{x}}^{\mathrm{T}}$. Then $\mathrm{S} = \frac{1}{N}\mathrm{X}^T\mathrm{X}$ and

$$\frac{1}{N}\mathrm{X}^T\mathrm{X}\mathrm{u}_i = \lambda_i \mathrm{u}_i. \tag{2.201}$$

$$\frac{1}{N}\mathrm{X}\mathrm{X}^T(\mathrm{X}\mathrm{u}_i) = \lambda_i(\mathrm{X}\mathrm{u}_i) \tag{2.202}$$

$$\frac{1}{N}\mathrm{X}\mathrm{X}^T\mathrm{v}_i = \lambda_i \mathrm{v}_i \tag{2.203}$$

where $\mathrm{v}_i = \mathrm{X}\mathrm{u}_i$. And

$$(\frac{1}{N}\mathrm{X}^T\mathrm{X})(\mathrm{X}^T\mathrm{v}_i) = \lambda_i(\mathrm{X}^T\mathrm{v}_i), \tag{2.204}$$

so $(\mathrm{X}\mathrm{v}_i)$ is an eigenvector of S with eigenvalue λ_i.

2.9.2 Probabilistic PCA

Previously we discussed PCA based on a linear projection of the data onto a subspace of lower dimensionality than the original data space. We now show that PCA can also be expressed as the maximum likelihood solution of a probabilistic latent variable model. This reformulation of PCA has several advantages compared with the conventional PCA.
We can formulate probabilistic PCA by first introducing an explicit latent variable $\mathbf{z}$ corresponding to the principal-component subspace. Next we define a Gaussian prior distribution $p(\mathbf{z})$ with a Gaussian conditional distribution $p(\mathbf{x}|\mathbf{z})$.

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}) \tag{2.205}$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mu, \sigma^2\mathbf{I}) \tag{2.206}$$

In a generative view, the D-dimensional observed variable $\mathbf{x}$ is defined by a linear transformation of the M-dimensional latent variable $\mathbf{z}$ plus additive Gaussian noise, so that

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mu + \epsilon \tag{2.207}$$

where $\mathbf{z}$ is an M-dimensional Gaussian latent variable and ϵ is a D-dimensional zero-mean Gaussian-distributed noise variable with covariance $\sigma^2\mathbf{I}$. So, the reverse the mapping, from data space to latent space, will be obtained shortly using Bayes' theorem. Now we want to determine the values of $\mathbf{W}, \mu, \sigma^2$ using maximum likelihood.

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\mathrm{d}\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mu, \mathbf{C}), \tag{2.208}$$

where

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I} \tag{2.209}$$

Note that if we define $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{R}$ where $\mathbf{R}$ is orthogonal, then $\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T = \mathbf{W}\mathbf{W}^T$

$$\mathbf{C}^{-1} = \sigma^{-1}\mathbf{I} - \sigma^{-2}\mathbf{W}\mathbf{M}^{-1}\mathbf{W}^T, \tag{2.210}$$

where $\mathbf{M} \in \mathbb{R}^{M\times M}$ and $\mathbf{M} = \mathbf{W}^T\mathbf{W}$, reducing computing cost by changing reversing $\mathbf{C}$ to reversing $\mathbf{M}$.
So,

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mathbf{M}^{-1}\mathbf{W}^T(\mathbf{x} - \mu), \sigma^{-2}\mathbf{M}) \tag{2.211}$$

where the mean is related to $\mathbf{x}$ while covariance is not.

Maximum likelihood

Given a data set $\mathbf{X} = \{\mathbf{x}_n\}$, then,

$$\ln p(\mathbf{X}|\mu, \mathbf{W}, \sigma^2) = \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\mathbf{W}, \mu, \sigma^2) = -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\mathbf{C}| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \mu)^T\mathbf{C}^{-1}(\mathbf{x}_n - \mu) \tag{2.212}$$

and we obtain

$$\mu_{\mathrm{ML}} = \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \tag{2.213}$$

$$\ln p(\mathbf{x}|\mathbf{W}, \mu, \sigma^2) = -\frac{N}{2}\{D\ln(2\pi) + \ln|\mathbf{C}| + \mathrm{Tr}(\mathbf{C}^{-1}\mathbf{S})\} \tag{2.214}$$

where $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$. Maximizing with respect to $\mathbf{W}$ and σ^2, we find that all stationary points satisfy

$$\mathbf{W}_{\mathrm{ML}} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}\mathbf{R} \tag{2.215}$$

where $\mathbf{U}_M \in \mathbb{R}^{D\times M}$ with columns choosing from the eigenvectors $\mathbf{v}_i$ of $\mathbf{S}$, $\mathbf{L}_M$ offers the eigenvalues λ_i, and $\mathbf{R}$ is any orthogonal matrix. And maximum is reached when we choose the most largest M eigenvalues and their corresponding eigenvectors. Then corresponding value for σ^2 is

$$\sigma^2_{\mathrm{ML}} = \frac{1}{D-M}\sum_{i=M+1}^{D}\lambda_i \tag{2.216}$$

which is unchanged by rotation in the latent variable.
For $\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$, on direction $\mathbf{v}, \mathbf{v}^T\mathbf{v} = 1$, if $\mathbf{v}$ is orthogonal to the subspace, which is shown as $\mathbf{v}^T\mathbf{U} = 0$, then $\mathbf{v}^{\mathbf{C}}\mathbf{v} = \mathbf{v}^T\mathbf{W}\mathbf{W}^T\mathbf{v} + \sigma^2\mathbf{v}^T\mathbf{v} = \sigma^2$, in which, we omitted $\mathbf{W}^T\mathbf{v} = \mathbf{R}^T[(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}]^T\mathbf{U}_M\mathbf{v} = 0$.
If $\mathbf{v} = \mathbf{u}_i \in \mathbb{R}^D$, is an eigenvector, then $\mathbf{v}^T\mathbf{W}\mathbf{W}^T\mathbf{v} = \lambda_i$.
If we choose $M = D$, no reduction will be achieved and $\mathbf{U}_M = \mathbf{U}, \mathbf{L}_M = \mathbf{L}, SS = \mathbf{U}\mathbf{L}\mathbf{U}^T$, therefore, $\mathbf{C} = \mathbf{S}$.
Probabilistic PCA is most natural expressed as a mapping from the latent space into the data space $\mathbf{x} = \mathbf{W}\mathbf{x} + \mu + \epsilon$.

EM algorithm for PCA

In very high dimensional space, although there is close-formed solution for PCA, the computing is very time-consuming, so we turn to using EM algorithm to so PCA to reduce the time costing. We write down the complete-data log likelihood and take its expectation with respect to the

posterior distribution of the latent distribution evaluated using 'old' parameter values. Maximization of this expected comple-data likelihood then yields the 'new' ;parameter values.

$$\ln p(\mathbf{X}, \mathbf{Z}|\mu, \mathbf{W}, \sigma^2) = \sum_{n=1}^{N} \{\ln p(\mathbf{x}_n|\mathbf{z}_n) + \ln p(\mathbf{z}_n)\} \tag{2.217}$$

where the n$^{\text{th}}$ row of the matrix $\mathbf{Z}$ is given by $\mathbf{z}_n$. We can compute μ_{ML} using 2.213 and it is convenient to substitute for μ at this stage.

$$\mathbb{E}[\ln p(\mathbf{X}, \mathbf{Z}|\mu, \mathbf{W}, \sigma^2)] = -\frac{n=1}{N}\{\frac{D}{2}\ln(2\pi\sigma^2) + \frac{1}{2}\text{Tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T]) \tag{2.218}$$

$$+ \frac{1}{2\sigma^2}\|\mathbf{x}_n - \mu\|^2 - \frac{1}{\sigma^2}\mathbb{E}[\mathbf{z}_n]^T\mathbf{W}^T(\mathbf{x}_n - \mu) \tag{2.219}$$

$$+ \frac{1}{2\sigma^2}\text{Tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T]\mathbf{W}^T\mathbf{W})\} \tag{2.220}$$

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{M}^{-1}\mathbf{W}^T(\mathbf{x}_n - \bar{\mathbf{x}}) \tag{2.221}$$

$$\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] = \sigma^2\mathbf{M}^{-1} + \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^T \tag{2.222}$$

where $\mathbf{M} = \mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I}$. In the M step, we maximize with respect to $\mathbf{W}$ and σ^2, keeping the posterior statistics fixed. Maximization with respect to σ^2 is straightforward, and

$$\mathbf{W}_{\text{new}} = [\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})\mathbb{E}[\mathbf{z}_n]^T][\sum_{n=1}^{N}\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T]]^{-1} \tag{2.223}$$

$$\sigma^2_{\text{new}} = \frac{1}{ND}\sum_{n=1}^{N}\{\|\mathbf{x}_n - \bar{\mathbf{x}}\|^2 - 2\mathbb{E}[\mathbf{z}_n]^T\mathbf{W}_{\text{new}}^T(\mathbf{x}_n - \bar{\mathbf{x}}) + \text{Tr}(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T]\mathbf{W}_{\text{new}}^T\mathbf{W}_{\text{new}})\} \tag{2.224}$$

Bayesian PCA

Factor analysis

The only difference between FA and PCA is that we assume

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \mu, \Psi) \tag{2.225}$$

in FA, where Ψ is diagonal. Now, the marginal distribution of $\mathbf{x}$ is $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu, \mathbf{C}), \mathbf{C} = \mathbf{W}\mathbf{W}^T + \Psi$. When we do maximum likelihood, μ_{ML} is still the sample mean, while there is no close-formed solution for $\mathbf{W}$ and we must turn to EM algorithm. And here, the sufficient statistics are

$$\mathbb{E}[\mathbf{z}_n] = \mathbf{G}\mathbf{W}^T\Psi^{-1}(\mathbf{x}_n - \bar{\mathbf{x}}) \tag{2.226}$$

$$\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^T] = \mathbf{G} + \mathbb{E}[\mathbf{z}_n]\mathbb{E}^T[\mathbf{z}_n] \tag{2.227}$$

$$\tag{2.228}$$

where $\mathbf{G} = (\mathbf{I} + \mathbf{W}^T\Psi^{-1}\mathbf{W})^{-1}$.

2.9.3 Kernel PCA

In this section, we will apply the technique of kernel to PCA.
Suppose we have a data set of N observations $\{\mathbf{x}_n\}$ and $\sum_{n=1}^{N}\mathbf{x}_n$, $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^T$

$$\mathbf{S}\mathbf{u}_i = \lambda_i\mathbf{u}_i, \mathbf{u}_i^T\mathbf{u}_i = 1. \tag{2.229}$$

Now consider a nonlinear transformation $\phi(\mathbf{x})$ into an M-dimensional feature space, so that each data point $\mathbf{x}_n$ is thereby projected onto a point $\phi(\mathbf{x}_n)$. For the moment, let us assume that the projected data set also has zero mean, so that $\sum_{n=1}^{N}\phi(\mathbf{x}_n) = \mathbf{0}$. The $M \times M$ sample covariance matrix in feature space is given by

$$\mathbf{C} = \frac{1}{N}\sum_{n=1}^{N}\phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T \tag{2.230}$$

and its eigenvectors is defined by

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \ \mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j = \delta_{ij} \tag{2.231}$$

$i = 1, \ldots, M$. We want to solve the eigenvalue problem without having to work explicitly in the feature space. For the definition of $\mathbf{C}$, the eigenvectors $\mathbf{v}_i$ satisfies

$$\frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_n)\{\phi(\mathbf{x}_n)^T\mathbf{v}_i\} = \lambda_i \mathbf{v}_i \tag{2.232}$$

and we see (provided $\lambda_i > 0$) that the vector $\mathbf{v}_i$ is given by a linear combination of the $\phi(\mathbf{x}_n)$ and so can be written in the form

$$\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n). \tag{2.233}$$

Substituting this expansion back into the eigenvector equation, we obtain

$$\frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}_n)\phi(\mathbf{x}_n)^T \sum_{m=1}^{N} a_{im}\phi(\mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}\phi(\mathbf{x}_n). \tag{2.234}$$

Multiply both sides by $\phi(\mathbf{x}_l)^T$ and we obtain

$$\frac{1}{N}\sum_{n=1}^{N} k(\mathbf{x}_l, \mathbf{x}_n) \sum_{m=1}^{N} a_{im}k(\mathbf{x}_n, \mathbf{x}_m) = \lambda_i \sum_{n=1}^{N} a_{in}k(\mathbf{x}_l, \mathbf{x}_n). \tag{2.235}$$

This can be written in matrix notation as

$$\mathbf{K}^2\mathbf{a}_i = \lambda_i N \mathbf{K}\mathbf{a}_i \tag{2.236}$$

where $\mathbf{a}_i$ is an N-dimensional column vector with elements a_{ni} for $n = 1, \ldots, N$. We can find solutions for $\mathbf{a}_i$ by solving:

$$\mathbf{K}\mathbf{a}_i = \lambda_i N \mathbf{a}_i. \tag{2.237}$$

We require the eigenvectors in feature space be normalized.

$$\mathbf{v}_i^T\mathbf{v}_i = \sum_{n=1}^{N}\sum_{m=1}^{N} a_{in}a_{im}\phi(\mathbf{x}_n)^T\phi(\mathbf{x}_m) = \mathbf{a}_i\mathbf{K}\mathbf{a}_i = \lambda_i N \mathbf{a}_i^T\mathbf{a}_i. \tag{2.238}$$

Then the projection of a point $\mathbf{x}$ onto the eigenvector i is given by

$$y_i(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{v}_i = \sum_{n=1}^{N} a_{in}\phi(\mathbf{x})^T\phi(\mathbf{x}_n) = \sum_{n=1}^{N} a_{in}k(\mathbf{x}, \mathbf{x}_n). \tag{2.239}$$

The dimension M of the feature space can be much larger than D and thus we can find a number of nonlinear principal components that can exceed D. However, the number of nonzero eigenvalues cannot exceed the number N of data points.

2.9.4 Nonlinear Latent Variable Methods

2.10 FOR NEW

Chapter 3

Algorithms

3.1 Nonparametric Methods

In probability theory, we have focussed on the use of probability distributions having specific functional forms governed by a small number of parameters whose values are to be determined from a data set.**This is called the parametric approach**. **to density modelling histogram**

$$p_i = \frac{n_i}{N\Delta_i} = \frac{n_i}{N\Delta}$$

how to choose proper Δ is very important and sensitive

3.1.1 Kernel density estimators

Consider some small region $\mathcal{R}$ containing x, the probability mass is given by

$$P = \int_{\mathcal{R}} p(\mathrm{x})\mathrm{dx}$$

N observations drawn from $p(\mathrm{x})$
Total number K of points that lie inside $\mathcal{R}$ will be distributed according to the binomial distribution.

$$\mathrm{Bin}(K|N,P) =$$

so mean faction in the region will be $\mathrm{E}[K/N] = P$, and $\mathrm{var}[K/N] = P(1-P)/N$.
$P = p(\mathrm{x})V$ if $\mathcal{R}$ is small enough, so $p(\mathrm{x}) = \frac{K}{NV}$
based on two contradictory assumptions: region is sufficiently small that the density is approximately constant over the region and yet sufficiently large that the number K of points falling inside the region is sufficient for the binomial distribution to be sharply peaked.
If we fix V and determine K from the data, giving rise to the kernel approach. $k(\mathrm{u}) = 1$ if $|u_i| \leq 1/2, i = 1, 2, \ldots, D$, else 0 .
This represents a unit cube centred on the origin. The function $k(\mathrm{u})$ is an example of a *kernel function*
So

$$K = \sum_{n=1}^{N} k(\frac{\mathrm{x}-\mathrm{x}_n}{h}), p(\mathrm{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}k(\frac{\mathrm{x}-\mathrm{x}_n}{h})$$

and common choice for a kernel is Gaussian, which give rise to the following kernel density model:

$$p(\mathrm{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{1/2}}\exp\{-\frac{||\mathrm{x}-\mathrm{x}_n||^2}{2h^2}\}$$

also a trade-off between sensitivity to noise at small h and over-smoothing at large h
computational cost.

3.1.2 Nearest-neighbour methods

idea: allow the radius of the sphere to grow until it contains precisely K data points, then the volume of the sphere is set to V.
K-nearest-neighbour technique for density estimation can be extended to classification. N_k points in class $\mathcal{C}_k$, $\sum_k N_k = N$
If we want to classify a new point x, we draw a sphere centred on x containing precisely K points. Suppose this sphere contains K_k points from $\mathcal{C}_k$. Then

$$p(\mathrm{x}|\mathcal{C}_k) = \frac{K_k}{N_k V}, p(\mathrm{x}) = \frac{K}{NV}, p(\mathcal{C}_k) = \frac{N_k}{N}$$

so $p(\mathcal{C}_k|\mathrm{x}) = \frac{K_k}{K}$

3.2 Inference Algorithms in Graphical Models

3.2.1 Inference on a chain

Consider the graph shown in figure2.3. The joint distribution for this graph takes the form

$$p(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{Z}}\psi_{\mathbf{1,2}}(\mathbf{x_1}, \mathbf{x_2})\psi_{\mathbf{2,3}}(\mathbf{x_2}, \mathbf{x_3})...\psi_{\mathbf{N-1,N}}(\mathbf{x_{N-1}}, \mathbf{x_N}). \tag{3.1}$$

If we want to get marginal distribution $p(x_n)$ for a specific node, we should sum the joint distribution over all variables except x_n, so that

$$p(x_n) = \sum_{x_1} ... \sum_{x_{n-1}} \sum_{x_{n+1}} ... \sum_{x_N} p(\mathbf{x}) \tag{3.2}$$

But this way will cost so much computational resources, so we want to find another simpler way:

$$p(x_n) = \frac{1}{Z}[\sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n) \dots [\sum_{x_2} \psi_{2,3}(x_2, x_3)[\sum_{x_1} \psi_{1,2}(x_1, x_2)]] \dots] \tag{3.3}$$

$$[\sum_{x_{n+1}} \psi_{n,n+1}(x_n, x_{n+1}) \dots [\sum_{x_N}] \psi_{N-1,N}(x_{N-1}, x_N)] \dots] \tag{3.4}$$

$$= \frac{1}{Z}\mu_\alpha(x_n)\mu_\beta(x_n) \tag{3.5}$$

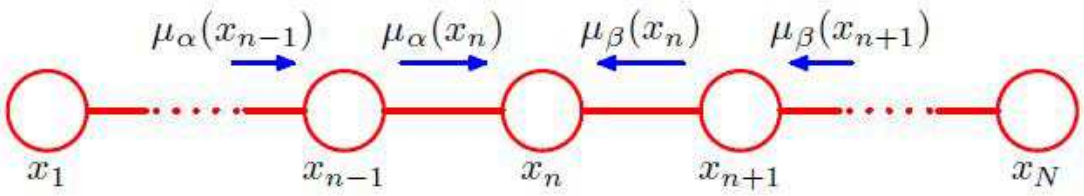
As shown in figure3.2.1, we can see message passed in the chain and we can compute $\mu_\alpha(x_i)$ and $\mu_\beta(x_i)$ recursively for any $i = 1, 2, \dots, N$.

$$\mu_\alpha(x_n) = \sum_{x_{n-1}} \psi_{n-1,n}(x_{n-1}, x_n)\mu_\alpha(x_{n-1}). \tag{3.6}$$

and

$$\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2). \tag{3.7}$$

Similar for $\mu_\beta(x_n)$



3.2.2 Inference on a tree: sum-product algorithm

Tree do not have loops.

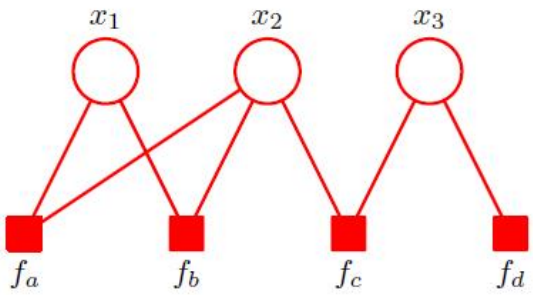


Figure 3.1: Example of a factor graph

factor graphs

In a factor graph, there is a node (depicted as usual by **a circle**) for every variable in the distribution, as was the case for directed and undirected graphs. There are also additional nodes (depicted by **small squares**) for each factor fs(xs) in the joint distribution.
For example,

$$p(\mathbf{x}) = \mathbf{f_a}(\mathbf{x_1}, \mathbf{x_2})\mathbf{f_b}(\mathbf{x_1}, \mathbf{x_2})\mathbf{f_c}(\mathbf{x_2}, \mathbf{x_3})\mathbf{f_d}(\mathbf{x_3}) \tag{3.8}$$

can be expressed by the factor graph shown in figure3.1 Factor graphs are said to be bipartite because they consist of two distinct kinds of nodes, and all links go between nodes of opposite type.

sum-product algorithm

We shall now make use of the factor graph framework to derive a powerful class of efficient, exact inference algorithms that are applicable to tree-structured graphs. Here we shall focus on the problem of evaluating local marginals over nodes or subsets of nodes, which will lead us to the sum-product algorithm.
Our goal is to exploit the structure of the graph to achieve two things: (i) to obtain an efficient, exact inference algorithm for finding marginals; (ii) in situations where several marginals are required to allow computations to be shared efficiently. **finding the marginal $p(x)$ for a particular variable node x**

$$p(\mathbf{x}) = \prod_{\mathbf{s} \epsilon \mathbf{ne(x)}} \mathbf{F_s}(\mathbf{x}, \mathbf{X_s})$$

$$p(x) = \prod_{s \epsilon ne(x)} [\sum_{X_s} F_s(x, X_s)] = \prod_{s \epsilon ne(x)} \mu_{f_s \to x}(x)$$

$$F_s(x, X_s) = f_s(x, x_1, ...x_M) G_1(x, X_{s1}) ... G_M(x, X_{sM})$$

$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm})$$

We have therefore introduced two distinct kinds of message, those that go from factor nodes to variable nodes denoted $\mu_{f \to x}(x)$, and those that go from variable nodes to factor nodes denoted $\mu_{x \to f}(x)$. **Illustration: example max-sum algorithm** Two other common tasks are to find a setting of the variables that has the largest probability and to find the value of that probability. We therefore seek an efficient algorithm for finding the value of x that maximizes the joint distribution p(x) and that will allow us to obtain the value of the joint distribution at its maximum.

3.2.3 Exact inference in a general graph

Goal: deal with graphs having loops.
Loopy belief progpagation One simple approach to approximate inference in graphs with loops, which builds directly on the previous discussion of exact inference in trees. The idea is simply to apply the sum-product algorithm even though there is no guarantee that it will yield good results. For some graphs, the algorithm will converge, whereas for others it will not.

We will say that a (variable or factor) node a has a message pending on its link to a node b if node a has received any message on any of its other links since the last time it send a message to b.

3.3 Mixture Models and EM

This section is mainly about chap.9 in **PRML**.
We can use mixture models to cluster data, as well as build more complex probability distributions. Therefore we begin our discussion of mixture distributions by considering the problem of finding clusters in a set of data points. Then we introduce the latent variable view of mixture distributions in which the discrete variables can be interpreted as defining assignments of data points to specific component of the mixture. Finally we discuss EM in some generality.

3.3.1 K-means Clustering

Suppose we have a data set $\{\mathrm{x}_1, \cdots, \mathrm{x}_N\}$ consisting of N observations of a random D–dimensional Euclidean variable x. Our goal is to partition the data set into some number K of clusters. Firstly we consider a fixed value of K. We can formalize this notion by first introducing a set of D–dimensional vectors μ_k, where $k = 1, \cdots, K$, in which μ_k is a prototype associated with the k^{th} cluster. For each data point x_n, we introduce a corresponding set of binary indicator variables $r_{nk} \in \{0, 1\}$, describing which cluster x_n is assigned to. We can then define an objective function

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathrm{x}_n - \mu_k\|^2. \tag{3.9}$$

Firstly we choose some initial values for the μ_k. Then in the first phase we minimize J with respect to the r_{nk}, keeping the μ_k fixed. In the second phase we minimize J with respect to the μ_k, keeping r_{nk} fixed. This two-stage optimization is then repeated until convergence. And we obtain

$$r_{nk} = \begin{cases} 1 & \text{if} k = \mathrm{argmin}_{\mathrm{j}} \|\mathrm{x}_n - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

$$\mu_k = \frac{\sum_n r_{nk} \mathrm{x}_n}{\sum_n r_{nk}} \tag{3.11}$$

A direct implementation of the K–means algorithm as discussed here cna be slow because in each E step it is necessary to compute the Euclidean distance between every prototype vector and every data point. Various schemes have been proposed for speeding up this algorithm.

3.3.2 Mixtures of Gaussians

Recall from previous chapter the Gaussian mixture distribution can be written as linear superposition of Gaussians in the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k). \tag{3.12}$$

Then we can introduce a K–dimensional binary random variable $\mathbf{z}$ having a 1-of-K representation in which a particular element z_k is equal to 1 and all other elements are equal to 0. So $z_k \in \{0, 1\}$, $\sum_k z_k = 1$, and $p(z_k = 1) = \pi_k$. So

$$p(\mathrm{z}) = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{3.13}$$

Similarly, the conditional distribution of x given a particular value for z is a Gaussian

$$p(\mathrm{x}|z_k = 1) = \mathcal{N}(\mathrm{x}|\mu_k, \Sigma_k). \tag{3.14}$$

So the marginal distribution of x is then obtained:

$$p(\mathrm{x}) = \sum_{\mathrm{z}} p(\mathrm{z}) p(\mathrm{x}|\mathrm{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathrm{x}|\mu_k, \Sigma_k) \tag{3.15}$$

And we can gain the conditional probability of z given x using Bayes' theorem

$$\gamma(z_k) \equiv p(z_k = 1|\mathrm{x}) = \frac{p(z_k = 1)p(\mathrm{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathrm{x}|z_j = 1)} = \frac{\pi_k \mathcal{N}(\mathrm{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathrm{x}|\mu_j, \Sigma_j)}. \tag{3.16}$$

Maximum likelihood

Suppose we have a data set of observations $\{\mathrm{x}_1, \cdots, \mathrm{x}_N\}$, and we wish to model this data using a mixture of Gaussians. Denote this data set as an $N \times D$ matrix X in which the n^{th} row is given by x_n^T, latent variables as an $N \times K$ matrix Z with rows z_n^T. Assume i.i.d observations, we can obtain

$$\ln p(\mathrm{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathrm{x}_n|\mu_k, \Sigma_k)\}. \tag{3.17}$$

Here, we may encounter a significant problem about singularity.

EM for Gaussian mixtures

Setting the derivatives of $\ln p(\mathrm{X}|\pi, \mu, \Sigma)$ with respect to the means μ_k of the Gaussian components to zero, we obtain

$$0 = -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathrm{x}_n|\mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathrm{x}_n|\mu_j, \Sigma_j)} \Sigma_k(\mathrm{x}_n - \mu_k). \tag{3.18}$$

We assume Σ_k to be nonsingular so

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\mathrm{x}_n \tag{3.19}$$

where we have defined

$$N_k \equiv \sum_{n=1}^{N} \gamma(z_{nk}) \tag{3.20}$$

We see that the mean μ_k for the k^{th} Gaussian component is obtained by taking a weighted mean of all of the points in the data set, in which the weighting factor for data point x_n is given by the posterior probability $\gamma(z_{nk})$ that component k was responsible for generating x_n.

If we set the derivative of $\ln p(\mathrm{X}|\pi, \mu, \Sigma)$ with respect to Σ_k to zero , we obtain

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathrm{x}_n - \mu_k)(\mathrm{x}_n - \mu_k)^T. \tag{3.21}$$

Finally we maximize $\ln p(\mathrm{X}|\pi, \mu, \Sigma)$ with respect to π_k using a Lagrange multiplier and we obtain

$$\pi_k = \frac{N_k}{N} \tag{3.22}$$

We must emphasize that these results 3.19, 3.21, 3.22 do not constitute a close-form solution for the parameters of the mixture model because $\gamma(z_{nk})$ depend on those parameters. But we can use the EM algorithm in a simple iteration:

1. Initialize the means μ_k, covariances Σ_k and mixing coefficients π_k, and evaluate the initial value of the log likelihood.

2. **E step.** Evaluate the responsibilities using the current parameter values using 3.16

3. **M step.** Re-estimate the parameters using the current responsibilities using 3.19, 3.21,**??**.

4. Evaluate the log likelihood

$$\ln p(\mathrm{X}|\mu, \Sigma, \pi) = \sum_{n=1}^{N} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathrm{x}_n|\mu_k, \Sigma_k)\} \tag{3.23}$$

3.3.3 An Alternative View of EM

In this section, we present a complementary view of the EM algorithm that recognizes the key role played by latent variables.

Denote:

- X: all observed data, n^{th} row represents x_n^T.
- Z: all latent variables, n^{th} row represents z_n^T.
- θ: all model parameters.

So the log likelihood function is given by

$$\ln p(\mathrm{X}|\theta) = \ln\{\sum_{\mathrm{z}} p(\mathrm{X},\mathrm{Z}|\theta)\}. \tag{3.24}$$

In practice, we are given only incomplete data set X rather than complete $\{\mathrm{X},\mathrm{Z}\}$, so our state of knowledge of the latent variables in Z is given only by the posterior distribution $p(\mathrm{Z}|\mathrm{X},\theta)$.

So the general EM algorithm is summarized below. Given a joint distribution $p(\mathrm{X},\mathrm{Z}|\theta)$ over observed variables X and latent variables Z, governed by parameters θ, the goal is to maximize the likelihood function $p(\mathrm{X}|\theta)$ with respect to θ.

1. Choose an initial setting for the parameters θ^{old}.
2. **E step** Evaluate $p(\mathrm{Z}|\mathrm{X},\theta^{\mathrm{old}})$.
3. **M step** Evaluate θ^{new} given by

$$\theta^{\mathrm{new}} = \mathrm{argmax}_\theta \mathcal{Q}(\theta,\theta^{\mathrm{old}}) \tag{3.25}$$

   where

$$\mathcal{Q}(\theta,\theta^{\mathrm{old}}) = \sum_{\mathrm{z}} p(\mathrm{Z}|\mathrm{X},\theta^{\mathrm{old}}) \ln p(\mathrm{X},\mathrm{Z}|\theta). \tag{3.26}$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

$$\theta^{\mathrm{old}} \leftarrow \theta^{\mathrm{new}} \tag{3.27}$$

   and return to step 2.

Gaussian mixtures revisited

Consider the problem of maximizing the likelihood for the complete data set $\{\mathrm{X},\mathrm{Z}\}$

$$p(\mathrm{X},\mathrm{Z}|\mu,\Sigma,\pi) = \prod_{n=1}^{N}\prod_{k=1}^{K} \pi_k^{z_{nk}} \mathcal{N}(\mathrm{x}_n|\mu_k,\Sigma_k)^{z_{nk}} \tag{3.28}$$

where z_{nk} denotes the k^{th} component of z_n. Taking logarithm and

$$\ln p(\mathrm{X},\mathrm{Z}|\mu,\Sigma,\pi) = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\{\ln \pi_k + \ln \mathcal{N}(\mathrm{x}_n|\mu_k,\Sigma_k)\}. \tag{3.29}$$

Consider first the maximization with respect to the means and covariances.

$$\pi_k = \frac{1}{N}\sum_{n=1}^{N} z_{nk} \tag{3.30}$$

so that the mixing coefficients are equal to the fractions of data points.

$$\mathrm{E}[z_{nk}] = \gamma(z_{nk} \tag{3.31}$$

$$\mathrm{E}_{\mathrm{z}}[\ln p(\mathrm{X},\mathrm{Z}|\mu,\Sigma,\pi)] = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk})\{\ln \pi_k + \ln \mathcal{N}(\mathrm{x}_n|\mu_k,\Sigma_k)\}. \tag{3.32}$$

Relation to K-means

3.3.4 The EM Algorithm in General

EM algorithm is a general technique for finding maximum likelihood solutions for probabilistic models having latent variables. Here we give a very general treatment of the EM algorithm.

Consider, observed variables X, latent variables Z and parameters θ. Our goal is to maximize the likelihood function given by

$$p(\mathrm{X}|\theta) = \sum_{\mathrm{Z}} p(\mathrm{X}, \mathrm{Z}|\theta). \tag{3.33}$$

Suppose that direct optimization of $p(\mathrm{X}|\theta)$ is difficult, but that optimization of the complete-data likelihood function $p(\mathrm{X}, \mathrm{Z}|\theta)$ is easier. Next we introduce a distribution $q(\mathrm{Z})$ defined over the latent variables, and

$$\ln p(\mathrm{X}|\theta) = \mathcal{L}(q, \theta) + \mathrm{KL}(q\|p) \tag{3.34}$$

where

$$\mathcal{L}(q, \theta) = \sum_{\mathrm{Z}} q(\mathrm{Z}) \ln\{\frac{p(\mathrm{X}, \mathrm{Z}|\theta)}{q(\mathrm{Z})}\} \tag{3.35}$$

$$\mathrm{KL}(q\|p) = -\sum_{\mathrm{Z}} q(\mathrm{Z}) \ln\{\frac{p(\mathrm{Z}|\mathrm{X}, \theta)}{q(\mathrm{Z})}\}. \tag{3.36}$$

Suppose the current value of the parameter vector is θ^{old}. In the E step, the lower bound $\mathcal{L}(q, \theta^{old})$ is maximized with respect to $q(\mathrm{Z})$ while holding θ^{old} fixed.

In the subsequent M step, the distribution $q(\mathrm{Z})$ is held fixed and the lower bound $\mathcal{L}(q, \theta)$ is maximized with respect to θ to give some new value θ^{new}.

3.4 Sampling Methods

The fundamental problem that we want to address in this section is finding the expectation of some function $f(\mathbf{z})$ w.r.t a probability distribution $p(\mathbf{z})$. Sometimes it is hard to integrate analytically. So our idea is to obtain a set of samples $\mathbf{z}^{(l)}$, $l = 1, \ldots, L$ drawn from a specific distribution. Then we can approximate the true expectation

$$\mathbb{E}[f] = \int f(\mathbf{z}) p(\mathbf{z}) \mathrm{d}\mathbf{z} \tag{3.37}$$

by a finite sum

$$\hat{f} = \frac{1}{L} \sum_{l=1}^{L} f(\mathbf{z}^{(l)}. \tag{3.38}$$

As long as the samples are drawn from the distribution $p(\mathbf{z})$ independently, then $\mathbb{E}[f] = \mathbb{E}[\hat{f}]$ and the variance of the estimator $\hat{f}$ is given by

$$\mathrm{var}[\hat{f}] = \frac{1}{L} \mathbb{E}[(f - \mathbb{E}[f])^2]. \tag{3.39}$$

So the accuracy of the estimator therefore does not depend on the dimensionality of z, and that, in principle, high accuracy may be achievable with a relatively small number of samples $\mathbf{z}^{(l)}$.

3.4.1 Basic sampling algorithms

Here we shall assume that we have algorithms to generate *pseudo-random numbers* distributed uniformly over (0,1).

Standard distribution

Suppose that z is uniformly distributed over (0,1), and that we transform the values of z using some function $f(\cdot)$ so that $y = f(z)$, then the distribution of y is will be governed by

$$p(y) = p(z)|\frac{\mathrm{d}z}{\mathrm{d}y}| = |\frac{\mathrm{d}z}{\mathrm{d}y}|. \tag{3.40}$$

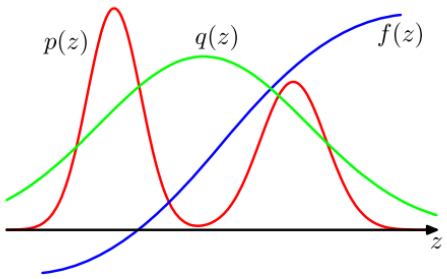


Figure 11.8 Importance sampling addresses the problem of evaluating the expectation of a function $f(z)$ with respect to a distribution $p(z)$ from which it is difficult to draw samples directly. Instead, samples $\{z^{(l)}\}$ are drawn from a simpler distribution $q(z)$, and the corresponding terms in the summation are weighted by the ratios $p(z^{(l)})/q(z^{(l)})$.

Our goal is to choose the function $f(\cdot)$ such that the resulting values of y have some specific desired distribution $p(y)$. Obviously we have

$$z = f^{-1}(y) = \int_{-\infty}^{y} p(\hat{y})\mathrm{d}\hat{y}. \tag{3.41}$$

So we choose $f(\cdot)$ to be the inverse function of the indefinite integral of $p(y)$.

Rejection sampling

Suppose $p(z)$ is easily evaluated for any given z up to some normalizing constant Z, so that

$$p(z) = \frac{1}{Z_p}\tilde{p}(z) \tag{3.42}$$

where $\tilde{p}(z)$ can be easily evaluated while Z_p is unknown. We propose some simpler distribution $q(z)$, sometimes called *proposal distribution*, from which we can easily drawn samples. Then set a constant k such that $kq(z) \geq \tilde{p}(z)$ for all values of z. We first generate a number $z_0 \sim q(z)$, next we generate a number $u_0 \sim \mathrm{U}[0, kq(z_0)]$. Finally, if $u_0 > p\tilde{p}(z_0)$ then the sample is rejected, otherwise u_0 is retained. Thus the remaining pairs then have uniform distribution under the curve of $\tilde{p}(z)$, and hence the corresponding z values are distributed according to $p(z)$.

The probability of accepting the sample is

$$p(\text{accept}) = \int \{\tilde{p}(z)/kq(z)\}q(z)\mathrm{d}z = \frac{1}{k}\int \tilde{p}(z)\mathrm{d}z. \tag{3.43}$$

So we want k to be as big as possible subject to the limitation that $kq(z) \geq \tilde{p}(z)$ for any z.

Adaptive rejection sampling

An adaptive method for finding *proposal distribution*.

Importance sampling

The technique is to approximate the expectation directly while not provide a mechanism for drawing samples from distribution $p(\mathbf{z})$.

We can express the expectation in the form of a finite sum over samples $\{\mathbf{z}^{(l)}\}$ drawn from $q(\mathbf{z})$

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})\mathrm{d}\mathbf{z} = \int f(\mathbf{z})\frac{p(\mathbf{z})}{q(\mathbf{z})}q(\mathbf{z})\mathrm{d}\mathbf{z} \simeq \frac{1}{L}\sum_{l=1}^{L}\frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})}f(\mathbf{z}^{(l)}). \tag{3.44}$$

The quantities $r_l = p(\mathbf{z}^{(l)})/q(\mathbf{z}^{(l)})$ are known as *importance weights*.

Some more things in sampling

3.4.2 Markov Chain Monte Carlo

The advantage of MCMC is that it allows sampling from a large class of distributions and scales well with the dimensionality of the sample space.

Here, we maintain a record of the current state $\mathbf{z}^{(l)}$ and the proposal distribution $q(\mathbf{z}^{(l+1)}|\mathbf{z}^{(l)})$ and so the sequence of samples $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots$ forms a Markov chain. The proposal distribution is simple enough to be sampled. At each cycle of the algorithm, we generate a new candidate sample $\mathbf{z}^*$ from proposal distribution and then accept it according to some criterion.

In the basic *Metropolis* algorithm, we assume $q(\mathbf{z}_A|\mathbf{z}_B) = q(\mathbf{z}_B|\mathbf{z}_A)$. Then the candidate sample is accepted with probability

$$A(\mathbf{z}^*, \mathbf{z}^{(\tau)}) = \min(1, \frac{\tilde{p}(\mathbf{z}^*)}{\tilde{p}(\mathbf{z}^{(\tau)})}. \tag{3.45}$$

This can be achieved by choosing a random number u with uniform distribution over the unit interval $(0, 1)$ and then accepting the sample if $A(\mathbf{z}^*, \mathbf{z}^{(\tau)} > u$. Note that if $\tilde{p}(\mathbf{z}^*) > \tilde{p}(\mathbf{z}^{(\tau)})$, then the candidate point is certain to be kept.

If the candidate is kept then $\mathbf{z}^{(\tau+1)} = \mathbf{z}^*$, otherwise the candidate is discarded and $\mathbf{z}^{(\tau+1)}$ is set to be $\mathbf{z}^{(\tau)}$. Note that the sequence $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots$ is not a set of independent samples from $p(\mathbf{z})$ because they are correlated strongly. But we can obtain independent samples by discarding most of the sequence and just retaining every W^{th} sample.

Markov chain

A distribution is said to be invariant, or stationary, with respect to a Markov chain if each step in the chain leaves that distribution invariant. Thus, for a homogeneous Markov chain with transition probabilities $T(\mathbf{z}, \mathbf{z}')$, the distribution $p^*(\mathbf{z})$ is invariant if

$$p^*(\mathbf{z}) = \sum_{\mathbf{z}'} T(\mathbf{z}', z) p^*(\mathbf{z}') = \sum_{\mathbf{z}'} p(\mathbf{z}|\mathbf{z}') p^*(\mathbf{z}'). \tag{3.46}$$

A sufficient but not necessary condition for ensuring that the required distribution $p(\mathbf{z})$ is invariant is to **choose the transition probabilities** to satisfy the property *detailed balance*, defined by

$$p^*(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = p^*(\mathbf{z}') T(\mathbf{z}', \mathbf{z}). \tag{3.47}$$

for a particular distribution $p^*(\mathbf{z})$. Note that if 3.47 holds, then

$$\sum_{\mathbf{z}'} p^*(\mathbf{z}') T(\mathbf{z}', \mathbf{z}) = \sum_{\mathbf{z}'} p^*(\mathbf{z}) T(\mathbf{z}, \mathbf{z}') = p^*(\mathbf{z}) \sum_{\mathbf{z}'} p(\mathbf{z}'|\mathbf{z}) = p^*(\mathbf{z}). \tag{3.48}$$

So $p^*(\mathbf{z})$ is invariant. A Markov chain that respects detailed balance is said to be *reversible*.

Our goal is to use Markov chains to sample from a given distribution. We can achieve this if we set up a Markov chain such that the desired distribution is invariant. However, we must also require that for $m \to +\infty$, the distribution $p(z^{(m)})$ converges to the required invariant distribution $p^*(\mathbf{z})$, irrespective of the choice of the initial distribution $p(z^{(0)})$. This property is called *ergodicity*, and the invariant distribution is then called the equilibrium distribution. Clearly, an ergodic Markov chain can have only one equilibrium distribution.

Metropolis-Hastings algorithm

Our goal is to use Markov chains to sample from a given distribution. We can achieve this if we set up a Markov chain such that the desired distribution is invariant. So we define a transition probability $q(\mathbf{z}'|\mathbf{z})$ and we wish the detailed balance $p(\mathbf{z})q(\mathbf{z}'|\mathbf{z}) = p(\mathbf{z}')q(\mathbf{z}|\mathbf{z}')$ holds, so we add factors $\alpha(\mathbf{z}, \mathbf{z}') = q(\mathbf{z}|\mathbf{z}')p(\mathbf{z}')$, $\alpha(\mathbf{z}', \mathbf{z}) = q(\mathbf{z}'|\mathbf{z})p(\mathbf{z})$ to both sides:

$$p(\mathbf{z})q(\mathbf{z}'|\mathbf{z})q(\mathbf{z}|\mathbf{z}')p(\mathbf{z}') = p(\mathbf{z}')q(\mathbf{z}|\mathbf{z}')q(\mathbf{z}'|\mathbf{z})p(\mathbf{z}) \tag{3.49}$$

which is sure to be true. So we obtain a new transition probability

$$q'(\mathbf{z}'|\mathbf{z}) = q(\mathbf{z}'|\mathbf{z})q(\mathbf{z}|\mathbf{z}')p(\mathbf{z}'). \tag{3.50}$$

The factor $\alpha(\mathbf{z}, \mathbf{z}')$ is called *accept probability*. When the state on the Markov chain jumps under the control of $q(\mathbf{z}, \mathbf{z}')$, we accept the jump with probability $\alpha(\mathbf{z}, \mathbf{z}')$. So the process of the MCMC algorithm can be described as:

Algorithm 1 MCMC Sampling Method

Require: Target distribution $p(\mathbf{z})$
Ensure: Samples drawn independently from $p(\mathbf{z})$
1: Initialize the state of the Markov chain $X_0 = x_0$
2: For $t = 0, 1, 2, \ldots$, repeat the following steps to sample

- For time slice t, the state is $X_t = x_t$, sample $y \sim q(x|x_t)$
- Sample $u \sim \text{Uniform}[0, 1]$
- if $u < \alpha(x_t, y) = p(y)q(x_t|y)$ the accept the trainsition, and $X_{t+1} = y$
- else don't accept the transition and $X_{t+1} = x_t$

However, sometimes the acceptance probability defined by $\alpha(\mathbf{z}, \mathbf{z}')$ is so small that the efficiency is not ensured, so we want to enlarge the probability. And our idea is to enlarge one of $\alpha(\mathbf{z}, \mathbf{z}'), \alpha(\mathbf{z}', \mathbf{z})$ to be as large as one, so we choose

$$\alpha(\mathbf{z}, \mathbf{z}') = \min\{\frac{p(\mathbf{z}')q(\mathbf{z}|\mathbf{z}')}{p(\mathbf{z})q(\mathbf{z}'|\mathbf{z})}, 1\}. \tag{3.51}$$

And we can see

$$\begin{aligned} p(\mathbf{z})q(\mathbf{z}'|\mathbf{z})\alpha(\mathbf{z}, \mathbf{z}') &= \min\{p(\mathbf{z}')q(\mathbf{z}|\mathbf{z}'), p(\mathbf{z})q(\mathbf{z}'|\mathbf{z})\} \\ &= p(\mathbf{z}')q(\mathbf{z}|\mathbf{z}') \min\{1, \frac{p(\mathbf{z})q(\mathbf{z}'|\mathbf{z})}{p(\mathbf{z}')q(\mathbf{z}|\mathbf{z}')}\} \\ &= p(\mathbf{z}'q(\mathbf{z}|\mathbf{z}')\alpha(\mathbf{z}', \mathbf{z}) \end{aligned} \tag{3.52}$$

and then detailed balance is satisfied.

3.4.3 Gibbs Sampling

Gibbs sampling is a special case of Metropolis-Hastings algorithm.

Consider the distribution $p(\mathbf{z}) = p(z_1, \ldots, z_M)$ from which we want to sample, and suppose we have some initial state for the Markov chain. Each step of the Gibbs sampling procedure involves replacing the value of one of the variables by a value drawn from the distribution of that variable conditioned on the values of the remaining variables: replace z_i by a value drawn from $p(z_i|\mathbf{z}_i$. So the procedure can be described as:

Algorithm 2 Gibbs Sampling algorithm

Require: Target distriibution $p(\mathbf{z})$
Ensure: Samples drawn independently from $p(\mathbf{z})$
Initialize $\{z_i : i = 1, 2, \ldots, M\}$
For $\tau = 1, \ldots, T$:

- Sample $z_1^{(\tau+1)} \sim p(z_1|z_2^{(\tau)}, \ldots, z_M^{(\tau)})$
- Sample $z_2^{(\tau+1)} \sim p(z_2|z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$

  $\vdots$
- Sample $z_j^{(\tau+1)} \sim p(z_j|z_1^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_j^{(\tau)}, \ldots, z_M^{(\tau)})$
- Sample $z_M^{(\tau+1)} \sim p(z_M|z_1^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$