PRML Chapter 1: Introduction

Generalization: The ability to perform accurately on new, unseen examples/tasks after having learned from a set of training examples.

1.1 Example: Polynomial Curve Fitting

1.1.1 Linear Basis Function Models

The simplest linear model for regression is one that involves a linear combination of fixed nonlinear functions of the input variables. This is known as a linear basis function model. The following equation is a linear model of the form:

$$y(x, \textbf{w}) = w_0 + w_1x + w_2x^2 + ... + w_Mx^M = \sum_{j=0}^{M}w_jx^j$$

where x is the input variable, y is the output variable, w is the weight vector, and M is the order of the polynomial. Although the model is nonlinear in the input variable x, it is linear in the parameters $\textbf{w}$.

We need to minimize the error function, which is the sum of the squares of the differences between the target values t and the values predicted by the model $y(x, \textbf{w})$:

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

where N is the number of data points.

So in this problem, we just need to find the optimal values of the weight vector $\textbf{w}$ that minimize the error function $E(\textbf{w})$. And it has a closed-form solution, denoted as $\textbf{w}^*$:

$$\textbf{w}^* = (\textbf{Φ}^T\textbf{Φ})^{-1}\textbf{Φ}^T\textbf{t}$$

where $\textbf{Φ}$ is the design matrix, whose elements are given by $\textbf{Φ}_{nj} = x_n^j$, and $\textbf{t}$ is the vector of target values.

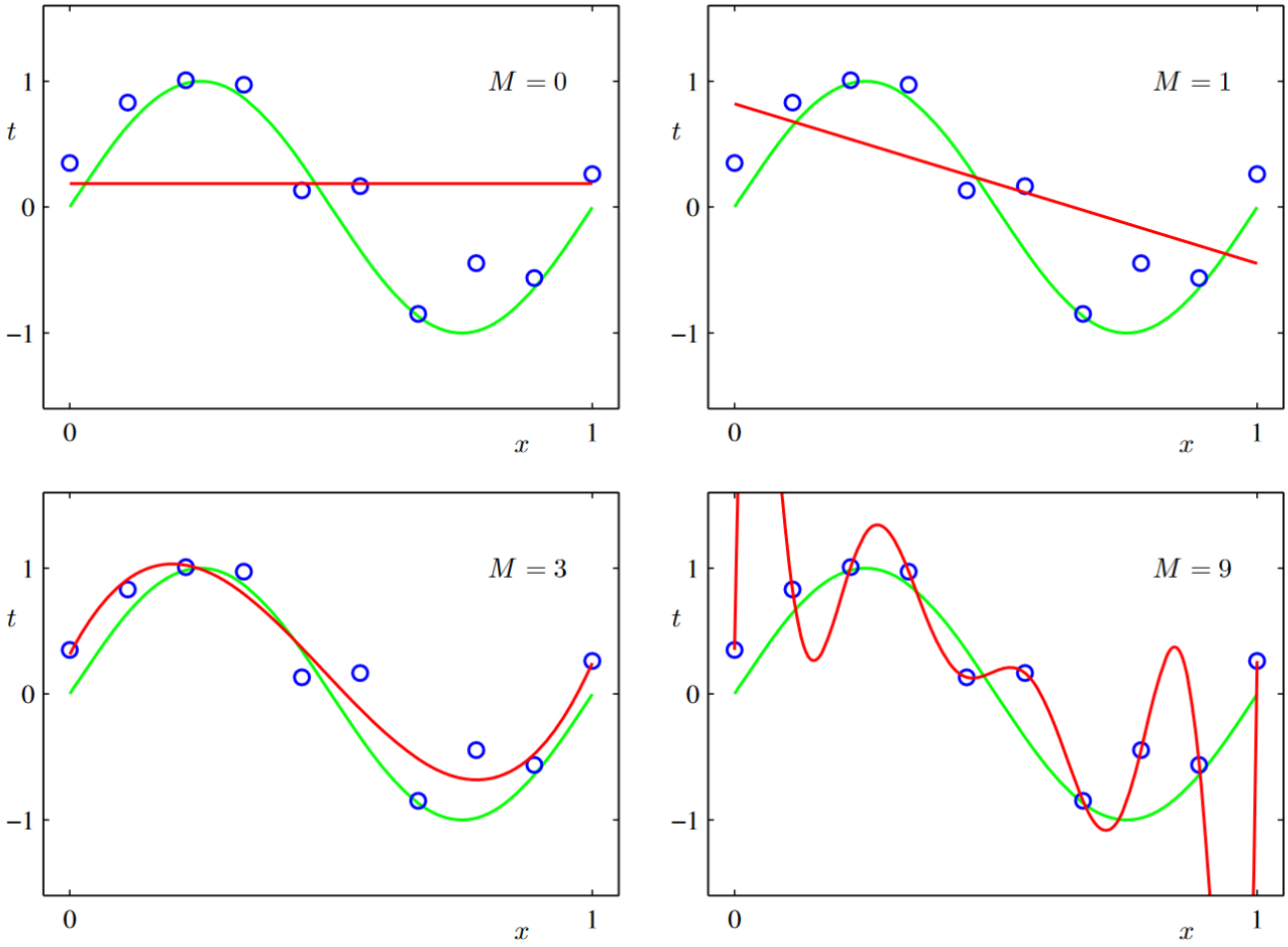
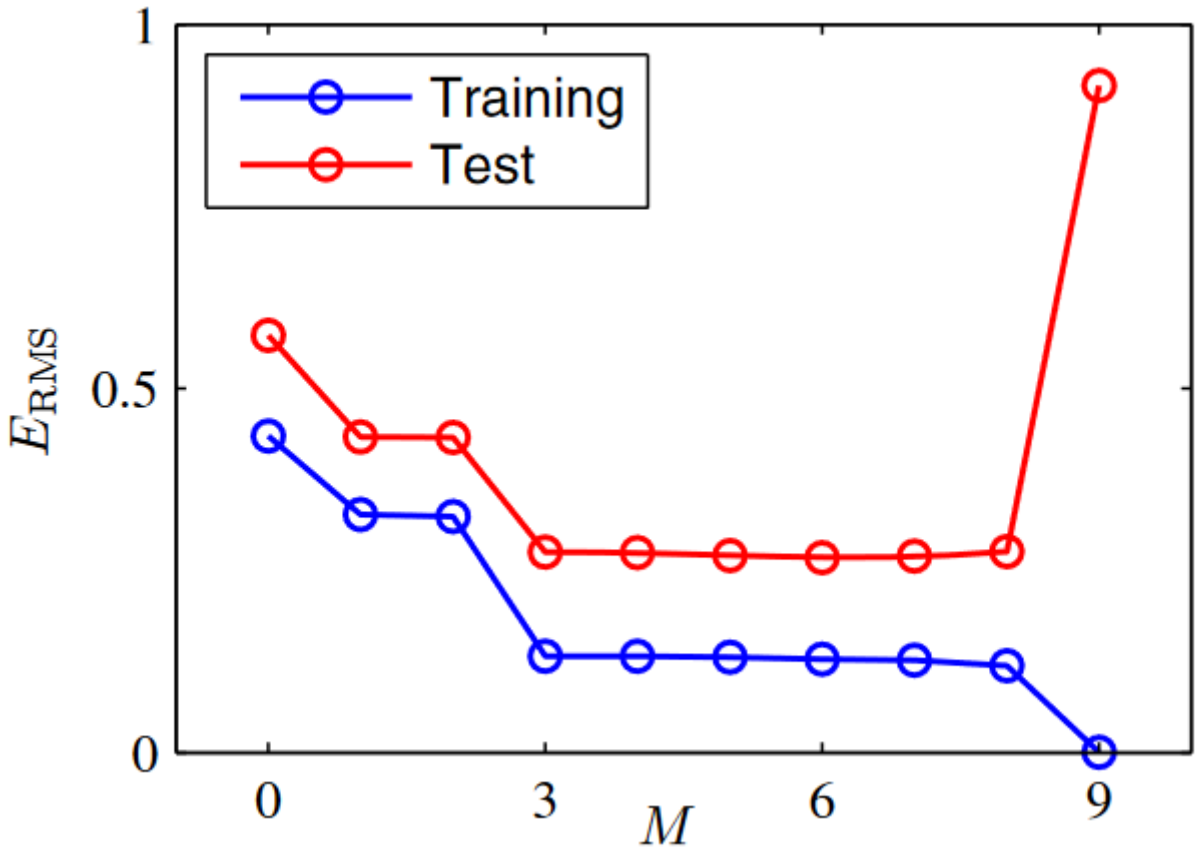


Figure 1 shows different value of polynomial order M and the corresponding polynomial curve fitting. As we can see, the polynomial curve fitting becomes more flexible as the order M increases. However, if we set M too large, the model will overfit the data, which means it will perform well on the training data but poorly on the test data.

Another error measuring equation is RMS, which is the square root of the mean of the squares of the error function:

$$E_{RMS} = \sqrt{2E(\textbf{w}^*)/N}$$



The figure above shows the RMS error as a function of the order M of the polynomial. As we can see, for training data, the RMS error decreases as the order M increases. However, the RMS error on the test data increases as the order M increases, which means the model is overfitting the data.

This is a little paradoxical, but it is a common phenomenon in machine learning. If $M=9$, there are 10 coefficients which means it should be able to fit 10 points exactly. However, as the figure shows, the RMS error on the test data is very large.

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

As the figure above shows, the coefficients of the polynomial model with $M=9$ are very large. So it only perform well on the training data. This a also a proof that we need add a regularization term to the error function to prevent overfitting.

What means regularization? It means we add a term to the error function to penalize the large coefficients. In fact, in the coefficient space, there are a lot of solutions that can minimize the error function. But some of them are too large, it will obviously perform poorly on the unknwon data. So we need to choose the solution that has the smallest coefficients that will have a better generalization ability.

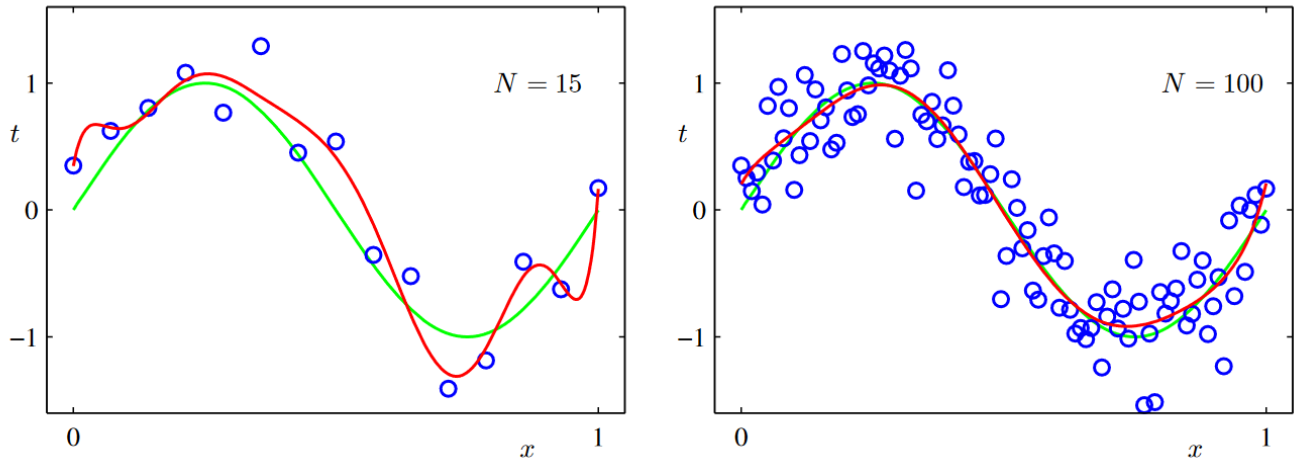
So the regularized error function is:

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

where λ is the regularization coefficient, and $||\textbf{w}||^2$ is the square of the Euclidean norm of the weight vector $\textbf{w}$. From a high level aspect, we consider that we need to minimized the error function, incluing two terms. On the one hand, we need to minimized the original error function which is like MSE. On the other hand, we need to minimized the Euclidean norm of the weight vector. λ is a trade off between the model complexity and the fitting ability.

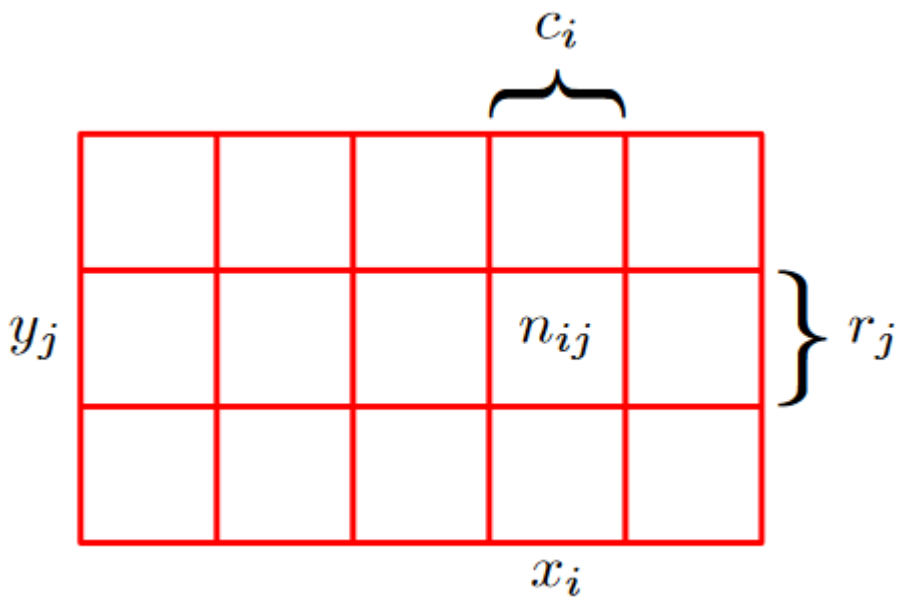
	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

And the figure above shows different λ and the corresponding coefficients of the polynomial model with $M=9$. As we can see, the coefficients become smaller as λ increases. And the RMS error on the test data decreases as λ increases. So the regularization term can prevent overfitting.



Another methods to prevent overfitting is to add more training data. As the figure below shows, if we add more training data, the RMS error on the test data will decrease. Another way to say this is that the larger the training data, the better the generalization ability of the model.

1.2 Probability Theory



Sum rule: $$p(X = x_i) = \sum_{j}p(X = x_i, Y = y_j) = \frac{c_i}{N}$$

Conditional probability: $$p(X = x_i|Y = y_j) = \frac{n_{i,j}}{c_i}=\frac{p(X = x_i, Y = y_j)}{p(Y = y_j)}$$

In conditional probability, $n_{i,j}$ is the number of times $X = x_i$ and $Y = y_j$ occur together, and c_i is the number of times $X = x_i$ occurs. We can say that the core of conditional probability is shrink the sample space. The original sample space is N while the new sample space is c_i. And based on it, we can give the equation of conditional probability as above.

And we can change the equation of conditional probability to the form of joint probability:

$$p(X = x_i, Y = y_j) =\frac{n_{i,j}}{N}=\frac{n_{i,j}}{c_i} \times \frac{c_i}{N} =p(Y = y_j|X = x_i)p(Y = x_i)$$

The equation above is also called product rule. It is the core of probability theory. It is the foundation of the Bayes' theorem.

And we can also get the Bayes' theorem from the product rule:

$$p(Y = y_j|X = x_i) = \frac{p(X = x_i|Y = y_j)p(Y = y_j)}{p(X = x_i)}$$

1.2.1 Probability densities

We call $p(x)$ the probability density function of the continuous variable x. And the probability that x lies in the range $[x, x + \delta x]$ is given by $p(x)\delta x$. And the probability that x lies in the range $[x, x + \delta x]$ and y lies in the range $[y, y + \delta y]$ is given by $p(x, y)\delta x \delta y$. So for the single variable, if we need to calculate the probability that x lies in the range $[a, b]$, we can use the following equation:

$$p(a \leq x \leq b) = \int_{a}^{b}p(x)dx$$

And for the joint probability, we can use the following equation:

$$p(a \leq x \leq b, c \leq y \leq d) = \int_{a}^{b}\int_{c}^{d}p(x, y)dxdy$$

If we need to calculate the probability that x lies in the range $[a, b]$, and y lies in the range $[-\infty, +\infty]$, we can use the following equation:

$$p(a \leq x \leq b) = \int_{a}^{b}\int_{-\infty}^{+\infty}p(x, y)dydx$$

And if the variable x is non-linearly transformed by y,which means $x=g(y)$, we can get the following equation:

$$ p(y) = p(x)\left|\frac{dx}{dy}\right| = p(g(y))\left|\frac{dg(y)}{dy}\right| $$

The equation above is the transformation of the probability density function. And the absolute value of the derivative of the transformation function is the Jacobian of the transformation. It will make sure the probability is conserved and non-negative.

This is the PDF(Probability Density Function) of the continuous variable. And the CDF(Cumulative Distribution Function) is the integral of the PDF. The CDF of the continuous variable x is defined as:

$$P(x) = \int_{-\infty}^{x}p(x')dx'$$

Obviously, the CDF is a monotonically increasing function. And the PDF is the derivative of the CDF. So we can get the PDF from the CDF. And if we need to calculate the probability that x lies in the range $[a, b]$, we can use the following equation:

$$p(a \leq x \leq b) = P(b) - P(a)$$

And it also have:

$$\lim_{x \to -\infty}P(x) = 0$$

$$\lim_{x \to +\infty}P(x) = 1$$

1.2.2 Expectations and Covariances

The expectation and variance of a function $f(x)$ with respect to the probability distribution $p(x)$ are defined as:

$$E[f] = \int f(x)p(x)dx $$

$$E[f] \approx \frac{1}{N}\sum_{n=1}^{N}f(x_n)$$

$$var[f] = E[(f(x) - E[f(x)])^2] = E[f(x)^2] - E[f(x)]^2$$

Expectation is the value to measure the center of the distribution. And variance is the value to measure the spread of the distribution. And the covariance between two variables x and y is defined as:

$$cov[x, y] = E[(x - E[x])(y - E[y])] = E[xy] - E[x]E[y]$$

Which is a measure of the degree to which x and y vary together. If x and y are independent, then $cov[x, y] = 0$. And the covariance matrix(for vectors of varibles) is defined as:

$$cov[\textbf{x}] = E[(\textbf{x} - E[\textbf{x}])(\textbf{x} - E[\textbf{x}])^T]$$

Obviously, accoding to the definition of covariance matrix, it is a symmetric matrix. And the diagonal elements of the covariance matrix are the variances of the corresponding variables. And the off-diagonal

elements are the covariances between the corresponding variables. For the vector $\textbf{x} = [x_1, x_2, ..., x_D]^T$, the covariance matrix is a $\mathbb{M} = D \times D$ matrix. And $\mathbb{M}*{i,j}$ is the covariance between x_i and x_j which is the same as $\mathbb{M}*{j,i}$. The calculation of the covariance matrix is:

$$\mathbb{M}*{i,j} = *frac{1}{N}*sum*{n=1}^{N}(x_{i,n} - \bar{x}*i)(x*{j,n} - \bar{x}_j)$$

where $\bar{x}_i$ is the mean of the variable x_i.

1.2.3 Bayesian Probability Theory

Modern machine learning and pattern recognition are based on the Bayesian probability theory. The Bayesian probability theory is a way to quantify uncertainty. It is a way to measure the degree of belief in a hypothesis. The Bayesian probability theory is based on the Bayes' theorem:

$$p(\textbf{w}|D, M) = \frac{p(D|\textbf{w}, M)p(\textbf{w}|M)}{p(D|M)}$$

where $p(\textbf{w}|D, M)$ is the posterior distribution, $p(D|\textbf{w}, M)$ is the likelihood function, $p(\textbf{w}|M)$ is the prior distribution, and $p(D|M)$ is the evidence. The posterior distribution is the distribution of the parameters $\textbf{w}$ given the data D and the model M. The likelihood function is the probability of the data given the parameters and the model. The prior distribution is the distribution of the parameters before observing the data. The evidence is the probability of the data given the model. The Bayesian probability theory is a way to update the prior distribution to the posterior distribution based on the data. It is a way to quantify the uncertainty of the parameters.

Let's consider the polynomial curve fitting problem. We have the data $D = {(x_1, t_1), (x_2, t_2), ..., (x_N, t_N)}$, and the model M is the polynomial model of order M. The coifficients of the polynomial model are the parameters $\textbf{w} = {w_0, w_1, ..., w_M}$.

The core of the Bayesian probability theory is the posterior distribution. And we will use the equation above to use prior distribution to calculate posterior distribution. Before we observe the data, the prior distribution is the distribution of the coifficients, denoted as $p(\textbf{w}|M)$. And after we observe the data, the posterior distribution is the distribution of the coifficients, denoted as $p(\textbf{w}|D, M)$. The likelihood function is the probability of the data given the parameters and the model, denoted as $p(D|\textbf{w}, M)$. The evidence is the probability of the data given the model, denoted as $p(D|M)$.

The likelihood function is the probability of the data given the parameters and the model:

$$p(D|\textbf{w}, M) = \prod_{n=1}^{N}p(t_n|x_n, \textbf{w}, M)$$

where $p(t_n|x_n, \textbf{w}, M)$ is the probability of the target value t_n given the input value x_n, the parameters $\textbf{w}$, and the model M.

So given the definition of the likelihood function, we can state Bayes' theorem in words:

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

It is hard to understand the meaning of $p(D|\textbf{w}, M)$ which is called likelihood function. Before I thought that the likelihood is meaningless because since the coiffient $\textbf{w}$ is determined, the likelihood is a binary value. I ignore that the problem is we can not determine the coiffient $\textbf{w}$, so the $\textbf{w}$ is a random varible too. So we can consider the polynomial fitting problem in possibility.

The $\textbf{w}$ is a random varible in the M order space, it's a joint distribution of M variable $w_1,w_2,\dots,w_M$. We need to maximize the likelihood function which is maximize the possibility to calculate the value t_n for given input x_n. And it will obviously has the minimized error.

1.2.4 The Gaussian Distribution

Gaussian distribution is the most important distribution in probability theory. It is a bell-shaped curve. The Gaussian distribution is defined as:

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

where μ is the mean of the distribution, and σ^2 is the variance of the distribution.

The vector x is a D-dimensional vector, and the Gaussian distribution is defined as:

$$\mathcal{N}(\textbf{x}|\textbf{μ}, \mathbb{Σ}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\mathbb{Σ}|^{1/2}}\exp{-\frac{1}{2}(\textbf{x} - \textbf{μ})^T\mathbb{Σ}^{-1}(\textbf{x} - \textbf{μ})}$$

where $\textbf{μ}$ is the mean vector, and $\mathbb{Σ}$ is the covariance matrix.

And if each dimension of vector $\textbf{x}$ has the same expectation and variance the possibility of the data given the Gaussian distribution is:

$$p(\textbf{x}|μ, \sigma^2) = \prod_{i=1}^{D}\mathcal{N}(x_i|\mu, \sigma^2)$$

And the log form of the possibility is:

$$\ln p(\textbf{x}|μ, \sigma^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{D}(x_i - μ)^2 - \frac{D}{2}\ln(2\pi\sigma^2)$$

Back to the polynomial curve fitting problem, if we use maximize likelihood methods to solve the problem, there will be bias.

Unbias Estimation: The definition of the unbiased estimation is that the expectation of the estimation is equal to the true value. In Gasussian distribution, the mean of the distribution is the unbiased estimation. While the variance of the distribution is the biased estimation.

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

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

The reason for the bias is that the variance of the distribution is calculated by the mean of the distribution. So the variance of the distribution is the biased estimation. And the unbiased estimation of the variance is:

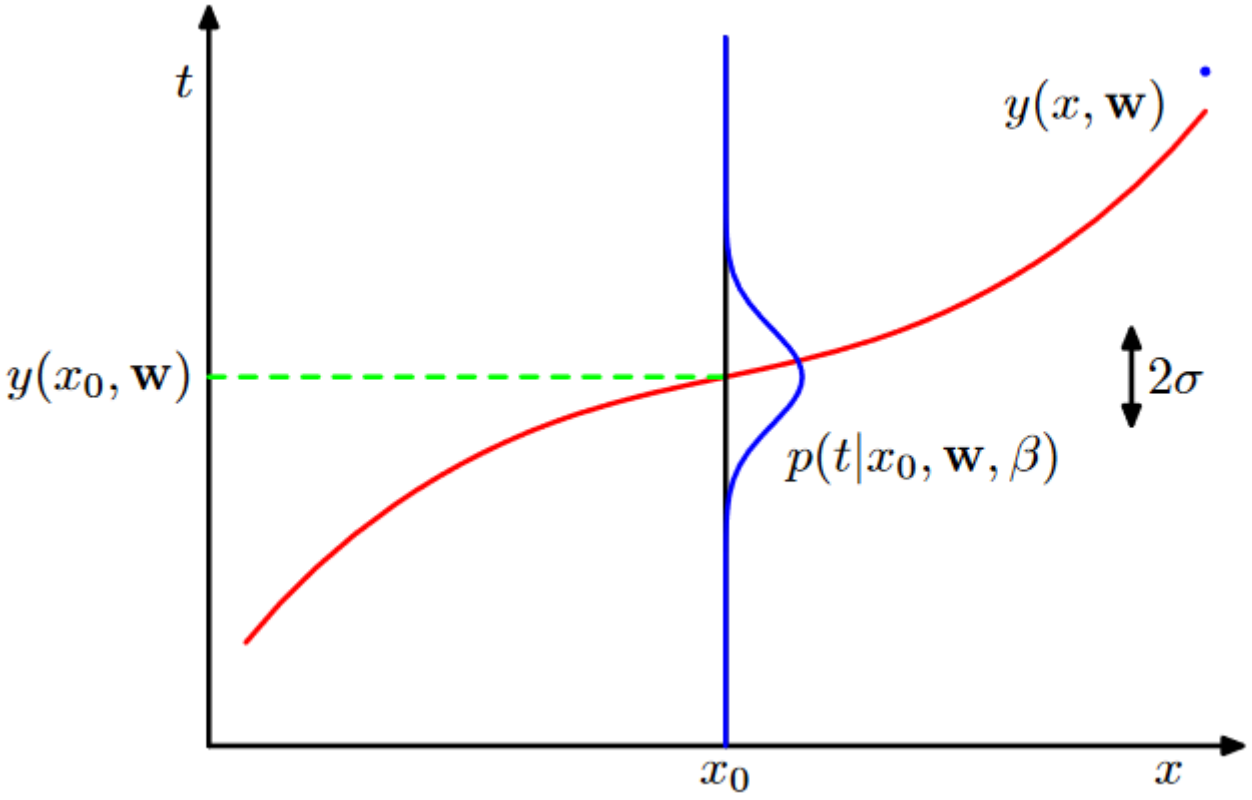
$$\sigma^2 = \frac{N}{N-1} \sigma_{ML}^2 = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \mu_{ML})^2$$

1.2.5 Curve Fitting Revisited

The goal of the curve fitting problem is to find the optimal values of the parameters $\textbf{w}$ that minimize the error function. We believe that $t \sim N(y(x,\textbf{w}),\sigma^2) $. In the Bayesian probability theory, we need to find the posterior distribution of the parameters $\textbf{w}$ given the data D and the model M. The posterior distribution is defined as:

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

where β is inverse variance of the Gaussian distribution. Thus $p(t|x, \textbf{w}, \beta)$ is the likelihood function.



We use MLE, the likelihood function is biased for the variance but unbiased for the mean, so for any given data x_i, and the coiffient we get $\textbf{w}$, we have: $E[p(t|x_i, \textbf{w}, \beta)] = y(x_i, \textbf{w})$.

And if we use the whole training data {$\textbf{x},\textbf{t}$} to calculate the likelihood function, we have:

$$p(\textbf{t}|\textbf{x}, \textbf{w}, \beta) = \prod_{n=1}^{N}\mathcal{N}(t_n|y(x_n, \textbf{w}), \beta^{-1})$$

The log likelihood function is:

$$\ln p(\textbf{t}|\textbf{x}, \textbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}{y(x_n, \textbf{w}) - t_n}^2 + \frac{N}{2}\ln \beta - \frac{N}{2}\ln(2\pi)$$

Consider we use machine learning methods to find a optimal solution $\textbf{w}_{ML}$, we need to maximize the likelihood function. And consider the second term and last term, it is not related to the parameters $\textbf{w}$, so we can ignore them. So the optimization problem is:

$$\textbf{w}*{ML} = \arg\max{\textbf{w}}{-\frac{\beta}{2}\sum_{n=1}^{N}{y(x_n, \textbf{w}) - t_n}^2}$$

And after that, let's consider the precision parameter β, if we find the optimal solution $\textbf{w}_{ML}$, we can get the optimal solution of β:

$$\beta_{ML} = \frac{N}{\sum_{n=1}^{N}{y(x_n, \textbf{w}_{ML}) - t_n}^2}$$

We know that β is the precision parameter of the Gaussian distribution, which is the inverse of the variance. So the larger the β, the smaller the variance of the Gaussian distribution. And the smaller the β, the larger the variance of the Gaussian distribution. So the β is a trade-off between the model complexity and the fitting ability. We need to find the largest β, we can also consider using MLE to find the optimal solution of β. We derive the optimal solution of β by maximizing the likelihood function of β:

$$\beta_{ML} = \arg\max_{\beta}{-\frac{N}{2}\ln \beta - \frac{\beta}{2}\sum_{n=1}^{N}{y(x_n, \textbf{w}_{ML}) - t_n}^2}$$

Taking the derivative of the equation above, we have: $$ \frac{\partial}{\partial \beta} \left(-\frac{\beta}{2} \sum_{n=1}^N {y(x_n, w) - t_n}^2 + \frac{N}{2} \ln \beta \right) = 0 $$

$$ -\frac{1}{2} \sum_{n=1}^N {y(x_n, w) - t_n}^2 + \frac{N}{2\beta} = 0 $$

So we can find the optimal solution of β:

$$\beta_{ML} = \frac{N}{\sum_{n=1}^{N}{y(x_n, \textbf{w}_{ML}) - t_n}^2}$$

1.3 Model Selection

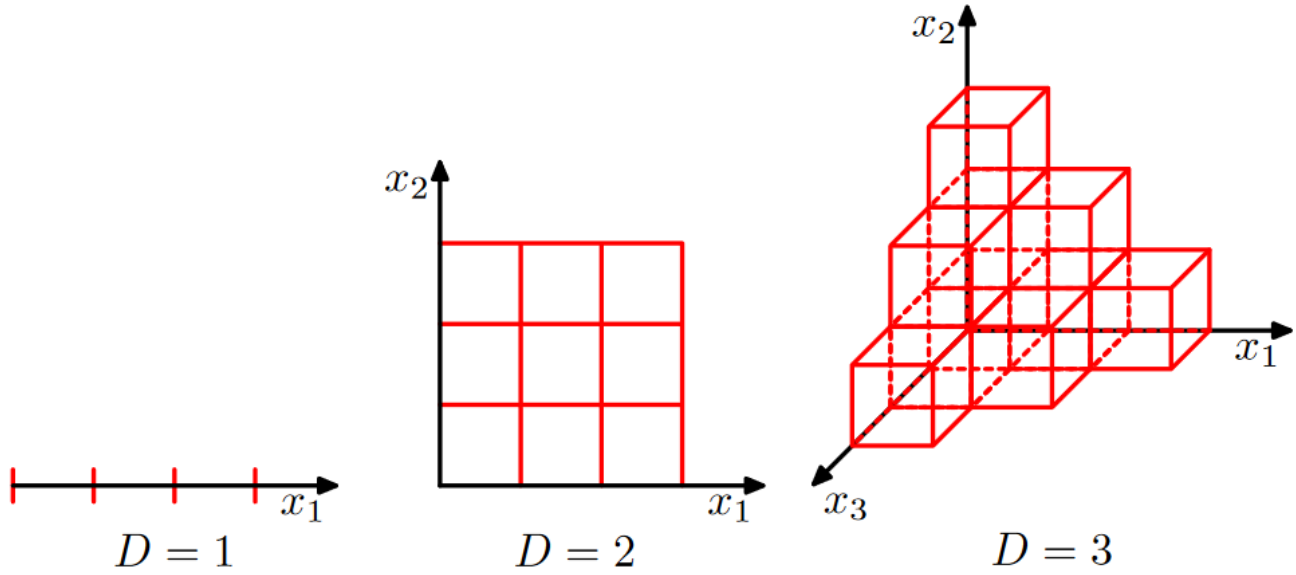
Model Selection is aim to find a best model during training process. We know that there will be much epochs during training and so we have many checkpoints. How to select the best model among thest checkpoints become a important problem. One potential solution is using validated set to testing the model and choose the best one according some metrics such as loss(for regression) or accuracy(for classification). But since the data is valuable, we can not spilt too much data into validated set so there must be some estimation error. It may not represent the real situation. So we need to find a best way to trade off the amount of data in validated set and quality of validation.

K-fold cross validation is a good way to solve this problem. We can split the data into K parts and use $K-1$ parts to train the model and use the rest part to validate the model. And we can repeat this process K times and get K results. We can use the average of the K results to evaluate the model.

This method has a shortcoming, we need to find the best parameter K which is the trade-off between the amount of data in validated set and the quality of validation. And the K-fold cross validation is time-consuming.

1.4 The Curse of Dimensionality

Previous, we talk about curve fitting problem. The input data is only a single variable. But in the real world, the input data is a vector. So the input data is a high-dimensional vector. The main difference between vector and single variable is the amount of dimensions.



The figure above shows different dimensions D visualized by the unit cube. As we can see, the volume of the unit cube decreases as the dimension D increases. Let's consider the polynomial curve fitting problem. What if the input is a vector? If the input dimension is $D = 3$, the polynomial model is:

$$\begin{aligned}y(\mathbf{x},\mathbf{w})=w_0+\sum_{i=1}^Dw_ix_i+\sum_{i=1}^D\sum_{j=1}^Dw_{ij}x_ix_j+\sum_{i=1}^D\sum_{j=1}^D\sum_{k=1}^Dw_{ijk}x_ix_jx_k.\end{aligned}$$

As D increased, the independent terms of the polynomial model increased exponentially. The coiffient of the polynomial model is $O(D^M)$, which means the number of the coiffient is exponential to the dimension D. So the model complexity is exponential to the dimension D.

When we deal with the high-dimension data, we meet a problem that since we live in three-dimensional space, we can not imagine the high-dimension space. So we can not use the geometric intuition to understand the high-dimension space.

And the high-dimension space may very sparse, let's consider the three-dimensional space we live. Assume there is a ball with radius r in the three-dimensional space. The volume of the ball is $V = \frac{4}{3}\pi r^3$. The more you near to the surface of the ball, the bigger volume you will get. So there is a very high density near the surface rather than the center.

1.5 Decision Theory

Decision making is a obvious process. If we understand the possibility, we will quickly know the princeple of decision making. For classification tasks, decision making is that we need a rule to assign each input value $\textbf{x}$ to a class C_k. Such a rule divides the input space into regions R_k called decision regions.

1.5.1 Minimizing the misclassification rate

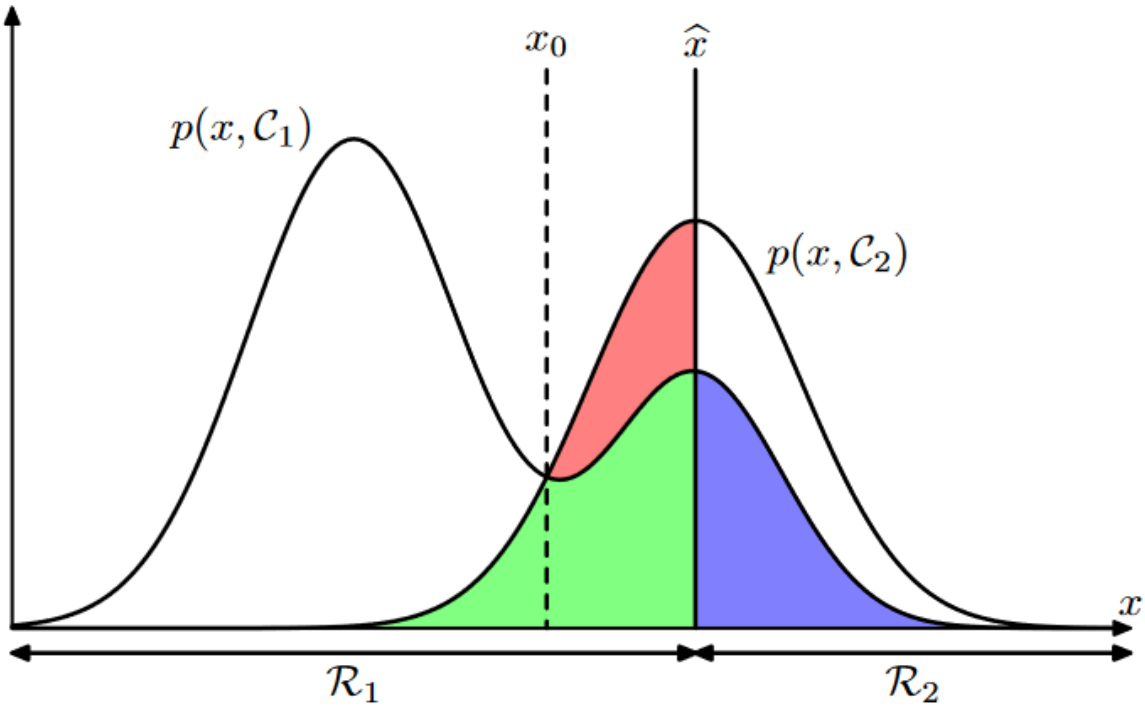
In classification tasks, we need to minimize the probability of misclassification. The probability of misclassification is:

$$p(mistake) = \sum_{k=1}^{K}p(C_k)\int_{R_k}p(x|C_k)dx$$

where $p(C_k)$ is the prior probability of class C_k, $p(x|C_k)$ is the likelihood function of class C_k, and R_k is the decision region of class C_k.

So if we want to minimize the probability of misclassification, we need to consider the value $p(x|C_k)p(C_k)$ which is the posterior probability of class C_k. The decision rule is:

$$\text{Assign } \textbf{x} \text{ to class } C_k \text{ if } p(C_k|\textbf{x}) > p(C_j|\textbf{x}) \text{ for all } j \neq k$$



The image above shows the decision regions of two classes, C_1 and C_2. $\hat{x}$ is the decision boundary. If x_0 is in the region R_1, we assign the input value $\textbf{x}$ to class C_1. If x_0 is in the region R_2, we assign the input value $\textbf{x}$ to class C_2. We can see that is we change the decision boundary, the error will change too. The error consists of two parts, the error of the region R_1 and the error of the region R_2 which is green, red and blue area in the image above. The decision boundary is the trade-off between the error of the region R_1 and the error of the region R_2. The sum of green one and blue one is constant, but we can change the red one. If we set $x_0 = \hat{x}$, the error will be minimized. Then we will use the minimize error possibility to assign the input value $\textbf{x}$ to class C_k.

And for "Loss martix", it is a matrix to measure the cost of misclassification. The diagonal elements of the loss matrix are the cost of correct classification. The off-diagonal elements are the cost of misclassification. The loss matrix is a way to quantify the cost of misclassification. The loss matrix is a way to quantify the cost of misclassification. So if we use this martix to do decision making, the decision rule is:

$$\text{Assign } \textbf{x} \text{ to class } C_k \text{ if } \sum_{j}L_{kj}p(C_j|\textbf{x}) < \sum_{j}L_{jj}p(C_j|\textbf{x})$$

where L_{kj} is the element of the loss matrix. In fact, I believe it is a just weighted sum of the posterior probability of class C_k. But it is a trade off between the cost of misclassification and the cost of correct classification. Sometimes we need to avoid the high cost if misclassification class C_i to C_j, so we will rather to assign the input value $\textbf{x}$ to class C_i.

1.5.2 Minimizing the expected loss

Similar to the loss matrix, the expected loss is a way to quantify the cost of misclassification. The expected loss is the expectation of the loss function:

$$\text{Expected loss} = \int \int L(C_k, \hat{C}|\textbf{x})p(\textbf{x}, C_k)d\textbf{x}dC_k$$

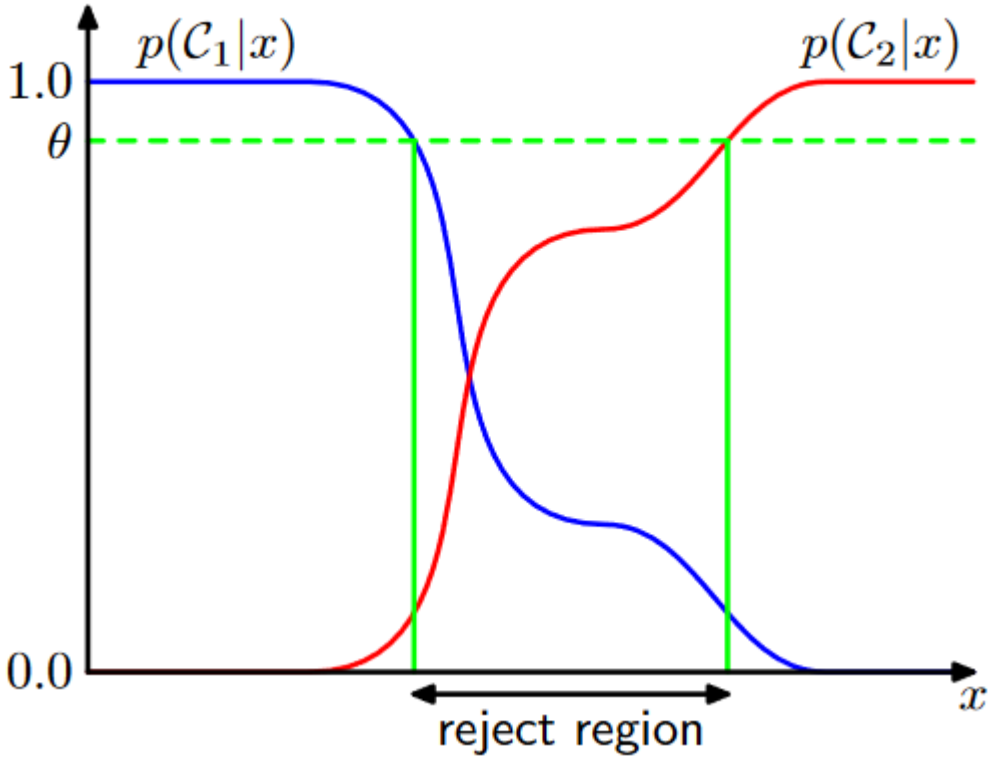
where $L(C_k, \hat{C}|\textbf{x})$ is the loss function, $\hat{C}$ is the predicted class, and C_k is the true class. The expected loss is the expectation of the loss function.

So the question become how to minimize the expected loss. The decision rule is:

$$\text{Assign } \textbf{x} \text{ to class } C_k \text{ if } \sum_{j}L_{kj}p(C_j|\textbf{x}) < \sum_{j}L_{jj}p(C_j|\textbf{x})$$

where L_{kj} is the element of the loss matrix. The decision rule is the same as the loss matrix.

1.5.3 The reject option



The image above shows reject region. We know that the misclassification error is due to the posterior probability of class C_k. If we set a threshold θ, we can reject the input value $\textbf{x}$ if the posterior probability of class C_k is less than the threshold θ. The reject option is a way to reduce the misclassification error. Only the posterior probability of class C_k is greater than the threshold θ, we will assign the input value $\textbf{x}$ to class C_k. Otherwise, we will reject the input value $\textbf{x}$ and leave the decision to the human. The reject option is a way to reduce the misclassification error. So now the decision rule is:

$$\text{Assign } \textbf{x} \text{ to class } C_k \text{ if } \sum_{j}L_{kj}p(C_j|\textbf{x}) < \sum_{j}L_{jj}p(C_j|\textbf{x}) \text{ and } p(C_k|\textbf{x}) > \theta$$

where θ is the threshold. The decision rule is the same as the loss matrix. But we add a threshold θ to reduce the misclassification error.

1.5.4 Inference and decision

We have broken the classification problem into two parts, inference and decision. Inference is the process of estimating the posterior probability of class C_k given the input value $\textbf{x}$. Decision is the process of assigning the input value $\textbf{x}$ to class C_k. So if we need to do classification tasks, we need two stage. What if we combine the two stage into one? Which means we need to learn a single function f that maps the input value $\textbf{x}$ to the class C_k. Such a function is called discriminant function. There are three ways to do classification task.

1. **Generative modeling**: We need to estimate the class-conditional density $p(\textbf{x}|C_k)$ and the prior probability $p(C_k)$. Then we can use the Bayes' theorem to calculate the posterior probability $p(C_k|\textbf{x})$.

$$f(\textbf{x}) = p(C_k|\textbf{x})$$

$$ p(C_k|\textbf{x}) = \frac{p(\textbf{x}|C_k)p(C_k)}{p(\textbf{x})}$$

$$ p(\textbf{x}) = \sum_{j}p(\textbf{x}|C_j)p(C_j)$$

In fact, we explicitly or implicitly learn the joint distribution of the input value $\textbf{x}$ and the class C_k. And we use the Bayes' theorem to calculate the posterior probability of class C_k given. This modeling process is called generative modeling.

2. **Discriminative modeling**: We need to learn the posterior probability $p(C_k|\textbf{x})$ directly. And then use decision rule to assign the input value $\textbf{x}$ to class C_k.

3. **Discriminant function**: We need to learn a single function f that maps the input value $\textbf{x}$ to the class C_k. For example, if we need to do binary classification, we can learn a function f that maps the input value $\textbf{x}$ to the class C_1 or C_2. The decision rule is:

$$\text{Assign } \textbf{x} \text{ to class } C_1 \text{ if } f(\textbf{x}) = 1$$

$$\text{Assign } \textbf{x} \text{ to class } C_2 \text{ if } f(\textbf{x}) = 0$$

For generative modeling and discriminative modeling, there is difference: Even if they all need to calculate the posterior probability of class C_k given the input value $\textbf{x}$, the generative modeling need to calculate the class-conditional density $p(\textbf{x}|C_k)$ and the prior probability $p(C_k)$, while the discriminative modeling need to learn the posterior probability $p(C_k|\textbf{x})$ directly(May use SVM, Regression etc.).

1.6 Information Theory

We need to know how to measure the amount of information. We can say that if we can measure the amount of information by the "degree of surprise". If the event is very likely to happen, the amount of information is very small. If the event is very unlikely to happen, the amount of information is very large which means we have learnt something that we do not know before.

Thus, the amount of information is depended on the possibility of the event. The more likely the event is, the less information we will get. The less likely the event is, the more information we will get. Let's consider a function $h(.)$ that measures the amount of information. If we have two event x_1 and x_2, the amount of information of the event x_1 is $h(x_1)$, and the amount of information of the event x_2 is $h(x_2)$. If the two events are independent, the amount of information of the two events is the sum of the amount of information of the two events:

$$h(x_1, x_2) = h(x_1) + h(x_2)$$

And since the events are independent, the possibility of the two events is the product of the possibility of the two events:

$$p(x_1, x_2) = p(x_1)p(x_2)$$

So $h(x)$ should be given by the logarithm of the probability of the event x:

$$h(x) = -\log p(x)$$

The negative assign is to make sure the amount of information is non-negative. In transmitting the information, since all of the data is stored by the binary code, the amount of information is measured by the bit. This is the reason why the logarithm is base 2. Then we can define the entropy of the event x as the expectation of the amount of information:

☑ = -\sum_x p(x)\log p(x)$$

It can be seen as the quantity of the uncertainty. The smaller the entropy, the less uncertainty. The larger the entropy, the more uncertainty. The maximum entropy is when the possibility of the event is uniform which means we don't know anything about the event. The minimum entropy is when the possibility of the event is 1 which means we know everything about the event.

If the variable is continuous, the entropy is defined as:

☑ = -\int p(x)\log p(x)dx$$

Since we know：

$$ \int p(x)dx = 1 $$

☑ = \mu $$

$$ \int (x-\mu)^2 p(x)dx = E[(x-\mu)^2] = \sigma^2 $$

So we can set these three equations as constraints to maximize the entropy. The Lagrange function is:

$$ L = -\int p(x)\log p(x)dx + \lambda_1(\int p(x)dx - 1) + \lambda_2(\int xp(x)dx - \mu) + \lambda_3(\int (x-\mu)^2 p(x)dx - \sigma^2) $$

Taking the derivative of the Lagrange function, we have:

$$ \frac{\partial L}{\partial p(x)} = -\log p(x) - 1 + \lambda_1 + \lambda_2 x + \lambda_3(x-\mu)^2 = 0 $$

So we can get the optimal solution of $p(x)$:

$$ p(x) = \exp{-1 + \lambda_1 + \lambda_2 x + \lambda_3(x-\mu)^2} $$

And we can get the optimal solution of λ_1, λ_2, and λ_3 by the constraints and the final solution of $p(x)$ is:

$$ p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp{-\frac{(x-\mu)^2}{2\sigma^2}} $$

Which is the Gaussian distribution. The Gaussian distribution is the distribution that maximizes the entropy. And the entropy of the Gaussian distribution is:

☑ = \frac{1}{2}\ln 2\pi e\sigma^2 $$

Suppose we have joint distribution $p(x, y)$, the entropy of the joint distribution is:

$$ H[y|x] = -\sum_{x, y}p(x, y)\log p(y|x) = - \int \int p(x, y)\log p(y|x)dxdy $$

$$ H[x, y] = -\sum_{x, y}p(x, y)\log p(x, y) = \int \int p(x, y)\log p(x, y)dxdy $$

$$ H[x, y] = -\sum_{x, y}p(x, y)\log p(x|y)p(y) = -\int \int p(x, y)\log p(x|y)dxdy - \int \int p(x, y)\log p(y)dxdy $$

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

1.6.1 Relative entropy and mutual information

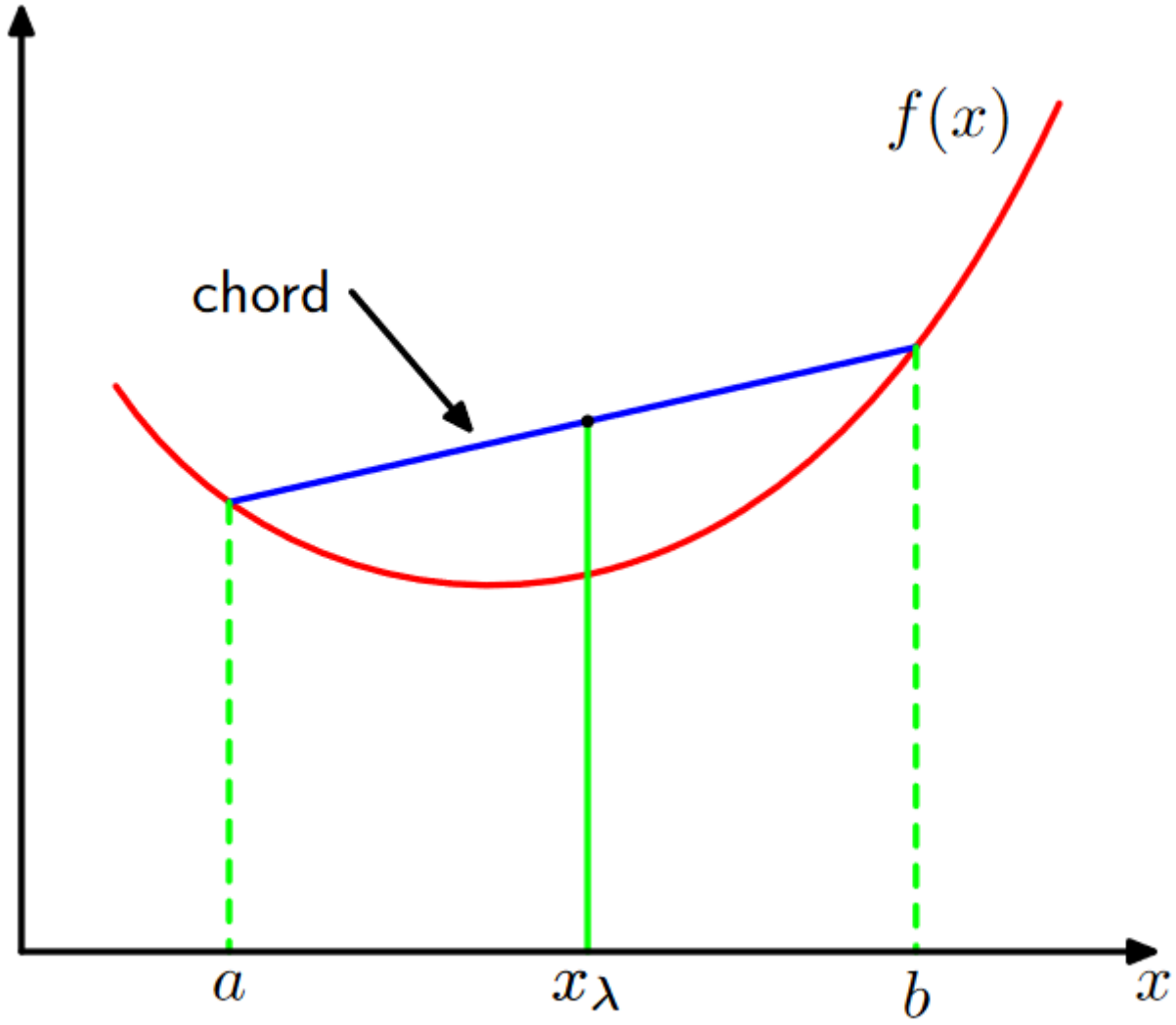
Consider we have modelled a distribution $q(x)$ to approximate the true distribution $p(x)$. If we use $q(x)$ to transmit the information of the event x, we need some addtional bits to transmit the information. The additional bits is the relative entropy between the two distributions:

$$ D_{KL}(p||q) = \sum_x p(x)\log \frac{p(x)}{q(x)} = -\sum_x p(x)\log q(x) + \sum_x p(x)\log p(x) $$

This term is called the Kullback-Leibler divergence. It is a way to measure the difference between the two distributions. Note that it is not a symmetrical quantity, which means $ D_{KL}(p||q) \neq D_{KL}(q||p) $.

A function is convex if for any $x \in [a,b]$ and $0 \leq \lambda \leq 1$, we have:

$$ f(\lambda a + (1-\lambda)b) \leq \lambda f(a) + (1-\lambda)f(b) $$



And if the function has a opposite properties, we call it is concave. And if $f(x)$ is convex, $-f(x)$ is concave.

A convex function will satisfy:

$$ f(\sum_i \lambda_i x_i) \leq \sum_i \lambda_i f(x_i) $$

where $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$. If we consider \sum_i \lambda_i = 1 as the probability, the convex function will satisfy:

$$ f(\sum_i p_i x_i) \leq \sum_i p_i f(x_i) $$

where $0 \leq p_i \leq 1$ and $\sum_i p_i = 1$.

$$ f(\int xp(x)dx) \leq \int f(x)p(x)dx $$

We can put the equation above to the KL-divergence and we have:

$$ D_{KL}(p||q) = -\int p(x) \log { \frac{q(x)}{p(x)} }dx \geq -\log \int q(x)dx = 0 $$

Consider if we wish to model a unknown distribution $p(x)$, and we try to apporximate it by a distribution $q(x,\theta)$, where θ is the parameters. One way to find the optimal solution of θ is to minimize the KL-divergence between the two distributions. The optimal solution of θ is:

$$ \theta^* = \arg\min_{\theta} D_{KL}(p||q) $$

The KL-divergence is:

$$ D_{KL}(p||q) = -\int p(x) \log q(x,\theta)dx + \int p(x) \log p(x)dx = \sum_x p(x) \log p(x) - \sum_x p(x) \log q(x,\theta) $$

It is hard to calculate the KL-divergence directly because we need to know the true distribution $p(x)$. But we can observe a finite set of data ${x_1, x_2, ..., x_N}$ from the true distribution $p(x)$. So we can use the empirical distribution $\hat{p}(x)$ to approximate the true distribution $p(x)$. So the KL-divergence is:

$$ D_{KL}(p||q) = \sum_x { -\ln q(x_i,\theta) + \ln \hat{p}(x_i) } $$

We want to minimize the KL-divergence, we can see that the second term is a constant. So we can minimize the first term. The first term is the negative log-likelihood function. So we can minimize the KL-divergence by maximizing the likelihood function. So the optimal solution of θ is:

$$ \theta^* = \arg\max_{\theta} \sum_{n=1}^{N} \ln q(x_n, \theta) $$

Let's consider the joint distribution $p(x, y)$, if the two variables are independent, the joint distribution is the product of the two marginal distribution:

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

If the two variables are not independent, we want to see how "close" they are. We can use the KL-divergence to measure the difference between the joint distribution and the product of the two marginal distribution:

$$ I[x, y] = D_{KL}(p(x, y)||p(x)p(y)) = \sum_{x, y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} = - \int \int p(x, y) \log \frac{p(x)p(y)}{p(x,y)}dxdy $$

which is called the **mutual information**. The mutual information is a way to measure the dependence between the two variables. If the two variables are independent, the mutual information is zero. If the two variables are dependent, the mutual information is positive.

It is calculated by the KL-divergence:

- [x] - H[x|y] = H[y] - H[y|x] $$

So we can consider mutual information as the reduction of uncertainty. If we know the value of the variable y, the uncertainty of the variable x is reduced by the mutual information. For Bayesian probability, $p(x)$ is prior distribution, and $p(x|y)$ is posterior distribution. The mutual information is the reduction of uncertainty.