Introduction to Machine Learning (Draft)

Qiang Liu

***** This is an unfinished draft with errors and typos. I am keep updating it. Please do not distribute outside of the class *****

Contents

1. Linear Regression

We start with linear regression, which is perhaps the simplest supervised learning method. We will also take the chance to illustrate the important concepts of overfitting and underfitting, hyperparameters and model selection, cross validation, and bias-variance trade-off.

1.1 One Parameter Cases

Given a collection of n data points $\mathcal{D} := \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$, we want to construct a function f, which, when taking $x^{(i)}$ as input, outputs a value that closely approximates $y^{(i)}$, that is, $y^{(i)} \approx f\left(x^{(i)}\right)$, $\forall i = 1, \ldots, n$.

$$x \longrightarrow \boxed{f} \longrightarrow y.$$

Linear regression considers the case when f is a linear function of x. To illustrate the general idea, we start with a simple one-parameter case when f has a form of

$$f(x;\ a) = ax,$$

where a an unknown parameter that we want to decide using data $\mathcal{D}$. The best estimation of a should make the curve fit best with the data. The simplest way to achieve this is by picking a that minimizes the *mean square error* (MSE):

$$\hat{a} = \arg\min_{a} \left\{ L(a) := \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - ax^{(i)})^2 \right\},$$

where we minimize the average square errors between the predicted value $f(x^{(i)})$ and the real value $y^{(i)}$ for each data point. Here we use $\arg\min$ to represent the value of a that minimizes the function $L(a)$, and $\hat{a}$ represents an estimation of a (and we use the hat notation "ˆ" to represent estimated values in general).

The next step is to solve this optimization. It turns out this is an easy task; the loss $L(a)$ is a simple quadratic function of a, whose optimum has a simple close-form solution. To see this, let us expand the squares in $L(a)$,

$$L(a) = a^2 \left(\frac{1}{n} \sum_{i=1}^{n} (x^{(i)})^2 \right) \; - \; 2a \left(\frac{1}{n} \sum_{i=1}^{n} x^{(i)} y^{(i)} \right) \; + \; \left(\frac{1}{n} \sum_{i=1}^{n} (y^{(i)})^2 \right),$$

The point a that minimizes $L(a)$ must have a zero gradient. Calculating the gradient:

$$\nabla L(a) = 2a \left(\frac{1}{n} \sum_{i=1}^{n} (x^{(i)})^2 \right) \; - \; 2 \left(\frac{1}{n} \sum_{i=1}^{n} x^{(i)} y^{(i)} \right).$$

The optimal point $\hat{a}$ of $L(a)$ must satisfies $\nabla L(\hat{a}) = 0$. This gives

$$\hat{a} = \frac{\sum_{i=1}^{n} y^{(i)} x^{(i)}}{\sum_{i=1}^{n} (x^{(i)})^2}. \tag{1.1}$$

By noting that the leading coefficient in $L(a)$ is non-negative $\frac{1}{n} \sum_{i=1}^{n} (x^{(i)})^2 \geq 0$, we can verify that that $\hat{a}$ is indeed a minimum point $L(a)$ (instead of a maximum point).

Exercise 1.1 To check if the formula in (1.1) makes sense, we can assume $y^{(i)} = a^* x^{(i)}$, $\forall i = 1, \ldots, n$, where a^* is the true parameter; this means that the labels $y^{(i)}$ are generated from the true functions without observation noise. Please verify that the estimation above recovers that true parameter, that is, $\hat{a} = a^*$. ■

Exercise 1.2 Assume $f(x;\ a) = x + a$. Please derive an estimation of a following the procedure above. ■

1.2 Multi-Dimensional Cases

We have more than one features and unknown parameters in more general cases. Estimating multiple parameters simultaneously in linear regression reduces to solving multivariate linear equations, which is best represented and solved using matrices (especially, matrix inversion).

Formulation

Assume we have a collection of points $\{\boldsymbol{x}^{(i)},\ y^{(i)}\}$, where each feature $\boldsymbol{x}^{(i)}$ is now a vector:

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_d^{(i)} \end{bmatrix}. \tag{1.2}$$

Each of the dimensions represents different information. For example, if $y^{(i)}$ represents the annual income of a person i, the feature $\boldsymbol{x}^{(i)}$ may include

$$\begin{aligned} x_1^{(i)} &= \textit{age} \\ x_2^{(i)} &= \textit{highest degree} \\ x_3^{(i)} &= \textit{address}, \\ &\vdots \end{aligned}$$

We want to construct a function f that can predict $y^{(i)}$ given $\boldsymbol{x}^{(i)}$, that is, $y^{(i)} \approx f(\boldsymbol{x}^{(i)})$. Here f should be a multivariate function. A simplest case is to assume that it is a linear function:

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{a}) &= a_1 x_1 + a_2 x_2 + \ldots + a_d x_d \\ &= \begin{bmatrix} a_1 & a_2 & \ldots & a_d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \\ &= \boldsymbol{a}^\top \boldsymbol{x}, \end{aligned} \tag{1.3}$$

where $\boldsymbol{a} = [a_1, \ldots, a_d]^\top$ is a set of unknown coefficients (parameters) to be estimated from data; each feature x_k ($k = 1, \ldots, d$) is multiplied by the corresponding coefficient a_k and then summed together. If a_k is positive (resp. negative), it means x_k is positively (resp. negatively) corrected with the output label y. The magnitude of a_k represents the strength of correlation between x_k and y. Of course, we will need to determine the values of all a_k using a set of empirical observation of feature-label pairs $\{\boldsymbol{x}^i, y^i\}_{i=1}^n$.

Remark 1.1 A more general form of linear functions is $f(\boldsymbol{x}, \boldsymbol{a}) = \boldsymbol{a}^\top \boldsymbol{x} + a_{d+1}$ where a_{d+1} is an additional constant coefficient. However, this can be subsumed by the formula in (1.3) by defining a "dummy feature" $x_{d+1} = 1$, such that a_{d+1} is the coefficient of x_{d+1}.

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{a}) &= a_1 x_1 + a_2 x_2 + \ldots + a_d x_d + a_{d+1} \\ &= \begin{bmatrix} a_1 & a_2 & \ldots & a_d & a_{d+1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \\ 1 \end{bmatrix} \\ &= \boldsymbol{a}^\top \overline{\boldsymbol{x}}, \end{aligned} \tag{1.4}$$

where $\overline{\boldsymbol{x}} = [x_1, \ldots, x_d, 1]^\top$ denotes the augmented feature vector. ∎

Loss Function and Optimization

Similar to the 1D case, we can measure the "fitness" of a candidate parameter $\boldsymbol{a}$ using the mean square error (MSE):

$$\begin{aligned} L(\boldsymbol{a}) &= \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{a}^\top \boldsymbol{x}^{(i)} - y^{(i)}\right)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(a_1 x_1^{(i)} + a_2 x_2^{(i)} + \cdots + a_d x_d^{(i)} - y^{(i)}\right)^2. \end{aligned} \tag{1.5}$$

The optimal estimation of $\boldsymbol{a}$ should minimize the loss function

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} L(\boldsymbol{a}).$$

A necessary condition of optimality is $\nabla L(\hat{\boldsymbol{a}}) = 0$. Here the gradient $\nabla_{\boldsymbol{a}} L(\boldsymbol{a})$ is taken w.r.t. a vector $\boldsymbol{a}$. It is a vector of the same size as $\boldsymbol{a}$, consisting of the partial derivatives $\frac{\partial L(\boldsymbol{a})}{\partial a_k}$ for each $k = 1, \ldots, d$:

$$\nabla L(\boldsymbol{a}) = \begin{bmatrix} \dfrac{\partial L(\boldsymbol{a})}{\partial a_1} \\ \dfrac{\partial L(\boldsymbol{a})}{\partial a_2} \\ \vdots \\ \dfrac{\partial L(\boldsymbol{a})}{\partial a_d} \end{bmatrix}. \tag{1.6}$$

$\nabla L(\boldsymbol{a}) = 0$ is equivalent to $\frac{\partial L(\boldsymbol{a})}{\partial a_d} = 0$ for all $k = 1, \ldots, d$. We need to calculate $\nabla L(\boldsymbol{a})$ and solve the equation.

Theorem 1.1 Let us represent the features and labels using the following matrices:

$$Y = \underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}}_{n\times 1}, \qquad X = \begin{bmatrix} (\boldsymbol{x}^{(1)})^\top \\ (\boldsymbol{x}^{(2)})^\top \\ \vdots \\ (\boldsymbol{x}^{(n)})^\top \end{bmatrix} = \underbrace{\begin{bmatrix} x_1^{(1)} & \cdots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix}}_{n\times d}. \tag{1.7}$$

We have for $L(a)$ in (1.5),

$$L(\boldsymbol{a}) = \frac{1}{n}\left\|\boldsymbol{X}\boldsymbol{a} - \boldsymbol{Y}\right\|^2, \qquad \nabla_{\boldsymbol{a}} L(\boldsymbol{a}) = \frac{2}{n}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{a} - \boldsymbol{Y}) \tag{1.8}$$

Exercise 1.3 Check if the formula in (1.8) makes sense in the simple case when $\boldsymbol{X}$, $\boldsymbol{a}$ and $\boldsymbol{Y}$ are all numbers. ■

Exercise 1.4 Go through the matrix products in $\nabla_{\boldsymbol{a}} L(\boldsymbol{a}) = \frac{2}{n}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{a} - \boldsymbol{Y})$ and check if the matrix sizes are consistent. ■

Proof. It is easy to verify that $L(\boldsymbol{a}) = \frac{1}{n}\|\boldsymbol{X}\boldsymbol{a} - \boldsymbol{Y}\|^2$ from the definition in (1.5). To calculate the gradient, we need to calculate each partial derivative, and put the terms together. See Theorem A.12 for a different proof purely based on the matrix representation.

Each partial derivative is

$$\frac{\partial L(\boldsymbol{a})}{\partial a_k} = \frac{2}{n}\sum_{i=1}^{n} x_k^{(i)}\left(\boldsymbol{x}^{(i)\top}\boldsymbol{a} - y^{(i)}\right).$$

Putting them into a vector, we get

$$\begin{aligned}
\nabla L(\boldsymbol{a}) &= \begin{bmatrix} \frac{\partial L(\boldsymbol{a})}{\partial a_1} \\ \frac{\partial L(\boldsymbol{a})}{\partial a_2} \\ \vdots \\ \frac{\partial L(\boldsymbol{a})}{\partial a_d} \end{bmatrix} = \frac{2}{n}\sum_{i=1}^{n} \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_d^{(i)} \end{bmatrix} (\boldsymbol{x}^{(i)\top}\boldsymbol{a} - y^{(i)}) \\
&= \frac{2}{n}\sum_{i=1}^{n} \boldsymbol{x}^{(i)}(\boldsymbol{x}^{(i)\top}\boldsymbol{a} - y^{(i)}) \\
&= \frac{2}{n}\Big(\big[\sum_{i=1}^{n} \boldsymbol{x}^{(i)}(\boldsymbol{x}^{(i)})^\top\big]\boldsymbol{a} \;-\; \big[\sum_{i=1}^{n} \boldsymbol{x}^{(i)} y^{(i)}\big]\Big).
\end{aligned} \tag{1.9}$$

We just need to show that $\sum_{i=1}^n \boldsymbol{x}^{(i)}(\boldsymbol{x}^{(i)})^\top = \boldsymbol{X}^\top\boldsymbol{X}$ and $\sum_{i=1}^n \boldsymbol{x}^{(i)} y^{(i)} = \boldsymbol{X}^\top\boldsymbol{Y}$. This can be verified as follows.

$$\boldsymbol{b} \overset{def}{=} \sum_{i=1}^{n} \boldsymbol{x}^{(i)} y^{(i)} = \begin{bmatrix} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \dots & \boldsymbol{x}^{(n)} \end{bmatrix} \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix} = \boldsymbol{X}^\top\boldsymbol{Y}, \tag{1.10}$$

$$A \overset{def}{=} \sum_{i=1}^{n} \boldsymbol{x}^{(i)}(\boldsymbol{x}^{(i)})^\top = \begin{bmatrix} \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} & \dots & \boldsymbol{x}^{(n)} \end{bmatrix} \begin{bmatrix} (\boldsymbol{x}^{(1)})^\top \\ (\boldsymbol{x}^{(2)})^\top \\ \vdots \\ (\boldsymbol{x}^{(n)})^\top \end{bmatrix} = \boldsymbol{X}^\top \boldsymbol{X}. \tag{1.11}$$

■

Recall from equation (1.9) that minimizing the loss function is equivalent to find $\hat{\boldsymbol{a}}$ such that $\nabla L(\hat{\boldsymbol{a}}) = 0$. We have

$$\begin{aligned} \nabla L(\hat{\boldsymbol{a}}) &= \frac{2}{n}\boldsymbol{X}^\top\big(\boldsymbol{X}\boldsymbol{a} - \boldsymbol{Y}\big) \\ &= \frac{2}{n}(\boldsymbol{X}^\top \boldsymbol{X}\hat{\boldsymbol{a}} - \boldsymbol{X}^\top \boldsymbol{Y}) = 0. \end{aligned} \tag{1.12}$$

This gives

$$\begin{aligned} \boldsymbol{X}^\top \boldsymbol{X}\hat{\boldsymbol{a}} - \boldsymbol{X}^\top \boldsymbol{Y} &= 0 \\ \boldsymbol{X}^\top \boldsymbol{X}\hat{\boldsymbol{a}} &= \boldsymbol{X}^\top \boldsymbol{Y}, \\ \hat{\boldsymbol{a}} &= (\boldsymbol{X}^\top \boldsymbol{X})^{-1}(\boldsymbol{X}^\top \boldsymbol{Y}). \end{aligned} \tag{1.13}$$

where $\boldsymbol{A}^{-1}$ denotes the matrix inversion of $\boldsymbol{A}$ (we assume $\boldsymbol{A}$ is invertible), which satisfies $\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}$. Here $\boldsymbol{I}$ denotes the identity matrix of the same size as $\boldsymbol{A}$. We call $\boldsymbol{X}^\top \boldsymbol{X}$ the gram matrix sometimes. Note that the size of $\boldsymbol{X}^\top \boldsymbol{X}$ is $d \times d$.

Discussion of number of data points n and dimension of data d

- If $n = d$, the data matrix $\boldsymbol{X}$ is a $n \times n$ square matrix. If we further assume X is invertible (equivalently, $\{\boldsymbol{x}^{(1)} \cdots \boldsymbol{x}^{(n)}\}$ is linearly independent), we get

  $$\begin{aligned} \hat{\boldsymbol{a}} &= (\boldsymbol{X}^\top \boldsymbol{X})^{-1}(\boldsymbol{X}^\top \boldsymbol{Y}) \\ &= \boldsymbol{X}^{-1}(\boldsymbol{X}^\top)^{-1}\boldsymbol{X}^\top \boldsymbol{Y} && \text{// } (\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \text{ if both } \boldsymbol{A} \text{ and } \boldsymbol{B} \text{ are invertible} \\ &= \boldsymbol{X}^{-1}\boldsymbol{Y}. && \text{// } \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I} \end{aligned}$$

  In this case, $\hat{\boldsymbol{a}}$ is also the solution of

  $$\boldsymbol{X}\hat{\boldsymbol{a}} = \boldsymbol{Y}, \tag{1.14}$$

  which representing fitting all the data points exactly, that is, $y^{(i)} = \hat{\boldsymbol{a}}^\top \boldsymbol{x}^{(i)}$ for all $i = 1, \dots, n$.
- If $n < d$, the gram matrix $\boldsymbol{X}^\top \boldsymbol{X}$ is never invertible (why?), and the formula above can not be applied. In fact, in this case, there exists infinite many values of $\boldsymbol{a}$ which fit with the data points perfectly (meaning that the equation in (1.14) is under-determined and has an infinite number of solutions).
- If $n > d$ and the gram matrix $X^\top X$ is invertible (which is equivalent to X having a rank that is no smaller than d), the equation (1.14) is over-determined and does not have a feature solution (that is, it is impossible to perfectly fix all the data simutanously), but $\hat{\boldsymbol{a}}$ still admits an minimum of the loss function $L(\boldsymbol{a})$.
  Note that $X^\top X$ is not always invertible when $n > d$ (think about the case when all the n points are exactly the same).

TODO: add figure

■ **Example 1.1 — Polynomial Regression.** Consider fitting the data with a polynomial function of degree d ($d \geq 1$)

$$f(x, \boldsymbol{a}) = a_1 + a_2 x + \cdots + a_{d+1} x^d,$$

where $\boldsymbol{a} = [a_1, \ldots, a_{d+1}]^\top$ collects the parameters we need to esstimate from data. When $d = 1$, this reduces to standard linear regression. As d increases, the function $f(x, \boldsymbol{a})$ is increasingly more flexible and can fit more complex linear patterns. Of course, this is at the cost of increasing the number of parameters need to be estimated.

Even though this allows us to fit nonlinear curves, the estimation of the coefficient $\boldsymbol{a}$ can be solved the same above using linear regression. To see this, denote $\boldsymbol{\phi}(x) = [1,\ x,\ x^2, \cdots, x^d]^\top$. We have

$$f(x, \boldsymbol{a}) = a_1 + a_2 x + \cdots + a_{d+1} x^d = \boldsymbol{a}^\top \boldsymbol{\phi}(x).$$

Therefore, we can estimate $\boldsymbol{a}$ as before by treating $\boldsymbol{\phi}(x)$ as the feature vector.

1.3 Ridge Regression

When the data size is small (e.g., $n < d$), or when the data is very noisy, it is useful to add a regularization term into the objective function. For example,

$$\min_{\boldsymbol{a}} \left\{ L(\boldsymbol{a}) := ||\boldsymbol{y} - X\boldsymbol{a}||^2 + \lambda ||\boldsymbol{a} - \boldsymbol{a}_0||^2 \right\},$$

where $\boldsymbol{a}_0$ is a guess of the value of $\boldsymbol{a}$, assuming no data is observed. By fault, we may set $\boldsymbol{a}_0 = 0$ as a default guess, unless there are some strong preference otherwise. here λ, called the regularization coefficient, is a positive number that trades off the data fitting error and the regularization term. It controls how much we want to rely on the prior guess a_0 or the data.

Taking the gradient of $L(\boldsymbol{a})$:

$$\nabla_{\boldsymbol{a}} L(\boldsymbol{a}) = 2X^\top (X\boldsymbol{a} - \boldsymbol{y}) + 2\lambda(\boldsymbol{a} - \boldsymbol{a}_0).$$

Setting $\nabla_{\boldsymbol{a}} L(\boldsymbol{a}) = 0$ gives the optimal solution:

$$\hat{\boldsymbol{a}}(\lambda) = (X^\top X + \lambda I)^{-1}(X^\top \boldsymbol{y} + \lambda \boldsymbol{a}_0),$$

where we use notation $\hat{\boldsymbol{a}}(\lambda)$ to reflect that the value of $\hat{\boldsymbol{a}}$ depends on the λ. If $\lambda > 0$, then $(X^\top X + \lambda I)$ is always invertible even when $X^\top X$ is not. (why?) If the regularization is strong with a large λ, the estimator would be close to $\boldsymbol{a}_0$. In particular, you can check from the formula above that $\hat{\boldsymbol{a}}(+\infty) = \boldsymbol{a}_0$.

Remark 1.2 If there is no strong prior information regarding the value of $\boldsymbol{a}_0$, it is common to set $\boldsymbol{a}_0$ to zero, which gives

$$\min_{\boldsymbol{a}} \|\boldsymbol{y} - X\boldsymbol{a}\|^2 + \lambda \|\boldsymbol{a}\|^2,$$

where the penalty term now reduces to square of norm $\|\boldsymbol{a}\|^2$, which favors coefficients with smaller values. This is the common form of L_2 *regression or ridge regression* in the literature.

Because setting a_i to be zero corresponds to discarding its corresponding feature x_i, encouraging small values of $\boldsymbol{a}$ allows us to control how sensitive the output can depend on the input and hence prevent over-fitting. ■

Remark 1.3 In practice, we may only apply the regularization on the non-constant coefficients. Suppose $f(\boldsymbol{x}, \boldsymbol{a}) = \boldsymbol{a}^\top \boldsymbol{x} + a_0$, where we separate the constant coefficient a_0 with the other coefficients. We may consider

$$\min_{\boldsymbol{a}, a_0} \|\boldsymbol{y} - X\boldsymbol{a} - a_0\|^2 + \lambda \|\boldsymbol{a}\|^2 .$$

If we apply a very strong regularization ($\lambda \to \infty$), the resulting model would be $f(\boldsymbol{x}, \boldsymbol{a}) = a_0$, which is a simple constant model. ∎

Remark 1.4 A more general case is to assign each coordinate of parameters with an individual regularization coefficient:

$$\min_{\boldsymbol{a}} \|\boldsymbol{y} - \boldsymbol{X}a\|^2 + \sum_{k=1}^{d} \lambda_k a_k^2, \tag{1.15}$$

where $\{\lambda_k\}$ is a set of regularization coefficients. This is useful, for example, for polynomial regression (Example 1.1, in which the coefficients of different degrees should be regularized differently (the higher order coefficients should be penalized more heavily than the lower order ones).

We can rewrite the loss function of linear regression as:

$$L(\boldsymbol{a}) = \|Y - X\boldsymbol{a}\|^2 + \boldsymbol{a}^\top \Lambda \boldsymbol{a}, \tag{1.16}$$

where $\Lambda = \mathrm{diag}([\lambda_1, \ldots, \lambda_d])$ denotes diagonal matrix formed by $\{\lambda_i\}$. To find the optimum, we take the derivative:

$$\nabla L(\boldsymbol{a}) = -2X^\top (Y - X\boldsymbol{a}) + 2\Lambda \boldsymbol{a}.$$

Let $\nabla L(\boldsymbol{a}) = 0$, we can get the optimal $\boldsymbol{a}$:

$$\boldsymbol{a} = (X^\top X + \Lambda)^{-1} X^\top Y.$$

∎

1.4 Bias-variance Trade-off: Theoretical Analysis

In practice, the guess $\boldsymbol{a}_0$ is unlikely to exactly equal the true value, especially when it is set to be zero. If so, is regularization still a reasonable thing to do in general? The answer is YES. And this is one of the most important ideas in machine learning. Let us illustrate this with some simple analysis.

Background 1.1 — Random Variable. Let us have a brief review of random variables. Assume ξ is a (continuous) random variable in $\mathbb{R}$ with density function $p(\xi)$.

$$\begin{aligned} \text{Mean:} \quad & \mathbb{E}[\xi] = \int \xi p(\xi) dx. \\ \text{Variance:} \quad & \mathrm{Var}(\xi) = \mathbb{E}[(\xi - \mathbb{E}[\xi])^2] \\ \text{Covariance:} \quad & \mathrm{Cov}(\xi, \xi') = \mathbb{E}[(\xi - \mathbb{E}[\xi])(\xi' - \mathbb{E}[\xi'])]. \end{aligned}$$

If two random variables are independent, we have zero covariance:

$$\xi \perp \xi' \qquad \Longrightarrow \qquad \mathrm{Cov}(\xi,\ \xi') = 0.$$

But the opposite is not true, that is, random variables with zero covariance may be dependent.

If ξ and ξ' are independent, then

$$\mathrm{Var}(a\xi + b\xi') = a^2\mathrm{Var}(\xi) + b^2\mathrm{Var}(\xi'). \tag{1.17}$$

(what if they are not independent?) ■

For simplicity, consider the 1D case in which the feature x is a scalar and we want to fit a one-parameter model $y \approx ax$. In order to develope a theoretical analysis, we need to make some assumption on the data. Assume the data is generated in the following way:

$$y^{(i)} = a^* x^{(i)} + \xi^{(i)}, \qquad \forall i,$$

where a^* is an unknown true value of a, and $\{\xi^{(i)}\}$ are independent random noises with zero mean and variance σ^2, that is,

$$\mathbb{E}[\xi^{(i)}] = 0, \qquad \mathrm{Var}[\xi^{(i)}] = \sigma^2, \quad \forall i.$$

The regularized loss function is

$$\min_a \left\{ \lambda(a - a_0)^2 + \sum_{i=1}^n (y^{(i)} - ax^{(i)})^2 \right\}.$$

The optimal solution is

$$\hat{a}(\lambda) = \frac{\lambda a_0 + \sum_{i=1}^n x^{(i)} y^{(i)}}{\lambda + \sum_{i=1}^n (x^{(i)})^2}.$$

We are interested in knowing how much $\hat{a}(\lambda)$ deviates from the true a^*. Because $\hat{a}(\lambda)$ is a random variable (since it is a function of the data, which is random), we need to measure the error using the expected mean square error (MSE):

$$\mathrm{MSE}(\lambda) = \mathbb{E}[(\hat{a}(\lambda) - a^*)^2].$$

where the expectation $\mathbb{E}[\cdot]$ is taken w.r.t. the data noise $\{\xi^{(i)}\}$.

Importantly, MSE yields the following **bias-variance decomposition**:

$$\begin{aligned} \mathrm{MSE}(\lambda) &= \mathbb{E}[(\hat{a}(\lambda) - a^*)^2] \\ &= \mathrm{Var}(\hat{a}(\lambda)) + (\mathbb{E}[\hat{a}(\lambda)] - a^*)^2, \end{aligned}$$

where the first term is variance of the estimator $\hat{a}(\lambda)$, the second term the square of the bias: $\mathrm{bias}(\hat{a}(\lambda)) = \mathbb{E}[\hat{a}(\lambda)] - a^*$. Prove this yourself.

This model is simple enough so that we can calculate both the bias and variance explicitly. Substitute $y^{(i)} = a^* x^{(i)} + \xi^{(i)}$, and define $\beta = \frac{1}{n}\sum_{i=1}^n (x^{(i)})^2$ (it is the second moment of the features). We have

$$\begin{aligned} \hat{a} - a^* &= \frac{\lambda a_0 + \sum_{i=1}^n x^{(i)} y^{(i)}}{\lambda + n\beta} - a^* \\ &= \frac{\lambda a_0 + \sum_{i=1}^n x^{(i)} (a^* x^{(i)} + \xi^{(i)})}{\lambda + n\beta} - a^* \\ &= \frac{\lambda a_0 + \sum_{i=1}^n (x^{(i)})^2 a^* + \sum_{i=1}^n x^{(i)} \xi^{(i)}}{\lambda + n\beta} - a^* \\ &= \frac{-\lambda(a^* - a_0)}{\lambda + n\beta} + \sum_{i=1}^n \frac{x^{(i)}}{\lambda + n\beta} \xi^{(i)}, \end{aligned}$$

where the first term is a deterministic term that only contributes bias (if $a_0 \neq a^*$), and the second term only contributes random zero-mean noise. We can calculate the mean:

$$\text{bias}(\hat{a}(\lambda)) = \mathbb{E}[\hat{a}(\lambda)] - a^* = \frac{-\lambda(a^* - a_0)}{\lambda + n\beta}. \qquad \text{//note } \mathbb{E}[\xi^{(i)}] = 0$$

The variance:

$$\text{Var}(\hat{a}(\lambda)) = \frac{n\beta\sigma^2}{(\lambda + n\beta)^2}. \qquad \text{//Use (1.17); note } \{\xi^{(i)}\} \text{ are independent;}$$

Combing them gives the explicit form of the MSE function:

$$\text{MSE}(\lambda) = \frac{\lambda^2(a^* - a_0)^2}{(\lambda + n\beta)^2} + \frac{n\beta\sigma^2}{(\lambda + n\beta)^2}.$$

Consider the special case:

$$\text{MSE}(0) = \frac{\sigma^2}{n\beta}, \qquad \text{//No regularization}$$
$$\text{MSE}(\infty) = (a^* - a_0)^2. \qquad \text{//what does this mean?}$$

We can also find the optimal λ^* to minimize $\text{MSE}(\lambda)$. After some derivation (by finding the zero gradient point), the optimal solution is

$$\lambda^* = \frac{\sigma^2}{(a^* - a_0)^2}.$$

So the intuition is that λ^* should be large if the guess a_0 is close to a^* (the prior is good), or when σ^2 is large (the data is noisy). This gives

$$\text{MSE}(\lambda^*) = \frac{\sigma^2}{n\beta + \lambda^*}.$$

The optimal $\text{MSE}(\lambda^*)$ is smaller than $\text{MSE}(0)$ once $\lambda^* > 0$.

Try to think how the curve of $\text{MSE}(\lambda)$ and $\text{var}(\hat{a}(\lambda))$ and $\text{bias}(\hat{a}(\lambda))^2$ look like. Plot them in matlab or python.

1.5 Cross Validation

Although the models are trained on "known" datasets (training data), we are more interested knowing how the models may perform on future "unknown" datasets. Cross validation is technique that allows us to evaluate the performance of the models on future datasets by creating "held-out" testing sets within the known dataset.

k-fold Cross Validation

K-fold cross validation splits the data into k smaller subsets, where each subset takes a turn as the test set See algorithm 1.

Figure 1.1 illustrates a 3-fold cross validation procedure: we split the dataset into 3 partitions evenly and calculate the mean square error of the model for 3 times, and the final error can be reported as:

$$\overline{\text{MSE}} = \frac{1}{3}(\text{MSE}_1 + \text{MSE}_2 + \text{MSE}_3). \tag{1.18}$$

Larger k may give better estimation but is also more expensive (since you need to train the model for k times). The extreme case when $k = n - 1$ is called leave-one-out cross validation. In practical, it is common to use $k = 10$, or $k = 5$ when you want to save computation.

Algorithm 1 k-fold cross validation

k-fold CVk, dataset: split the dataset M into k smaller datasets $M_i, i \in \{1, 2, \ldots, k\}$ evenly $j = 1$ to k Train the model on $M \setminus M_i$, that is, $\hat{\mathtt{f}} = \mathtt{train}(\mathtt{M} \setminus \mathtt{M_i})$. Evaluate the trained model f on M_i, that is, $\mathtt{MSE_i} = \mathtt{evaluate}(\hat{\mathtt{f}},\ \mathtt{M_i})$. Report average value of k times of evaluation Note. $\hat{\mathtt{f}} = \mathtt{train}(\mathtt{M})$ denotes training the model on dataset M. $\mathtt{MSE} = \mathtt{evaluate}(\hat{\mathtt{f}},\ \mathtt{M})$ evaluates the mean square of model $\hat{f}$ on dataset M.

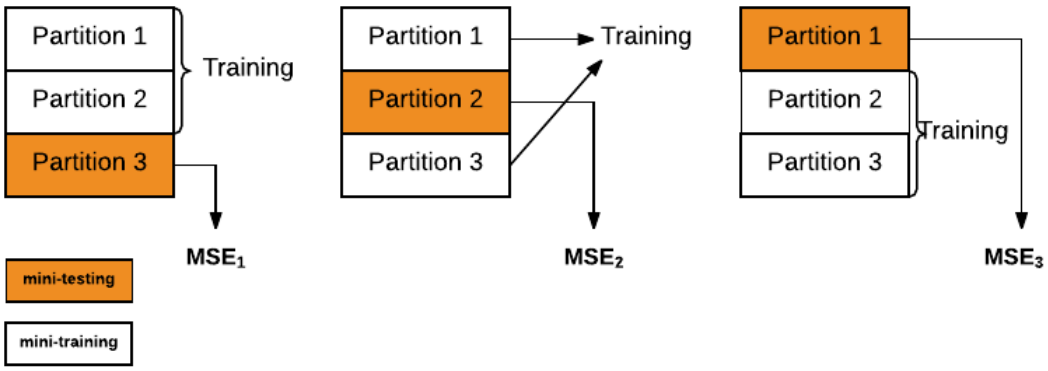


Figure 1.1: k-fold cross validation. Cross validation partitions the data (usually the training set) into k subsets, which are taking in turn as "mini-training" sets and "mini-testing" sets.

Model Selection

Cross validation can also be used to select the best model (model selection), or the best hyper-parameters. Suppose you have several models and want to decide which model is the best choice for your prediction problem, here is what you should do in practice:

- Split your data into a `training dataset` and a `testing datasets`;
- Perform cross validation on the `training dataset` for each model;
- Choose the model that has the lowest average cross validation error;
- Re-train the best model on the `whole training dataset`;
- Test the trained best model on the `testing dataset`.

2. Learning with Gradient Descent

A central component of learning is the *optimization problem* of minimizing the loss function. Although we are lucky enough to have a simple closed form solution for linear regression, optimization is often more difficult in general cases and can only be solved using iterative numerical algorithms. *Gradient descent* algorithm is a simple yet remarkably powerful numerical optimization algorithm; it has been widely used in practice, especially for very large scale problems for which more advanced algorithms, such as Newton's method are computationally prohibitive.

We start with a set of examples for which iterative numerical optimization is required.

■ **Example 2.1 — Large Scale Problems.** Recall that linear regression with L2 regularization requires to solve

$$\min_{\boldsymbol{a}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{a}\|_2^2 \; + \; \lambda \|\boldsymbol{a}\|_2^2 , \tag{2.1}$$

whose solution yields a simple formula:

$$\hat{\boldsymbol{a}} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda I)^{-1} (\boldsymbol{X}^\top \boldsymbol{y}).$$

However, calculating $\hat{\boldsymbol{\alpha}}$ exactly using the formula above requires a time complexity of $O(d^3+dn)$ where d is the number of features and n is the number of data points; here $O(d^3)$ comes from calculating the matrix inverse and $O(dn)$ comes from matrix multiplications. Therefore, when d and/or n is very large, exact calculation is not feasible (e.g., d can be the number of words in natural language process, or number of genes in bioinformatics; n can be the number of websites for web classification). In these large scale cases, we often prefer to use *iterative approximate algorithms* that solve the optimization problem with less accuracy, in exchange of much faster computation. This is something that can be achieved by using gradient descent, or stochastic gradient descent that we describe in the sequel.

■ **Example 2.2 — Robust Regression.** Mean square error (MSE) is not the only choice of loss functions. We can alternatively use the mean absolute error as the loss function, which is the sum of absolute values of the errors:

$$L(\boldsymbol{a}) = \sum_{i=1}^{n} |y^i - \boldsymbol{a}^\top \boldsymbol{x}^i| := \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{a}\|_1 .$$

This is called *least absolute regression*, where $\|\boldsymbol{\theta}\|_1 := \sum_i |\theta_i|$ is called the L1 norm of vector $\boldsymbol{\theta}$ (in contrast to the L2 norm). Compared with the typical mean square loss, the mean absolute loss is more robust to outliers, and the regression with mean absolute loss is also called robust regression sometimes. add why; add figure

■ **Example 2.3 — Sparse Regression.** Besides the L2 regularization, we can use the following L1 regularization:

$$L(\boldsymbol{a}) = \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_1 .$$

where $\|\boldsymbol{a}\|_1 := \sum_{i=1}^{d} |a_d|$ and is called the L_1 norm of $\boldsymbol{a}$. Compared with L_2 regularization, L_1 regularization has the property of favouring sparse solutions (that is, the minimizers $\hat{\boldsymbol{a}}$ of the L_1 regularized loss functions tends to have more zero elements).

To see why L1 regularization can encourage sparsity, let us consider a very simple case when all the different dimensions of the features vector $\boldsymbol{x} = [x_1, \ldots, x_d]^\top$ are identical, that is, $x_1 = \cdots = x_d$. In this case, it is easy to see that the following two coefficient vectors fit the data equally well:

$$\boldsymbol{a}_1 = [1, 0, \ldots, 0, 0]^\top, \qquad \boldsymbol{a}_2 = [1/d, \cdots, 1/d]^\top.$$

The difference is that $\boldsymbol{a}_1$ put all the weights on the first feature x_1, while $\boldsymbol{a}_2$ distribute the weights evenly. Depending on the application scenarios, we may prefer one solution over the other. For example, $\boldsymbol{a}_1$ yields a more computationally efficient "sparse" model which only require index feature x_1.

It turns out using L1 or L2 regularization allows us to encode our preference on solutions like $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ into the algorithm.

1. If we use L_2 regularization, then $\boldsymbol{a}_2$ is more preferred than $\boldsymbol{a}_2$ because $\|\boldsymbol{a}_2\|_2 = 1/\sqrt{d} \leq \|\boldsymbol{a}_1\|_2$.
2. If we use L_1 regularization, the more sparse solution $\boldsymbol{a}$ will be more favored because $\|\boldsymbol{a}\|_1 < \|\boldsymbol{b}\|_1$.

■ **Example 2.4 — Nonlinear Regression.** Complex loss function may arise when the model is not a linear function of the parameters. For example, consider $f(\boldsymbol{x},\ \boldsymbol{a}) = \sigma(\boldsymbol{a}^\top \boldsymbol{x})$ where $\sigma(\cdot)$ is a nonlinear function, the sigmoid function $\sigma(t) = 1/(1 + \exp(-x))$, or ReLU function $\sigma(t) = \max(0, x)$. This model is a neural network with a single neuron (see Chapter 8). In this case, the MSE loss function is

$$L(\boldsymbol{a}) = \sum_{i=1}^{n} \left(y^i - \sigma(\boldsymbol{a}^\top \boldsymbol{x}^i)\right)^2 .$$

Minimizing it requires approximate numerical methods.

2.1 Gradient Descent

Gradient descent is a simple iterative optimization algorithm for finding a (local) minimum of a function $f(\boldsymbol{\theta})$. It starts with an initial guess $\boldsymbol{\theta}_0$, and perform iterative update of form

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha \nabla f(\boldsymbol{\theta}_t),$$

where at each iteration t, the parameter $\boldsymbol{\theta}$ is updated to move along the *negative* gradient descent, which is expected to decreases the function value. Here α is a small positive number called *step size*, or *learning rate*, which controls how much we move at each iteration. TODO: figure all two dimensional curve

Why does gradient descent decrease the function?

Claim 2.1 When the step size α is "sufficiently" small, the gradient descent update decreases (or never increases) that loss function. Formally, we have

$$f(\boldsymbol{\theta}_{t+1}) = f(\boldsymbol{\theta}_t) \ - \alpha \left\|\nabla f(\boldsymbol{\theta}_t)\right\|^2 \ + \ O(\alpha^2).$$

Therefore, when α is infinitesimal, gradient descent decreases the objective function by an amount of $\alpha \left\|\nabla f(\boldsymbol{\theta}_t)\right\|^2$.

Proof. Based on the first order Taylor expansion, we have

$$f(\boldsymbol{\theta} + \alpha\boldsymbol{\delta}) = f(\boldsymbol{\theta}) \ + \ \alpha \nabla f(\boldsymbol{\theta})^\top \boldsymbol{\delta} \ + \ O(\alpha^2)$$

By taking $\boldsymbol{\delta} = -\nabla f(\boldsymbol{\theta})$, we get

$$\begin{aligned} f(\boldsymbol{\theta}_{t+1}) f(\boldsymbol{\theta}_t - \alpha \nabla f(\boldsymbol{\theta}_t)) &\approx f(\boldsymbol{\theta}_t) - \alpha \nabla f(\boldsymbol{\theta}_t)^\top \nabla f(\boldsymbol{\theta}_t) + O(\alpha^2) \\ &= f(\boldsymbol{\theta}_t) - \alpha ||\nabla f(\boldsymbol{\theta}_t)||^2 + O(\alpha^2). \end{aligned}$$

■

How small is small? Here is a more precise statement. (Optional)

Theorem 2.2 Assume $\lambda_{max}\left(\nabla^2 f(\boldsymbol{\theta})\right) \leq M$ for any $\boldsymbol{\theta}$, where $\lambda_{max}(\cdot)$ denotes the maximum eigenvalue of a matrix, we have

$$f(\boldsymbol{\theta}_{t+1}) \leq f(\boldsymbol{\theta}_t) \ - (\alpha - \frac{1}{2} M \alpha^2) \left\|\nabla f(\boldsymbol{\theta}_t)\right\|^2 .$$

Therefore, when the step-size satisfies $0 < \alpha < 2/M$, gradient descent strictly decreases the objective function unless the gradient is already zero (that is, $\nabla f(\boldsymbol{\theta}_t) = 0$).

Remark 2.1 Points that have a zero gradient $\nabla f(\boldsymbol{\theta}) = 0$ are called critical points (or stationary points). A critical point is not necessarily a local minimum; it may also be a saddle point or even local maximum. However, saddle points and local maxima are usually unstable under gradient descent, in that a small noise may kick the parameter departing away (like a ball rolling down hill). The magnitude of noise required to escape a local optimal or saddle point depends on the "flatness" of the curve. If the curve is very flat, gradient descent may got stuck at a saddle point. ■

Proof. By Taylor expression,

$$f(\boldsymbol{\theta}_{t+1}) = f(\boldsymbol{\theta}_t) + \nabla f(\boldsymbol{\theta}_t)^\top (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) + R,$$

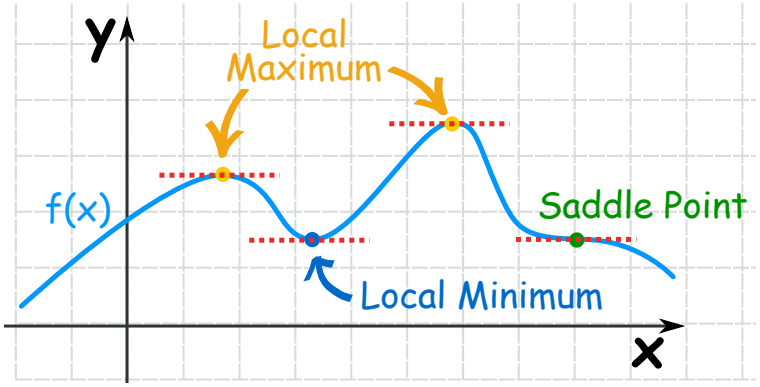


Figure 2.1: TODO: Replace for copyright

where the residual term is

$$R = \frac{1}{2}(\boldsymbol{\theta}_{t+1} - w_t)^\top H(\boldsymbol{\theta}_{t+1} - w_t),$$

where $H = \nabla^2 f(\boldsymbol{\theta}_t + \alpha(\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t))$ and α is a number in $[0, 1]$. Because $\lambda_{max}(H) \leq M$ by assumption, we have

$$R \leq \frac{1}{2} M \left\| \boldsymbol{\theta}_{t+1} - w_t \right\|^2.$$

We have

$$f(\boldsymbol{\theta}_{t+1}) \leq f(\boldsymbol{\theta}_t) + \nabla f(\boldsymbol{\theta}_t)^\top (\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t) + \frac{1}{2} M \left\| \boldsymbol{\theta}_{t+1} - w_t \right\|^2.$$

Recall that $\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t = -\alpha \nabla f(\boldsymbol{\theta}_t)$, we have

$$\begin{aligned} f(\boldsymbol{\theta}_{t+1}) &\leq f(\boldsymbol{\theta}_t) + \nabla f(\boldsymbol{\theta}_t)^\top \left(-\alpha \nabla f(\boldsymbol{\theta}_t)\right) + \frac{1}{2} M \left\| \alpha \nabla f(\boldsymbol{\theta}_t) \right\|^2 \\ &= f(\boldsymbol{\theta}_t) - \alpha \left\| \nabla f(\boldsymbol{\theta}_t) \right\|^2 + \frac{1}{2} M \alpha^2 \left\| \nabla f(\boldsymbol{\theta}_t) \right\|^2 \\ &= f(\boldsymbol{\theta}_t) - (\alpha - \frac{1}{2} M \alpha^2) \left\| \nabla f(\boldsymbol{\theta}_t) \right\|^2. \end{aligned}$$

■

Exercise 2.1 Consider function $f(\theta) = a\theta^2 + b\theta + c$, where $a > 0$. Starting from an arbitrary initialization θ_0, what is the best way to choose the step size in order to converge to the minimum point of $f(\theta)$ as fast as possible? ■

2.2 Convex functions

Gradient descent only guarantees to locally decrease the loss function (with sufficiently small step sizes), and only guarantees to converge to *stationary points* with zero gradients ($\nabla L(\boldsymbol{\theta}) = 0$). Convex functions are a nice class of functions whose all stationary points are global optima.

A functions $f \colon \mathcal{M} \to \mathbb{R}$ is said to be convex on domain $\mathcal{M}$ if

$$f\left(a\boldsymbol{\theta} + (1-a)\boldsymbol{\theta}'\right) \leq af(\boldsymbol{\theta}) + (1-a)f(\boldsymbol{\theta}'), \quad \text{for any } \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{M} \text{ and } a \in [0, 1]. \tag{2.2}$$

In order to make this definition make sense, the domain $\mathcal{M}$ must also be convex, in that if $\boldsymbol{\theta}, \boldsymbol{\theta}'$ are in $\mathcal{M}$, then $a\boldsymbol{\theta} + (1-a)\boldsymbol{\theta}'$ must also be in $\mathcal{M}$ for any a. Equation 2.2 is known as Jensen's inequality.

The negative of a convex function is called a concave function. One simple example of convex function is the quadratic function with non-negative leading coefficient:

$$f(\theta) = \frac{1}{2}a\theta^2 + b\theta + c, \quad a \geq 0,$$

where θ is a scalar. Note that $\nabla^2 f(\theta) = a \geq 0$, where $\nabla^2 f(\theta)$ is the second derivative of $f(\theta)$.

In fact, an general way to tell if a single-variable differentiable function is convex is to check if it has non-negative second derivative for all values of w in its definition domain:

$$f \text{ is convex if and only if } \quad \nabla^2 f(\theta) \geq 0, \quad \forall \theta.$$

This is equivalent to that its first derivative ∇f is monotonically increasing.

Exercise 2.2 Consider function

$$f(\theta) = \log(1 + \exp(\theta)).$$

Please show that it is convex and verify this by plotting its curve in Python or Matlab. ■

Exercise 2.3 Some non-convex function can be transformed to a convex function with monotonic mapping (which preserves the optimal points). Consider the bell curve:

$$f(\theta) = \exp(-\frac{1}{2}\theta^2).$$

Please show that $f(\theta)$ is not a concave function, but $g(\theta) = \log(f(\theta))$ is a concave function. ■

Convex Functions of Multiple Variables (Optional)

For vector $\boldsymbol{\theta} \in \mathbb{R}^d$, the second derivative (also called the Hessian matrix) is defined as a $d \times d$ matrix:

$$\nabla^2 f(\boldsymbol{\theta}) = \begin{bmatrix} \partial_{\theta_1,\theta_1} f(\boldsymbol{\theta}) & \cdots & \partial_{\theta_1,\theta_d} f(\boldsymbol{\theta}) \\ \vdots & \ddots & \vdots \\ \partial_{\theta_d,\theta_1} f(\boldsymbol{\theta}) & \cdots & \partial_{\theta_d,\theta_d} f(\boldsymbol{\theta}) \end{bmatrix},$$

where $\partial_{\theta_i,\theta_j} f(\boldsymbol{\theta})$ denotes the cross partial derivative w.r.t. θ_i and θ_j, evaluated at point $\boldsymbol{\theta}$. Note that $\nabla^2 f(\boldsymbol{\theta})$ is always a symmetric matrix because $\partial_{\theta_j,\theta_i} f = \partial_{\theta_i,\theta_j} f$.

It can be shown that for a second order differentiable function f, it is convex if and only if its Hessian matrix is positive semidefinite at every point $\boldsymbol{\theta}$:

$$f \text{ is convex if and only if } \quad \nabla^2 f(\boldsymbol{\theta}) \succeq 0, \quad \forall \boldsymbol{w} \in \mathcal{M}.$$

where $\nabla^2 f(\boldsymbol{\theta}) \succeq 0$ denotes that matrix $\nabla^2 f(\boldsymbol{\theta})$ is positive semidefinite.

Recall that a matrix H is positive semidefinite is equivalent to the following statements:

1. For any vector $\boldsymbol{x}$ in $\mathbb{R}^d$, we have $\boldsymbol{x}^\top H \boldsymbol{x} \geq 0$.

1. All its eigenvalues are non-negative.

2. There exists some matrix Φ such that $H = \Phi^\top \Phi$. The size of Φ is $k \times d$ where k is the rank of H.

■ **Example 2.5** The loss function of linear regression is

$$L(\boldsymbol{\theta}) = \|Y - \Phi\boldsymbol{\theta}\|^2$$

We can show that $\nabla^2 L(\boldsymbol{\theta}) = \Phi^\top \Phi$ which is always positive semidefinite. Therefore, $L(\boldsymbol{\theta})$ is always a convex function. Another way to show the convexity of $L(\boldsymbol{\theta})$ is to directly prove the inequality $L(\alpha\boldsymbol{\theta} + (1-\alpha)\boldsymbol{\theta'}) \leq \alpha L(\boldsymbol{\theta}) + (1-\alpha)L(\boldsymbol{\theta'})$:

$$\begin{aligned} L(\alpha\boldsymbol{\theta} + (1-\alpha)\boldsymbol{\theta'}) &= \left\|Y - \Phi(\alpha\boldsymbol{\theta} + (1-\alpha)\boldsymbol{\theta'})\right\|^2 \\ &= \left\|\alpha(Y - \Phi\boldsymbol{\theta}) \ + \ (1-\alpha)(Y - \Phi\boldsymbol{\theta'})\right\|^2 \\ &\leq \alpha \left\|Y - \Phi\boldsymbol{\theta}\right\|^2 + (1-\alpha)\left\|Y - \Phi\boldsymbol{\theta'}\right\|^2 \qquad \text{//See Lemma 2.3 below} \\ &= \alpha L(\boldsymbol{\theta}) + (1-\alpha)L(\boldsymbol{\theta'}). \end{aligned}$$

Lemma 2.3 For any two vectors $\boldsymbol{\theta}$ and $\boldsymbol{\theta'}$, and $\alpha \in [0,1]$, we have

$$\left\|\alpha\boldsymbol{\theta} + (1-\alpha)\boldsymbol{\theta'}\right\|^2 \leq \alpha \left\|\boldsymbol{\theta}\right\|^2 + (1-\alpha)\left\|\boldsymbol{\theta'}\right\|^2.$$

Proof. Let us directly show that the left side minus the right side is always non-negative:

$$\begin{aligned} &\alpha \left\|\boldsymbol{\theta}\right\|^2 + (1-\alpha)\left\|\boldsymbol{\theta'}\right\|^2 - \left\|\alpha\boldsymbol{\theta} + (1-\alpha)\boldsymbol{\theta'}\right\|^2 \\ &= \alpha \left\|\boldsymbol{\theta}\right\|^2 + (1-\alpha)\left\|\boldsymbol{\theta'}\right\|^2 - (\alpha^2 \left\|\boldsymbol{\theta}\right\|^2 + 2\alpha(1-\alpha)\boldsymbol{\theta}^\top\boldsymbol{\theta'} + (1-\alpha)^2\left\|\boldsymbol{\theta'}\right\|^2) \\ &= \alpha(1-\alpha)\left(\left\|\boldsymbol{\theta}\right\|^2 - 2\boldsymbol{\theta}^\top\boldsymbol{\theta'} + \left\|\boldsymbol{\theta'}\right\|^2\right) \\ &= \alpha(1-\alpha)\left\|\boldsymbol{\theta} - \boldsymbol{\theta'}\right\|^2 \\ &\geq 0. \end{aligned}$$

■

2.3 Stochastic Gradient Descent

Recall that the loss function on a training data $\{\boldsymbol{x}^i, y^i\}_{i=1}^n$ can be decomposed as the average of loss on the n data points:

$$L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(\boldsymbol{\theta}),$$

where $\ell_i(\boldsymbol{\theta})$ is the loss on the i-th point, e.g., in the case of square loss, $\ell_i(\boldsymbol{\theta}) = (y^i - f(\boldsymbol{x}^i,\ \boldsymbol{\theta}))^2$. Related, the gradient is also the average of n terms:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta}).$$

This suggests that the complexity of calculating the gradient is $O(n)$. Therefore, if the data size n is very large, gradient descent can be become very slow.

Stochastic gradient descent provides a way to speed up the gradient descent in big data (large n) settings, by approximating the exact gradient by only averaging over a randomly sampled subset $\mathcal{M}$ of data points, that is,

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \approx \frac{1}{|\mathcal{M}|}\sum_{i\in\mathcal{M}} \nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta}),$$

where $\mathcal{M}$, called a mini-batch, is a subset of data points randomly drawn from the original data-set.

Algorithm 2 Stochastic Gradient Descent

Goal: Mininize loss function $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell_i(\boldsymbol{\theta})/n$.
Initialize $\boldsymbol{\theta}_0$
for Iteration t **do**
Draw a minibatch $\mathcal{M} \subset \{1, \ldots, n\}$ randomly. Update the parameter by

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \epsilon_t \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} \nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta}),$$

where ϵ_t is a stepsize.
end for

When data size n is very large, stochastic gradient is both faster and better.

1. It is unnecessary to eliminate the whole dataset to calculate the exact gradient because the gradient is used to move forward only a small step.
2. It is not necessary to solve the optimization in full precision, because the data is noisy anyway. The exact minimizer of loss $L(\boldsymbol{\theta})$ may not necessarily outperform decent approximation minimizers.
3. Introducing stochasticity help prevent overfitting.

Stepsize Selection

The choice of step size plays a critical role for practice performance and is an active research area. Beyond the simple method of choosing a "small" constant step size (which need to be decided by tunnning in practice), there has been some adaptive methods for tuning learning rates proposed recently, including adagrad, adam, etc. See this post for a overview: `http://sebastianruder.com/optimizing-gradient-descent/`.

2.4 Bias and Variance of SGD

Let us build some more intuitive understanding of SGD by considering the following simple objective function:

$$\min_{\theta} L(\theta) = \frac{1}{2}\mathbb{E}_{y \sim \mathcal{D}}[(\theta - y)^2].$$

This is equivalent to fitting a simple "constant predictor" $y = \theta$. Obviously, $L(\theta)$ is a simple quadratic function,

$$L(\theta) = \frac{1}{2}\left(\theta^2 - 2\mu\theta + v\right),$$

where we denote $\mu = \mathbb{E}_{\mathcal{D}}[y]$ and $v = \mathbb{E}_{\mathcal{D}}[y^2]$. The optimal solution is hence $\theta^* = \mu$.

Assume we run SGD with a mini-batch of size 1 starting from an initial value θ^0. The update of SGD is

$$\begin{aligned} &\text{Draw random sample } y_t \sim \mathcal{D}, \\ &\theta_{t+1} = \theta_t - \epsilon_t(\theta_t - y_t) \\ &\qquad = (1 - \epsilon_t)\theta_t + \epsilon_t y_t. \end{aligned}$$

where at each iteration t, we draw a simple sample y_t from the data distribution. Here ϵ_t is the step-size at the t-th iteration. This defines a recursive update in which θ_{t+1} is a weighted average of θ_t and y_t.

Decaying Step Size

If we choose $\epsilon_t = \frac{1}{t+2}$, the recursive formula can be reduced to a simple averaging formula that we can analyze easily. We have

$$\theta_{t+1} = \frac{t+1}{t+2}\theta_t + \frac{1}{t+2}y_t.$$

The first few numbers of updates look like

$$\begin{aligned}
\theta_1 &= \frac{1}{2}\theta_0 + \frac{1}{2}y_0 \\
\theta_2 &= \frac{2}{3}\theta_1 + \frac{1}{3}y_1 = \frac{1}{3}(\theta_0 + y_0 + y_1) \\
\theta_3 &= \frac{3}{4}\theta_2 + \frac{1}{4}y_2 = \frac{1}{4}(\theta_0 + y_0 + y_1 + y_2) \\
&\vdots
\end{aligned}$$

In general, we can show

$$\theta_t = \frac{\theta_0 + y_0 + y_1 + \cdots y_{t-1}}{t+1}. \tag{2.3}$$

Therefore, SGD can be viewed as a simple average (but computed in a recursive fashion) of the initial value of θ_0 and the sample y_t at different iterations. Using this, we can calculate the mean square error w.r.t. the true value θ^* at iteration t. Recall the bias-variance decomposition:

$$\mathrm{MSE}(\theta_t) := \mathbb{E}[(\theta_t - \theta^*)^2] = \underbrace{\mathbb{E}\left[(\theta_t - \mathbb{E}[\theta_t])^2\right]}_{\mathrm{Var}(\theta_t)} + \underbrace{(\mathbb{E}[\theta_t] - \theta^*)^2}_{(\mathrm{bias}(\theta_t))^2}.$$

Theorem 2.4 For θ_t in (2.3), we have

$$\mathrm{Var}(\theta_t) = \frac{1}{(t+1)^2}(\sigma_0^2 + t\sigma^2) \qquad\qquad \mathrm{bias}(\theta_t) = \frac{b_0}{t+1},$$

where $\sigma^2 = \mathrm{Var}(y_t)$, and $\sigma_0^2 = \mathrm{Var}(\theta_0)$ and $b_0 = \mathbb{E}[\theta_0] - \theta^*$ are the variance and bias of the initialization θ_0, respectively. Therefore, both the variance and bias decays to zero with a rate of $O(1/(t+1))$. The overall MSE is

$$\mathbb{E}[(\theta_t - \theta^*)^2] = \frac{\sigma^2}{t+1} + \frac{b_0^2 + \sigma_0^2}{(t+1)^2},$$

which also decay to zero as $t \to +\infty$. This suggests that θ_t will converge to the optimum point θ^* in probability.[a]

[a] Recall that θ_t converges to θ^* in probability if for any $\epsilon > 0$, we have $\lim_{t\to+\infty} \Pr(|\theta_t - \theta^*| \geq \epsilon) = 0$. This is implied by the convergence in MSE, that is, $\lim_{t\to+\infty} \mathbb{E}[(\theta_t - \theta^*)^2] = 0$, because, by Chebyshev's inequality, $\Pr(|\theta_t - \theta^*| \geq \epsilon) \leq \frac{\mathbb{E}[(\theta_t - \theta^*)^2]}{\epsilon^2}$ for any $\epsilon > 0$

Proof. Let us calculate the variance first:

$$\begin{aligned}
\mathrm{Var}(\theta_t) &= \mathrm{Var}\left(\frac{\theta_0 + y_0 + y_1 + \cdots y_{t-1}}{t+1}\right) \\
&= \frac{1}{(t+1)^2}\left(\mathrm{Var}(\theta_0 + y_0 + y_1 + \cdots y_{t-1})\right) \\
&= \frac{1}{(t+1)^2}\left(\mathrm{Var}(\theta_0) + \mathrm{Var}(y_0) + \mathrm{Var}(y_1) + \cdots + \mathrm{Var}(y_{t-1})\right) \qquad \text{//for independence} \\
&= \frac{1}{(t+1)^2}\left(\sigma_0^2 + t\sigma^2\right) \le \frac{\max(\sigma_0^2, \sigma^2)}{t+1},
\end{aligned}$$

where we assume $\sigma_0^2 = \mathrm{Var}(\theta_0)$ and $\sigma^t = \mathrm{Var}(y_t)$. Therefore, the variance of θ_t decays with a rate of $O(1/(t+1))$ as the iteration step increases.

The bias equals

$$\begin{aligned}
\mathrm{bias}(\theta_t) &= \mathbb{E}[\theta_t] - \theta^* \\
&= \mathbb{E}\left[\frac{\theta_0 + y_0 + y_1 + \cdots y_{t-1}}{t+1}\right] - \theta^* \\
&= \frac{1}{t+1}(\mathbb{E}[\theta_0] - \theta^*), \qquad \text{//because } \mathbb{E}[y_t] = \mu = \theta^*,
\end{aligned}$$

where $\mathbb{E}[\theta_0] - \theta^*$ denotes the initial bias introduced by the initialization, and the bias decays at a rate of $O(1/(t+1))$.

Overall, the MSE equals

$$\mathbb{E}[(\theta_t - \theta^*)^2] = \frac{\sigma_m^2}{t+1} + \frac{b_0^2}{(t+1)^2},$$

where $\sigma_m^2 := \max(\sigma_0^2, \sigma^2)$ and $b_0 = \mathbb{E}[\theta_0] - \theta^*$. ■

Constant Step Size

Assume the stepsize is taking to be a constant, that is, $\epsilon_t = c$ for small some constant c. In this case, we have

$$\theta_{t+1} = (1-c)\theta_t + cy_t.$$

The first few updates look like

$$\begin{aligned}
\theta_1 &= (1-c)\theta_0 + cy_0 \\
\theta_2 &= (1-c)\theta_1 + cy_1 = (1-c)^2\theta_0 + c(1-c)y_0 + cy_1 \\
\theta_3 &= (1-c)\theta_2 + cy_2 = (1-c)^3\theta_0 + c(1-c)^2 y_0 + c(1-c)y_1 + cy_2.
\end{aligned}$$

In general, we have

$$\begin{aligned}
\theta_t &= (1-c)^t\theta_0 + c(1-c)^{t-1}y_0 + \cdots + c(1-c)y_{t-2} + cy_{t-1} \\
&= (1-c)^t\theta_0 + \sum_{i=0}^{t-1} c(1-t)^i y_{t-i-1}.
\end{aligned} \tag{2.4}$$

Theorem 2.5 Assume $c \in (0,1)$. For θ_t in (2.4), we have

$$\begin{aligned}
\mathrm{bias}(\theta_t) &= \mathbb{E}[\theta_t] - \theta_* = (1-c)^t(\mathbb{E}\left[\theta_0\right] - \theta^*), \\
\mathrm{Var}(\theta_t) &= \mathbb{E}[(\theta_t - \mathbb{E}[\theta_t])^2] = (1-c)^{2t}\sigma_0^2 + c^2\sigma^2\frac{1-(1-c)^{2t}}{1-(1-c)^2}.
\end{aligned}$$

From this, we can see that the bias decays to zero exponentially fast as $t \to +\infty$, but the variance never vanish:

$$\lim_{t\to+\infty} \text{bias}(\theta_t) = 0, \qquad \lim_{t\to+\infty} \text{Var}(\theta_t) = \frac{c\sigma^2}{2-c} > 0.$$

This is because the step size is kept constant and never decays, which allows us to move θ_t towards the optimum θ^* quickly, but it will never converge to a definite number in this case, due to the random fluctuation from the sampled data.

Proof. Let us calculate the bias:

$$\begin{aligned}
\text{bias}(\theta_t) &= \mathbb{E}[\theta_t] - \theta_* \\
&= \mathbb{E}\left[(1-c)^t\theta_0 + \sum_{i=0}^{t-1} c(1-t)^i y_{t-i-1}\right] - \theta^* \\
&= (1-c)^t\mathbb{E}\left[\theta_0\right] + \sum_{i=0}^{t-1} c(1-t)^i\mathbb{E}\left[y_{t-i-1}\right] - \theta^* \\
&= (1-c)^t\mathbb{E}\left[\theta_0\right] + \sum_{i=0}^{t-1} c(1-t)^i\theta_* - \theta^* \qquad \text{//because } \mathbb{E}[y_i] = \mu = \theta^* \\
&= (1-c)^t\mathbb{E}\left[\theta_0\right] + c\frac{1-(1-c)^t}{1-(1-c)}\theta_* - \theta_* \qquad \text{//Use Lemma 2.6 with } a = (1-c) \\
&= (1-c)^t\mathbb{E}\left[\theta_0\right] + (1-(1-c)^t)\theta_* - \theta_* \\
&= (1-c)^t(\mathbb{E}\left[\theta_0\right] - \theta^*).
\end{aligned}$$

Therefore, when $0 < c < 1$, the bias decays to zero exponentially fast.

$$\lim_{t\to+\infty} \text{bias}(\theta_t) = \lim_{t\to+\infty} (1-c)^t(\mathbb{E}\left[\theta_0\right] - \theta^*) = 0.$$

For the variance:

$$\begin{aligned}
\text{Var}(\theta_t) &= \text{Var}\left((1-c)^t\theta_0 + \sum_{i=0}^{t-1} c(1-t)^i y_{t-i-1}\right) \\
&= \text{Var}((1-c)^t\theta_0) + \sum_{i=0}^{t-1} \text{Var}(c(1-t)^i y_{t-i-1}) \\
&= (1-c)^{2t}\sigma_0^2 + \sum_{i=0}^{t-1} c^2(1-c)^{2i}\sigma^2 \\
&= (1-c)^{2t}\sigma_0^2 + c^2\sigma^2\sum_{i=0}^{t-1}(1-c)^{2i} \\
&= (1-c)^{2t}\sigma_0^2 + c^2\sigma^2\frac{1-(1-c)^{2t}}{1-(1-c)^2} \qquad \text{//Use Lemma 2.6 with } a = (1-c)^2.
\end{aligned}$$

Assume $0 < c < 1$, we have

$$\lim_{t\to+\infty} \text{Var}(\theta_t) = \frac{c^2\sigma^2}{1-(1-c)^2} = \frac{c\sigma^2}{2-c} > 0.$$

Therefore, the variance never vanishes when using constant step size. The algorithm does not converge to a definite number and will keep fluctuate forever due to the random noise in the sampled data. ■

Lemma 2.6 For $a \in (0,1)$, we have

$$\sum_{i=0}^{t-1} a^i = 1 + a + a^2 + \cdots + a^t = \frac{1-a^t}{1-a}.$$

Proof. Note that

$$\begin{aligned}(1-a)(1+a+a^2+\cdots+a^{t-1}) &= (1+a+a^2+\cdots+a^{t-1}) - (a+a^2+\cdots+a^t) \\ &= 1-a^t.\end{aligned}$$

■

3. Classification With Surrogate Loss

Classification is the problem of predicting discrete labels. Similar to the case of regression, we are given a dataset of feature-label pairs, say $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^n$, where the labels $y^{(i)}$ are discrete valued variables, that is, $y^{(i)} \in \mathcal{Y}$ for some set $\mathcal{Y}$ with a finite or countable number of elements. In comparison, regression is the case when the labels are continuous variables, e.g., $\mathcal{Y} = \mathbb{R}$. In both regression and classification, the domain of the features, denoted by $\mathcal{X}$, is assumed to be a subset of $\mathbb{R}^d$.

The goal is again to find a prediction function $f : \mathcal{X} \to \mathcal{Y}$, called a **classifier**, to predict the labels y with the features $\boldsymbol{x}$, such that $y \approx f(\boldsymbol{x})$.

We focus on binary classification in this section, which is the case when $\mathcal{Y}$ contains two elements. Without loss of generality, we can assume $\mathcal{Y} = \{-1, +1\}$. A natural choice of classifier would be

$$f(\boldsymbol{x};\ \boldsymbol{\theta}) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b),$$

where $\boldsymbol{\theta} := [\boldsymbol{w}, b]$ are parameters that we need to decide using different methods; the $\text{sign}(\cdot)$ denotes the sign function, that is, $\text{sign}(x) = 1$ if $x > 0$, and $\text{sign}(x) = -1$ if $x \geq 0$.

This classifier is called ***a linear classifier***, because its decision boundary, which consists of the points on line $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$, is a line line or plane. But note that $f(\boldsymbol{x};\ \boldsymbol{\theta})$ is not a linear function of $\boldsymbol{x}$ itself, because of the sign function.

■ **Example 3.1** Consider a simple example of weather prediction, in which $y^{(i)}$ may represent whether it will rain on day i (e.g., we may label 1 as raining and -1 as sunny), and $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ represents features such as temperature, humidity and wind scale and historical data.

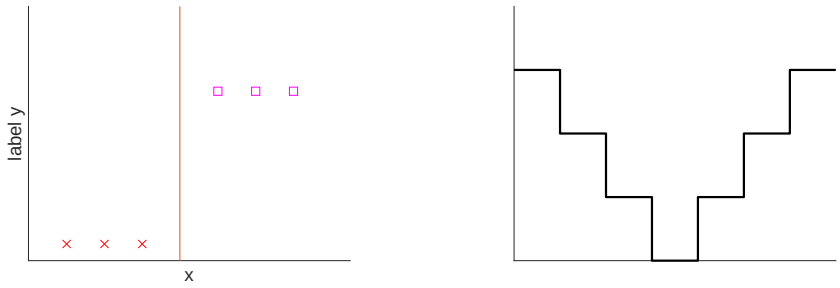
0-1 Loss for Classification is Dis-continuous

Ideally, the parameters $\boldsymbol{\theta} = [\boldsymbol{w}, b]$ should be chosen to minimize the number of mistakes in predicting the labels. This yields following 0/1 loss function:

$$L_{0/1}(\boldsymbol{\theta}) = \sum_{i=1}^{n} \mathbb{I}(f(\boldsymbol{x}^{(i)};\ \boldsymbol{\theta}) \neq y^{(i)}),$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. The 0/1 loss simply counts the number of mistakes made in predicting the labels.

A naïve idea for finding optimal $\boldsymbol{\theta}$ would be to minimize $L_{0/1}(\boldsymbol{\theta})$. Unfortunately, this turns out the be both difficult and undesirable. This is because $L_{0/1}(\boldsymbol{\theta})$ is a piecewise constant function that can not be easily optimized using typical numerical optimization methods like gradient descent.



(a) (b)

Figure 3.1: (a) Data points (x vs. y). (b) The loss function (# of mistakes) as the decision boundary moves.

Figure 3.1(a) shows a toy binary classification problem, where the cross points ($\times$) represent training data points with negative labels ($y = -1$) and the squares ($\square$) represent positive labels ($y = 1$). A linear classifier in this case corresponds to a vertical line shown in the figure that separates these two types of points. In this case, the decision rule has a simple form of

$$\hat{y} = \text{sign}(x - b),$$

where b denotes the location of the boundary. Related, the 0-1 loss function, which counts the number of mistakes by the classifier, can be written into

$$L(b) = \sum_{i=1}^{n} \mathbb{I}(\text{sign}(x^{(i)} - b) = y^{(i)}),$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. As show in Figure 3.1(b), the loss $L(b)$ is a dis-continuous step function. This is because the number of mistakes increase or decrease by one when the boundary comes across a data point.

However, the dis-continuity of the loss causes difficulties in both optimization and generalizability.

1) It is hard to numerically minimize a discontinuous function to find the optimal parameters (e.g., consider what happens in this case if you minimize this function using gradient descent).

2) Many different values of b have exactly the same number of mistakes, and it is unclear which one to pick by just looking at the 0-1 loss.

3.1 Surrogate Loss Functions

Almost all the practical classification algorithms are based on replacing the 0-1 loss with some "nicer" surrogate loss functions that are similar to 0-1 loss but easier to optimize.

To fix ideas, assume we classify by a model $y = \text{sign}(f(\boldsymbol{x},\ \boldsymbol{\theta}))$. For the class of linear classifiers, we have $f(\boldsymbol{x},\ \boldsymbol{\theta}) = \boldsymbol{w}^\top \boldsymbol{x} + b,\ \ \theta = [\boldsymbol{w}, b]$. Let us write down the expression of the 0-1 loss, which equals the number of mis-classified instances:

$$L_{0/1}(\theta) = \sum_{i=1}^{n} \Gamma_{0/1}(y^{(i)} f(\boldsymbol{x}^{(i)},\ \boldsymbol{\theta})),$$

where $\Gamma_{0/1}$ is a step function:

$$\Gamma_{0/1}(t) = \begin{cases} 1 & t \leq 0 \\ 0 & t > 0. \end{cases}$$

Here we can see that the discountinuousity of the loss function mainly comes from $\Gamma_{0/1}$. Therefore, we may replace $\Gamma_{0/1}$ with various continuous surrogate functions, yielding different classification algorithms.

1. Consider replacing $\Gamma_{0/1}$ with the following function:

   $$\Gamma_{hinge}(t) = \max(0,\ 1 - t).$$

   It is called Hinge Loss. Correspondingly, the loss function is

   $$L_{hinge}(\theta) = \sum_{i=1}^{n} \max(0,\ 1 - y^{(i)} f(\boldsymbol{x}^{(i)},\ \boldsymbol{\theta})).$$

   Methods that minimize the hinge loss are called large margin classifiers, or support vector machine (SVM). It is common to combine it with a L2 regularization, for example, in the case of linear classifiers, SVM that are implemented in practice is often

   $$\min_{\boldsymbol{w},b} \sum_{i=1}^{n} \max(0,\ 1 - y^{(i)} f(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)) \ +\ \alpha \left\|\boldsymbol{w}\right\|_2^2,$$

   where α is a regularization coefficient.
2. Consider replacing $\Gamma_{0/1}$ with the following *log-sum-exp* function:

   $$\Gamma_{log}(t) = \log(1 + \exp(-t)).$$

   Correspondingly, the loss function is

   $$L_{log}(\theta) = \sum_{i=1}^{n} \log(1 + \exp(-y^{(i)} f(\boldsymbol{x}^{(i)},\ \boldsymbol{\theta}))).$$

   Methods that minimize this loss are called logistic regression. It can be also be motivated using probabilistic modeling.

Exercise 3.1 Consider a simple data set of $(0, -1)$, $(1, +1)$, $(2, +1)$. Please plot its 0-1 loss, Hinge loss and log-sum-exp loss. Plot the corresponding optimal decision boundaries derived. ■

Exercise 3.2 Consider a more general class of surrogate loss:

$$\Gamma_\alpha(t) = \alpha \log(1 + \exp((1 - t)/\alpha)).$$

Show that it is equivalent to hinge loss when $\alpha \to 0^+$. ■

Exercise 3.3 Consider using the following logistic function as the surrogate loss:

$$\Gamma(t) = \frac{1}{1+\exp(t)}.$$

Do you think it is a good idea? Please elaborate. ■

3.2 Support Vector Machine

Here we provide an alternative derivation of support vector machine, via finding maximum margin decision boundary.

Margin

Support Vector Machines (SVMs) are based on the intuitive idea that we should select the decision boundary that leaves a large margin between the data points and the decision boundary, so the decision is robust against a small variation on the data feature.

In order to frame this mathematically, we need to consider the geometric distance (referred to as the margin) between points and lines. We illustrate the idea using linear decision boundaries:

$$\hat{y} = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b),$$

where $\boldsymbol{w}$, b are parameters to solve.

We formalize the notions of the geometric margins. Given a point $\boldsymbol{x}$ and a linear decision boundary characterized by $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$. Based on some simple geometry, we can show that the distance between a given data point $\boldsymbol{x}^0$ and the linear boundary is

$$\text{margin}_{\boldsymbol{w},b}[\boldsymbol{x}^{(i)}] = \frac{\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b}{\|\boldsymbol{w}\|} = \begin{cases} \frac{\boldsymbol{w}^\top \boldsymbol{x}^{(i)}+b}{\|\boldsymbol{w}\|}, & \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b > 0 \\ -\frac{\boldsymbol{w}^\top \boldsymbol{x}^{(i)}+b}{\|\boldsymbol{w}\|}, & \boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b \leq 0 \end{cases},$$

where $\|\boldsymbol{w}\| = \sqrt{\sum_{j=1}^d w_j^2}$ is the length of vector $\boldsymbol{w}$.

Now given a *labeled* point $(\boldsymbol{x}^{(i)}, y^{(i)})$, we may use the following "signed margin", sometimes called *functional margin*, to measure its consistency with the linear boundary:

$$\text{margin}_{\boldsymbol{w},b}[(\boldsymbol{x}^{(i)}, y^{(i)})] = \frac{y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)}{\|\boldsymbol{w}\|}.$$

Note that if the boundary correctly classifies point $(\boldsymbol{x}^{(i)}, y^{(i)})$, that is, $y^{(i)} = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)$, then the signed margin equals the geometric margin (the actual distance), this is because

$$\begin{aligned} \text{margin}_{\boldsymbol{w},b}[(\boldsymbol{x}^{(i)}, y^{(i)})] &= \frac{y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)}{\|\boldsymbol{w}\|} \\ &= \frac{\text{sign}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)}{\|\boldsymbol{w}\|} \\ &= \frac{|\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b|}{\|\boldsymbol{w}\|} \\ &= \text{margin}_{\boldsymbol{w},b}[\boldsymbol{x}^{(i)}]. \end{aligned}$$

On the other hand, if the boundary classifies the point incorrectly, that is, $y^{(i)} = -\text{sign}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)$, then the signed margin is the negative of the actual geometric distance:

$$\text{margin}_{\boldsymbol{w},b}[(\boldsymbol{x}^{(i)}, y^{(i)})] = -\text{margin}_{\boldsymbol{w},b}[\boldsymbol{x}^{(i)}].$$

We want to have the signed margin to be positive (so it correctly classify the points), and also have large magnitude (so the points are far away from the decision boundary).

Linearly Separable Case

We first consider the case the dataset is linearly separable, that is, you can draw a line (or hyperplane) to separate the two types of points. Formally, this means that there must exists some values of $\boldsymbol{w}$ and b, such that $y^{(i)} = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)$, which is equivalent to saying that

$$y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) > 0, \quad i \in \{1, \dots, n\}.$$

Intuitively we want to draw a decision boundary to separate two kinds of data points, so it makes sense that the distances of all data points should be far away from the boundary, thus we can maximize the distance between nearest data point and the decision boundary. So the target loss function is

$$\max_{\boldsymbol{w},b} \min_{1 \le i \le n} \frac{y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)}{\|\boldsymbol{w}\|}.$$

Another way to write this is

$$\max_{\boldsymbol{w},b} F(\boldsymbol{w}, b) \quad \text{where} \quad F(\boldsymbol{w}, b) = \min_{1 \le i \le n} \frac{y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)}{\|\boldsymbol{w}\|}.$$

where $\min_{1 \le i \le n}$ is a part of an objective function $F(\boldsymbol{w}, b)$, for which we want to minimize w.r.t. $\boldsymbol{w}$ and b.

We can reform this optimization into another form that is easier to solve. Note that you can multiple $\boldsymbol{w}$ and b by any positive constant without changing the decision boundary, that is, $\hat{y} = \text{sign}(100\boldsymbol{w}^\top \boldsymbol{x} + 100b)$ defines the same linear classifier as $\hat{y} = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$. Assume the boundary have already classify all the points correctly, that is, we have $y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) > 0$, then we can multiple $\boldsymbol{w}$ and b by a positive constant such that

$$\min_i y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) = 1,$$

in which case the objective function becomes

$$F(\boldsymbol{w}, b) = \min_{1 \le i \le n} \frac{y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)}{\|\boldsymbol{w}\|} = \frac{1}{||\boldsymbol{w}||}.$$

Therefore, we can reform the optimization into

$$\max_{\boldsymbol{w},b} \frac{1}{\|\boldsymbol{w}\|},$$
$$\text{s.t} \quad y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \ge 1, \quad \forall i \in \{1, \dots, n\}.$$

Because

$$\max_{\boldsymbol{w},b} \frac{1}{\|\boldsymbol{w}\|} \iff \min_{\boldsymbol{w},b} \|\boldsymbol{w}\| \iff \min_{\boldsymbol{w},b} \|\boldsymbol{w}\|^2,$$

we can further reform it into

$$\min_{\boldsymbol{w},b} \|\boldsymbol{w}\|^2,$$
$$\text{s.t} \quad y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \ge 1, \quad \forall i \in \{1, \dots, n\}.$$

This is relatively easy to solve because it is a quadratic programing (quadratic objective function + linear constraints). Scientific softwares such as MATLAB provide functions for solving it.

Non-linearly Separable Case

However, datasets in practice are not always linearly separable. To handle the non-separable cases, we can introduce another variable ϵ_i, associated with each data point i, which control the amount of "violation of the margin constraint",

$$y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \geq 1 - \epsilon_i, \tag{3.1}$$

$$\epsilon_i \geq 0, \tag{3.2}$$

where ϵ_i is also called **slack variable**. If a data point is classified at the wrong "side" of the decision boundary, then $y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) < 0$, so the value of ϵ_i should be very large to guarantee equation (3.1). But we also don't want too many of such data points, so we can add a penalty in the loss function:

$$\min_{\boldsymbol{w},b,\epsilon_i} \Big(\|\boldsymbol{w}\|^2 + R\sum_{i=1}^{n} \epsilon_i \Big), \tag{3.3}$$

$$\text{s.t} \quad y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \geq 1 - \epsilon_i, \tag{3.4}$$

$$\epsilon_i \geq 0, \quad \forall i \in \{1, \ldots, n\}, \tag{3.5}$$

where R is a constant that control the scale of the penalty term.

This constraint optimization can be further simplified. Because variable ϵ_i satisfies

$$\begin{cases} \epsilon_i \geq 0 \\ \epsilon_i \geq 1 - y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b) \end{cases} \quad \forall i \in \{1, \ldots, n\},$$

and we want ϵ_i to be as small as possible, the optimal ϵ_i should be taken to be

$$\epsilon_i = \max\big\{1 - y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b), \quad 0\big\}. \tag{3.6}$$

Replace ϵ_i with equation (3.6) in equation (3.3), we have

$$\min_{\boldsymbol{w},b} \Big(\|\boldsymbol{w}\|^2 + R\sum_{i=1}^{n} H\big(y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)\big) \Big), \tag{3.7}$$

where $H(t) = \max\big\{0, 1-t\big\}$ is called **hinge loss**. Hinge loss is usually used for training classifiers. By multiplying $\alpha = 1/R$ in equation (3.7), we have

$$\min_{\boldsymbol{w},b} \Big(\alpha \|\boldsymbol{w}\|^2 + \sum_{i=1}^{n} H\big(y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)\big) \Big), \tag{3.8}$$

which is the loss function we used for linear SVMs(also called max margin classifiers). Intuitively $H\big(y^{(i)}(\boldsymbol{w}^\top \boldsymbol{x}^{(i)} + b)\big)$ can be considered as the penalty that data points are in the wrong "side" of the decision boundary and term $\alpha \|\boldsymbol{w}\|^2$ can be seen as the regularization term we introduced before.

The hinge lose function can be viewed as a continuous surrogate function of the "ideal" 0-1 loss function corresponds to the exact training 0-1 error. Define the 0-1 loss function to be

$$\mathbb{I}(t) = \begin{cases} 1 & \text{if } t \leq 0 \\ 0 & \text{if } t > 0. \end{cases}$$

Then the number of mistakes (0-1 error) of the classifier on the training set is

$$L_{0\text{-}1}(\boldsymbol{w},\ b) = \sum_{i=1}^{n} \mathbb{I}(\boldsymbol{w}^\top x^{(i)} + b).$$

However, we mentioned in the class that it is difficult to directly minimize this loss function because it is a step function, whose gradient is either zero or infinite, and one can not use algorithms like gradient descent to minimize it in order to find the optimal $\boldsymbol{w}$ and b.

From the above derivation, we can view SVM as replacing the 0-1 loss with the hinge loss function (which is continuous, almost differentiable, and convex!), that makes it easier to optimize.

4. Maximum Likelihood and Bayesian Inference

4.1 Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is a fundamental method of estimating the parameters of probability distributions observed through empirical data. The idea is to search for the parameters and models under which the observed data is most probable, hence maximizing the likelihood.

Let us explain the key idea using the example below. If you need to review basics in probability, please check Appendix A.1.

■ **Example 4.1 — Guessing the coin.** Assume we are playing a guessing game with a friend. The friend has two coins at hand:

Coin A: a fair coin with equal probability of being head (H) or tail (T), that is,

$$\Pr(X = H) = \Pr(X = T) = 1/2.$$

Coin B: a biased coin with a probability of 0.9 being head (H), that is,

$$\Pr(X = H) = 1 - \Pr(X = T) = 0.9.$$

The friend secretly picked one of the coins, randomly threw it a few times, and get a sequence of $HHHHTT$ (all of which come from the same coin she picked). Could you guess which coin your friend picked?

Obviously, we can not tell for sure, since coin throwing is random and both cases are possible. What we can do it at most is to decide which case is *more likely* given the information at hand (which is the sequence $HHHHTT$).

Maximum likelihood estimation provides a quantitative tool for solving estimation problems based on random observations like this. The idea is to evaluate the probability of seeing the observation (called the *likelihood*) under each possible case (called the *hypothesis*), and choose the case that yields the maximum likelihood as an optimal estimation.

The reasoning of maximum likelihood estimation for the game above goes as follows:

- If coin A was used, what is the probability that the sequence $HHHHTT$ was observed? Because the probability of both H and T are 0.5 under coin A, the probability should be

$$\text{Likelihood}(coin\ A) := \Pr(HHHHTT \mid coin\ A) = (0.5)^6 \approx 0.0156,$$

  where we call the probability of seeing the observation under coin A the likelihood of the hypothesis A.
- If coin B was used, the probability of observing sequence $HHHHTT$ is

$$\text{Likelihood}(coin\ B) := \Pr(HHHHTT \mid coin\ B) = (0.9)^4 \times (0.1)^2 \approx 0.00656.$$

  This is because H (whose probability is 0.9) happens four times, and T (whose probability is 0.1) happens two times.

Because

$$\text{Likelihood}(coin\ A) > \text{Likelihood}(coin\ B),$$

coin A is more likely to be coin that was actually chosen. This is not obvious without the calculation we had, because one may feel coin B appears more likely because the data includes more H than T. But if you think more carefully, it makes a lot of sense — If it were coin B, we would have observed less T, because it is expected to happen only once every ten throws (while we observe two out of six).

If we observe $HHHHHT$ (only one T out of six throws), which coin is more likely? Let us do the calculate:

$$\text{Likelihood}(coin\ A) := \Pr(HHHHHT \mid coin\ A) = (0.5)^6 \approx 0.0156 \qquad \text{//the same as before,}$$
$$\text{Likelihood}(coin\ B) := \Pr(HHHHHT \mid coin\ B) = (0.9)^5 \times (0.1)^1 \approx 0.059.$$

We have $\text{Likelihood}(coin\ B) > \text{Likelihood}(coin\ A)$ and should choose coin B this time. We can see the power of quantitative calculation for problems like this. Intuition and casual guessing do not serve as a rigours tool for solving problems that need to reason with uncertainty and randomness.

Maximum Likelihood Estimation

Consider the general problem of *parameter estimation*: given a set of observation $\{x^{(i)}\}_{i=1}^n$, assumed to be drawn **independent and identically distributed (i.i.d.)** from a distribution P_{θ^*}, where θ^* is an unknown parameter that belongs to some set Θ. We want to estimate θ^* using the information from observation $\{x^{(i)}\}$. Maximum likelihood estimator solves this by seeking the parameter under which the probability of seeing the data is maximized:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} L(\theta), \qquad L(\theta) = P\left(x^{(1)}, \ldots, x^{(n)} \mid \theta\right) = \prod_{i=1}^{n} P\left(x^{(i)} \mid \theta\right),$$

where the joint probability equals the product of individual probabilities because the observations are assumed to be generated identically and independently (i.i.d). This probability, denoted by $L(\theta)$, is called the **likelihood function** of θ (when the data is viewed as a fixed constant), and its maximum point $\hat{\theta}$ is called the **maximum likelihood estimator** of θ^*.

In practice, it is often easier to consider the **log-likelihood function**:

$$\ell(\theta) := \log L(\theta) = \sum_{i=1}^{n} \log P(x^{(i)} \mid \theta).$$

Because $\log(\cdot)$ is a strictly increasing function, the maximum points $\hat{\theta}$ that maximize $\ell(\theta)$ also maximize $L(\theta)$ (and vice versa). Therefore, one can find the maximum likelihood estimator by maximizing $\ell(\theta)$:

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \ell(\theta) = \arg\max_{\theta \in \Theta} L(\theta).$$

■ **Example 4.2 — Biased Coin.** Assume there is a biased coin whose probability of being head (H) is θ, and the probability of tail (T) is $1-\theta$. Here θ must belong to interval $[0,1]$. Assume we observe a sequence $\{HHHHTT\}$. The question is to estimate θ.

In this case, the likelihood function (i.e., the probability of seeing observation $\{HHHHTT\}$, given a particular value of θ) is

$$\begin{aligned} L(\theta) &= P(\{HHHHTT\} \mid \theta) \\ &= P(H \mid \theta)^{m_H} P(T \mid \theta)^{m_T} \\ &= \theta^{m_H}(1-\theta)^{m_T}, \end{aligned}$$

where m_H and m_T denote the number of H and T in the dataset, respectively. We have $m_H = 4$ and $m_T = 2$ in our case.

To find the best θ, it is more convenient to work with the log-likelihood function:

$$\hat{\theta} = \arg\max_{\theta \in [0,1]} \ell(\theta), \qquad \text{where} \qquad \ell(\theta) = \log L(\theta) = m_H \log\theta + m_T \log(1-\theta).$$

When $m_H > 0$ and $m_T > 0$, the optimal $\hat{\theta}$ obviously does not equal 0 or 1, since we would have $\ell(\theta) = -\infty$ if $\theta = 0$ or 1. This means the optimal point $\hat{\theta}$ is strictly inside the open interval $(0,1)$, and hence should have a zero gradient,

$$\nabla \ell(\hat{\theta}) = \frac{m_H}{\hat{\theta}} - \frac{m_T}{1-\hat{\theta}} = 0,$$

yielding

$$\hat{\theta} = \frac{m_H}{m_H + m_T}.$$

This suggests that the maximum likelihood estimation of θ is simply counting the proportion of H in the dataset.

■ **Example 4.3 — Biased Coin Revisited.** The probability in Example 4.2 is parameterized by the probability itself, which should be constrained in interval $[0,1]$. This constraint may cause inconvenience sometimes. Alternatively, we can parameterize the probability with

$$\Pr(X = H) = \frac{\exp(\vartheta)}{1+\exp(\vartheta)}, \qquad \Pr(X = T) = \frac{1}{1+\exp(\vartheta)},$$

where any real-valued $\vartheta \in \mathbb{R}$ yields a properly normalized probability in interval $[0,1]$; it is related to the θ in Example 4.2 via

$$\theta = \frac{\exp(\vartheta)}{1+\exp(\vartheta)}, \qquad \text{or equivalently} \qquad \vartheta = \log\left(\frac{\theta}{1-\theta}\right),$$

assuming $\theta \neq 0$ or 1.

Using MLE on ϑ or θ will yield the equivalent estimation. But this transform is very useful in more complex models such as logisitic regression and neural networks.

Let us write down the log-likelihood function of ϑ on a dataset with m_H heads (H) and m_T

tails (T). We have

$$\begin{aligned}\ell(\vartheta) &= m_H \log \Pr(X = H) + m_T \log \Pr(X = T) \\ &= m_H \log \frac{\exp(\vartheta)}{1 + \exp(\vartheta)} + m_T \log \frac{1}{1 + \exp(\vartheta)} \\ &= m_H \vartheta - (m_H + m_T) \log(1 + \exp(\vartheta)).\end{aligned}$$

The MLE $\hat{\vartheta}$ should have a zero gradient:

$$\nabla \ell(\hat{\vartheta}) = m_H - (m_H + m_T) \frac{\exp(\hat{\vartheta})}{1 + \exp(\hat{\vartheta})} = 0.$$

Recall that $\theta = \exp(\vartheta)/(1 + \exp(\vartheta))$, we have

$$\hat{\theta} = \frac{\exp(\hat{\vartheta})}{1 + \exp(\hat{\vartheta})} = \frac{m_H}{m_H + m_T}.$$

This is equivalent to the result from Example 4.2. Solving $\hat{\vartheta}$ yields

$$\hat{\vartheta} = \log(m_H / m_T).$$

MLE with Density Functions

For distributions of continuous random variables (such as Gaussian distribution or uniform distributions on intervals), the maximum likelihood estimator should be defined by their probability density functions (PDF) (denoted by $p(x \mid \theta)$):

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} \left\{ L(\theta) := \prod_{i=1}^{n} p(x^{(i)} \mid \theta) \right\} \\ &= \arg\max_{\theta} \left\{ \ell(\theta) := \sum_{i=1}^{n} \log p(x^{(i)} \mid \theta) \right\}.\end{aligned}$$

■ **Example 4.4 — Univariate Gaussian Distribution.** Assume we observe a set of random numbers $\{x^{(1)}, x^{(2)}, \cdots, x^{(n)}\}$, which is assumed to be drawn i.i.d. from a Gaussian distribution $\mathcal{N}(\mu,\ \sigma^2)$ with unknown mean μ and variance σ^2, whose density function is

$$p(x \mid \mu,\ \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} (x - \mu)^2 \right).$$

The problem is to estimate μ and σ^2 using maximum likelihood estimator.

Writing down the log-likelihood function:

$$\begin{aligned}\ell(\mu,\ \sigma) &= \sum_{i=1}^{n} \log p(x^{(i)} \mid \mu, \sigma) \\ &= \sum_{i=1}^{n} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x^{(i)} - \mu)^2 \right) \right) \\ &= -n \log \sigma - n \log(\sqrt{2\pi}) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x^{(i)} - \mu)^2.\end{aligned}$$

The MLE $\hat{\mu}$ and $\hat{\sigma}$ should maximize $\ell(\mu,\ \sigma)$ and hence have a zero gradient:

$$\nabla_\mu \ell(\hat{\mu},\ \hat{\sigma}) = -\frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} (\hat{\mu} - x^{(i)}) = 0 \tag{4.1}$$

$$\nabla_\sigma \ell(\hat{\mu},\ \hat{\sigma}) = -\frac{n}{\hat{\sigma}} + \frac{1}{\hat{\sigma}^3} \sum_{i=1}^{n} (x^{(i)} - \hat{\mu})^2. \tag{4.2}$$

Solving (4.1) yields

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)},$$

that is, the estimation of the mean μ is the empirical average of the sample $\{x^{(i)}\}$.

Solving (4.2) yields

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (x^{(i)} - \hat{\mu})^2,$$

which is the empirical variance of the sample.

■ **Example 4.5 — Uniform Distribution.** Assume we observe a set of random numbers $\{x^{(1)}, x^{(2)}, \cdots, x^{(n)}\}$ i.i.d. drawn from a uniform distribution on interval $[-\theta,\ \theta]$:

$$p(x \mid \theta) = \frac{1}{2\theta} \mathbb{I}(x \in [-\theta,\ \theta]) = \begin{cases} \frac{1}{2\theta} & \text{if } x \in [-\theta,\ \theta] \\ 0 & \text{if otherwise,} \end{cases}$$

where $\mathbb{I}(\cdot)$ is the indicator function, with $\mathbb{I}(0) = 0$ and $\mathbb{I}(z) = 1$ for any $z \neq 0$. We can estimate

θ by MLE. The likelihood function is

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{n} p(x^{(i)} \mid \theta) \\ &= \frac{1}{(2\theta)^n} \prod_{i=1}^{n} \mathbb{I}(x^{(i)} \in [-\theta,\ \theta]) \\ &= \frac{1}{(2\theta)^n} \mathbb{I}\left(x^{(i)} \in [-\theta,\ \theta] \text{ holds for all } i = 1, \cdots n \right) \\ &= \frac{1}{(2\theta)^n} \mathbb{I}\left(\max_i |x^{(i)}| \leq \theta \right). \end{aligned}$$

Therefore, if $\theta < \max_i |x^{(i)}|$, we have $L(\theta) = 0$ (since it is impossible to have an $x^{(i)}$ outside of interval $[-\theta, \theta]$), and if $\theta \geq \max_i |x^{(i)}|$, we have $L(\theta) = 1/(2\theta)^n$, which decreases with θ. The value that maximizes θ should be hence

$$\hat{\theta} = \max_i |x^{(i)}|.$$

MLE for Regression

In regression, we are given a set of pairs $\{\boldsymbol{x}^{(i)},\ y^{(i)}\}_{i=1}^{n}$ of features and labels and are asked to construct a function that predicts y using $\boldsymbol{x}$. This is often solved by minimizing the mean square error (MSE) loss function. MLE provides an alternative derivation.

We need to set up a probabilistic model on how the labels are generated from features first. We assume the data is generated by

$$y = f(\boldsymbol{x};\ \boldsymbol{\theta}) + \xi,$$

where y equals a function $f(\boldsymbol{x};\ \boldsymbol{\theta})$ with an unknown parameter $\boldsymbol{\theta}$, added by a Gaussian noise ξ with an variance σ, which can be either given or unknown. This is equivalent to say that y follows a distribution of $\mathcal{N}(f(\boldsymbol{x};\ \theta),\ \ \sigma^2)$, and hence a density of

$$p(y \mid \boldsymbol{x}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y - f(\boldsymbol{x};\ \boldsymbol{\theta}))^2}{2\sigma^2} \right).$$

The log-likelihood function is

$$\begin{aligned} \ell(\boldsymbol{\theta};\ \sigma) &= \sum_{i=1}^{n} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)};\ \boldsymbol{\theta}) \\ &= \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - f(\boldsymbol{x}^{(i)};\ \boldsymbol{\theta}))^2}{2\sigma^2} \right) \right) \\ &= -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y^{(i)} - f(\boldsymbol{x}^{(i)};\ \boldsymbol{\theta}))^2 - n \log \sigma - n \log(\sqrt{2\pi}). \end{aligned}$$

Optimizing $\boldsymbol{\theta}$ yields

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (y^{(i)} - f(\boldsymbol{x}^{(i)};\ \boldsymbol{\theta}))^2$$

This is equivalent to minimizing the mean square error. Optimizing σ yields

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - f(\boldsymbol{x}^{(i)};\, \hat{\boldsymbol{\theta}}))^2,$$

which equals the mean square error obtained by the prediction with $f(\boldsymbol{x};\, \hat{\boldsymbol{\theta}})$.

MLE for Classification (Logistic Regression)
Given $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^n$ where $y^{(i)} \in \{0, 1\}$ for binary classification. Assume

$$P\left(y = 1 \mid \boldsymbol{x},\, \boldsymbol{\theta}\right) = \frac{\exp(f(\boldsymbol{x};\, \boldsymbol{\theta}))}{1 + \exp(f(\boldsymbol{x};\, \boldsymbol{\theta}))}, \qquad P\left(y = 0 \mid \boldsymbol{x},\, \boldsymbol{\theta}\right) = \frac{1}{1 + \exp(f(\boldsymbol{x};\, \boldsymbol{\theta}))}.$$

It is convenient to write the probability in the following unified form,

$$P\left(y \mid \boldsymbol{x},\, \boldsymbol{\theta}\right) = \frac{\exp(yf(\boldsymbol{x};\, \boldsymbol{\theta}))}{1 + \exp(f(\boldsymbol{x};\, \boldsymbol{\theta}))},$$

which can be verified by taking $y = 0$ and 1.

The log-likelihood function is

$$\begin{aligned} L(\boldsymbol{\theta}) &= \sum_{i=1}^{n} \log P(y^{(i)} \mid x^{(i)}) \\ &= \sum_{i=1}^{n} \log \frac{\exp(y^{(i)} f(\boldsymbol{x}^{(i)};\, \boldsymbol{\theta}))}{1 + \exp(f(\boldsymbol{x}^{(i)};\, \boldsymbol{\theta}))} \\ &= \sum_{i=1}^{n} \left(y^{(i)} f(\boldsymbol{x}^{(i)};\, \boldsymbol{\theta}) - \log(1 + \exp(f(\boldsymbol{x}^{(i)};\, \boldsymbol{\theta}))) \right). \end{aligned}$$

The maximum likelihood estimator $\hat{\theta}$ is

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left\{ L(\boldsymbol{\theta}) := \sum_{i=1}^{n} \left(y^{(i)} f(\boldsymbol{x}^{(i)};\, \boldsymbol{\theta}) - \log(1 + \exp(f(\boldsymbol{x}^{(i)};\, \boldsymbol{\theta}))) \right). \right\}.$$

It can be solved by numerical optimization methods (such as gradient descent).

4.2 Multi-class Logistic Regression (Optional)

Logistic regression can be easily extended to multi-class classification. Consider predicting a multi-class label $y \in \{1, \ldots, m\}$ from a feature vector $\boldsymbol{x} = [x_1, \ldots, x_d]^\top \in \mathbb{R}^{d \times 1}$. Logistic regression assumes the label y is randomly generated from $\boldsymbol{x}$ by the following conditional distribution:

$$\Pr(y = k \mid \boldsymbol{x},\, \boldsymbol{\Theta}) = \frac{\exp(\boldsymbol{\theta}_k^\top \boldsymbol{x})}{\sum_{\ell=1}^{m} \exp(\boldsymbol{\theta}_\ell^\top \boldsymbol{x})}, \tag{4.3}$$

where $\boldsymbol{\theta}_k := [\theta_{k,1}, \ldots, \theta_{k,d}] \in \mathbb{R}^{d \times 1}$ is a d-dimensional parameter vector for class k, and $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_m] \in \mathbb{R}^{d \times m}$ is a matrix that includes all the $\boldsymbol{\theta}_k$ vectors. Each $\boldsymbol{\theta}_k$ can be treated as the importance weight of the features in terms of predicting label $y = k$.

The log of the probability is

$$\log \Pr(y = k \mid \boldsymbol{x},\, \boldsymbol{\Theta}) = \boldsymbol{\theta}_k^\top \boldsymbol{x} - \log \sum_{\ell=1}^{m} \exp(\boldsymbol{\theta}_\ell^\top \boldsymbol{x}).$$

We can calculate the log-ratio of the probabilities (called the log-odds) of any two classes (e.g., $y = k$ vs. $y = \ell$):

$$\log\left(\frac{\Pr(y = k \mid \boldsymbol{x},\ \boldsymbol{\Theta})}{\Pr(y = \ell \mid \boldsymbol{x},\ \boldsymbol{\Theta})}\right) = (\boldsymbol{\theta}_k - \boldsymbol{\theta}_\ell)^\top \boldsymbol{x}.$$

This suggests that the log-odds between any two classes is a linear function of feature $\boldsymbol{x}$. Therefore, logistic regression is also called *log-linear models*.

Given n independent observed data points $\{\boldsymbol{x}^{(i)},\ y^{(i)}\}_{i=1}^n$, we estimate Θ by the maximum likelihood estimator. Define γ_{ik} to be the indicator of if $y^{(i)} = k$:

$$\gamma_{ik} := \mathbb{I}[y^{(i)} = k] = \begin{cases} 1 & \text{if } y^{(i)} = k \\ 0 & \text{if } y^{(i)} \neq k, \end{cases}$$

The joint log-likelihood of observing $\{\boldsymbol{x}^{(i)},\ y^{(i)}\}_{i=1}^n$ is

$$\begin{aligned}
\ell(\boldsymbol{\Theta}) &= \sum_{i=1}^n \log \Pr(y^{(i)} \mid \boldsymbol{x}^{(i)},\ \boldsymbol{\Theta}) \\
&= \sum_{i=1}^n \sum_{k=1}^m \gamma_{ik} \log \Pr(y^{(i)} = k \mid \boldsymbol{x}^{(i)},\ \boldsymbol{\Theta}) \qquad \text{//Introducing the indicator function} \\
&= \sum_{i=1}^n \sum_{k=1}^m \gamma_{ik} \cdot \left(\boldsymbol{\theta}_k^\top \boldsymbol{x}^{(i)} - \log \sum_{\ell=1}^m \exp(\boldsymbol{\theta}_\ell^\top \boldsymbol{x}^{(i)})\right) \\
&= \sum_{i=1}^n \sum_{k=1}^m \gamma_{ik} \boldsymbol{\theta}_k^\top \boldsymbol{x}^{(i)} - \sum_{i=1}^n \log \sum_{\ell=1}^m \exp(\boldsymbol{\theta}_\ell^\top \boldsymbol{x}^{(i)}). \qquad \text{//Used } \sum_{k=1}^m \gamma_{ik} = 1 \text{ for all } i
\end{aligned}$$

It turns out that $\ell(\boldsymbol{\Theta})$ is a concave function of $\boldsymbol{\Theta}$ and can be maximized using numerical methods such as gradient descent.

Gradient and Log-sum-exp Function (Optional)

Let us derive the gradient of $\ell(\boldsymbol{\Theta})$. It is useful to introduce the log-sum-exp function:

$$\Phi(\boldsymbol{t}) = \log \sum_{\ell=1}^m \exp(t_\ell),$$

which takes a vector $\boldsymbol{t} = [t_1, \ldots, t_m]^\top \in \mathbb{R}^{m \times 1}$ as input and outputs a number. Therefore, the log-likelihood can be rewritten into

$$\ell(\boldsymbol{\Theta}) = \sum_{i=1}^n \sum_{k=1}^m \gamma_{ik} \boldsymbol{\theta}_k^\top \boldsymbol{x}^{(i)} - \sum_{i=1}^n \Phi(\boldsymbol{\Theta}^\top \boldsymbol{x}^{(i)}),$$

where $\Theta^\top \boldsymbol{x}^{(i)} = [\boldsymbol{\theta}_1^\top \boldsymbol{x}^{(i)}, \ldots \boldsymbol{\theta}_m^\top \boldsymbol{x}^{(i)}]^\top$ is the collection of $\boldsymbol{\theta}_\ell^\top \boldsymbol{x}^{(i)}$ for all classes $\ell = 1, \ldots, m$.

It is not difficult to show that the derivative of $\Phi(\boldsymbol{t})$ w.r.t. each t_k is

$$\frac{\partial \Phi(\boldsymbol{t})}{\partial t_k} = \frac{\exp(t_k)}{\sum_{\ell=1}^m \exp(t_\ell)}.$$

If we denote by $p_k = \dfrac{\partial \Phi(\boldsymbol{t})}{\partial t_k}$, then $\{p_k\}$ forms a valid probability on $[1, \ldots, m]$. In a vector form, we may simply write

$$\nabla_{\boldsymbol{t}} \Phi(\boldsymbol{t}) = \frac{1}{\sum_{\ell=1}^m \exp(t_\ell)} \exp(\boldsymbol{t}), \tag{4.4}$$

where both sides are vectors of size $m \times 1$. In addition, one can show that $\Phi(\boldsymbol{t})$ is a convex function.

The derivative of $\ell(\boldsymbol{\Theta})$ w.r.t. each $\boldsymbol{\theta}_k$ can be shown to be

$$\nabla_{\boldsymbol{\theta}_k} \ell(\boldsymbol{\Theta}) = \sum_{i=1}^{n} \underbrace{\left(\gamma_{ik} - \Pr(y = k \mid \boldsymbol{x}^{(i)},\ \boldsymbol{\Theta})\right)}_{\delta_{ik}} \boldsymbol{x}^{(i)}, \tag{4.5}$$

where $\Pr(y = k \mid \boldsymbol{x}^{(i)},\ \boldsymbol{\Theta})$ is the same as that in (4.3), which defines the probability of $y^{(i)} = k$ given $\boldsymbol{x}^{(i)}$, predicted by the *current model* with parameter Θ, while $\gamma_{ik} := \mathbb{I}[y^{(i)} = k]$ can be viewed as the empirical estimation of the actual probability. Therefore, δ_{ik} is the difference between the empirical and prediction probabilities for label $y = k$ on the i-th data point.

From (4.5), we can see that the gradient of the likelihood is a weighted combination of the feature vectors $\boldsymbol{x}^{(i)}$, each weighted by the empirical-model residual δ_{ik}. The data points that are largely inconsistent with the model is going to have a large magnitude of δ_{ik} (either possitive or negative), and their $\boldsymbol{x}^{(i)}$ contributes a lot to the gradient, in order to update the model to make them correct. On the other hand, the data points that are predicted correctly with high confidence (for which the magnitude of δ_{ik} is small) is going to get small weights, and does not contribute much to the update of the model parameters, because they are already predicted correctly.

Note that each $\nabla_{\boldsymbol{\theta}_k} \ell(\boldsymbol{\Theta})$ is a $d \times 1$ vector of the same size as $\boldsymbol{\theta}_k$. Putting all of them together, $\nabla_{\boldsymbol{\Theta}} \ell(\boldsymbol{\Theta})$ is going to be a $d \times m$ matrix of the same size as $\boldsymbol{\Theta}$:

$$\nabla_{\boldsymbol{\Theta}} \ell(\boldsymbol{\Theta}) = \underbrace{\left[\nabla_{\boldsymbol{\theta}_1} \ell(\boldsymbol{\Theta}),\ \ldots,\ \nabla_{\boldsymbol{\theta}_m} \ell(\boldsymbol{\Theta})\right]}_{d \times m}.$$

To get a compact matrix representation for $\nabla_{\boldsymbol{\Theta}} \ell(\boldsymbol{\Theta})$, we define

$$\boldsymbol{\delta}^{(i)} = \underbrace{\begin{bmatrix} \mathbb{I}[y^{(i)} = 1] - \Pr(y = 1 \mid \boldsymbol{x}^{(i)},\ \boldsymbol{\Theta}) \\ \vdots \\ \mathbb{I}[y^{(i)} = m] - \Pr(y = m \mid \boldsymbol{x}^{(i)},\ \boldsymbol{\Theta}) \end{bmatrix}}_{m \times 1}$$

which is a $m \times 1$ vector that includes the differences between the empirical and model-assumed probability for each class $y = k$, on the i-th data point $(\boldsymbol{x}^i, y^{(i)})$. Then we have

$$\underbrace{\nabla_{\boldsymbol{\Theta}} \ell(\boldsymbol{\Theta})}_{d \times m} = \sum_{i=1}^{n} \underbrace{\boldsymbol{x}^i}_{d \times 1} \underbrace{(\boldsymbol{\delta}^i)^\top}_{1 \times m}$$

where we can check that both sides are $d \times m$ matrices.

Further, let $\boldsymbol{\Delta} = [\boldsymbol{\delta}^{(1)}, \ldots, \boldsymbol{\delta}^{(n)}]$ be a $m \times n$ matrix, and $\boldsymbol{X} = [\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{(n)}]$ be the $d \times n$ matrix that holds all the feature vectors. We have

$$\underbrace{\nabla_{\boldsymbol{\Theta}} \ell(\boldsymbol{\Theta})}_{d \times m} = \underbrace{\boldsymbol{X}}_{d \times n} \underbrace{\boldsymbol{\Delta}^\top}_{n \times m}.$$

This is the form that we should use to implement the algorithm, to avoid for loops.

Derivation of the Gradient in (4.5)

Recall

$$\ell(\boldsymbol{\Theta}) = \underbrace{\sum_{k=1}^{m} \big(\sum_{i=1}^{n} \gamma_{ik} \boldsymbol{x}^{(i)}\big)^\top \boldsymbol{\theta}_k}_{I} \quad - \quad \underbrace{\sum_{i=1}^{n} \Phi(\boldsymbol{\Theta}^\top \boldsymbol{x}^{(i)})}_{II}.$$

Because I is a linear function of $\boldsymbol{\theta}_k$, taking its derivative gives

$$\frac{\partial I}{\partial \boldsymbol{\theta}_k} = \sum_{i=1}^{n} \gamma_{ik} \boldsymbol{x}^{(i)},$$

which are the coefficients of $\boldsymbol{\theta}_k$ in I.

For the second term II, we need to use chain rule. Let $t_\ell = \boldsymbol{\theta}_\ell^\top \boldsymbol{x}$, then

$$\frac{\partial t_\ell}{\partial \boldsymbol{\theta}_k} = \frac{\partial (\boldsymbol{\theta}_\ell^\top \boldsymbol{x})}{\partial \theta_k} = \begin{cases} \boldsymbol{x} & \text{if } k = \ell \\ 0 & \text{if } k \neq \ell, \end{cases} \tag{4.6}$$

where $\frac{\partial t_\ell}{\partial \boldsymbol{\theta}_k}$ equals zero when $\ell \neq k$ because t_ℓ does not depend on $\boldsymbol{\theta}_k$.

$$\begin{aligned}
\frac{\partial \Phi(\boldsymbol{\Theta}^\top \boldsymbol{x})}{\partial \boldsymbol{\theta}_k} &= \frac{\partial \Phi([t_1 \ldots, t_m])}{\partial \boldsymbol{\theta}_k} \\
&= \sum_{\ell=1}^{m} \frac{\partial \Phi([t_1 \ldots, t_m])}{\partial t_\ell} \frac{\partial t_\ell}{\partial \boldsymbol{\theta}_k} && \text{//by multivariate chain rule} \\
&= \frac{\partial \Phi([t_1 \ldots, t_m])}{\partial t_k} \frac{\partial t_k}{\partial \boldsymbol{\theta}_k} && \text{//because } \frac{\partial t_\ell}{\partial \boldsymbol{\theta}_k} = 0 \text{ for } k \neq \ell \\
&= \Pr(y = k \mid \boldsymbol{x},\ \boldsymbol{\Theta}) \cdot \boldsymbol{x} && \text{//Using (4.4) and (4.6)}
\end{aligned}$$

Therefore

$$\frac{\partial II}{\partial \theta_k} = \sum_{i=1}^{n} \frac{\partial \Phi(\boldsymbol{\Theta}^\top \boldsymbol{x}^{(i)})}{\partial \boldsymbol{\theta}_k} = \sum_{i=1}^{n} \Pr(y = k \mid \boldsymbol{x}^{(i)},\ \boldsymbol{\Theta}) \cdot \boldsymbol{x}^{(i)}$$

Combining all the derivation together:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_k} \ell(\boldsymbol{\Theta}) &= \frac{\partial I}{\partial \theta_k} - \frac{\partial II}{\partial \theta_k} \\
&= \sum_{i=1}^{n} \gamma_{ik} \boldsymbol{x}^{(i)} - \sum_{i=1}^{n} \Pr(y = k \mid \boldsymbol{x}^{(i)},\ \boldsymbol{\Theta}) \cdot \boldsymbol{x}^{(i)} \\
&= \sum_{i=1}^{n} \big(\gamma_{ik} - \Pr(y = k \mid \boldsymbol{x}^{(i)},\ \boldsymbol{\Theta})\big) \boldsymbol{x}^{(i)}.
\end{aligned}$$

This completes the proof.

4.3 Bayesian Inference

Recall that the Bayesian formula is

$$p(\theta \mid D) = \frac{p(D \mid \theta)p_0(\theta)}{p(D)}, \qquad p(D) = \int p(D|\theta)p_0(\theta)d\theta.$$

where $p_0(\theta)$ is the prior distribution of θ, $p(D|\theta)$ is the probability of observation D giving θ, and $p(D)$ is the marginal distribution of seeing data (when θ is marginalized). Because $p(D)$ just serves a normalization constant and does not depend on θ, we often write

$$p(\theta \mid D) \propto p(D \mid \theta)p_0(\theta),$$

where $\propto$ denotes "proportional to". When the data consists of identical and independent distributed, $D = \{x^{(i)}\}_{i=1}^n$, we have

$$P(D \mid \theta) = p(\{x_1, \ldots, x_n\} \mid \theta) = \prod_{i=1}^n p(x^{(i)} \mid \theta),$$

and hence

$$p(\theta \mid D) \propto \left[\prod_{i=1}^n p(x^{(i)} \mid \theta)\right] p_0(\theta).$$

■ **Example 4.6 — Beta Prior for Bernoulli.** Assume $\mathcal{D} = \{x_1, \cdots, x_n\}$ are drawn i.i.d. from a $Bernoulli(\theta)$ distribution:

$$p(X = 1) = \theta, \qquad P(X = 0) = 1 - \theta.$$

Viewing θ as a random variable whose prior is a $Beta(\alpha, \beta)$ distribution, whose density is defined to be

$$p_0(\theta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)}, \qquad B(\alpha, \beta) = \int_0^1 \theta^{\alpha-1}(1-\theta)^{\beta-1}d\theta,$$

where α and β are "hyper-parameters" that users need to decide, and $B(\alpha, \beta)$ serves as the normalization constant and is known as the *Beta function*. It is one of the *special functions* in mathematics, whose value can be calculated numerically. One can derive the expectation of $Beta(\alpha, \beta)$:

$$\mathbb{E}_{p_0}[\theta] = \int \theta p_0(\theta)d\theta = \int \theta \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)}d\theta = \frac{B(\alpha+1, \beta)}{B(\alpha, \beta)} = \frac{\alpha}{\alpha+\beta},$$

where the last step comes from the special property of Beta function, that $B(\alpha + 1, \beta) = B(\alpha, \beta) \times \frac{\alpha}{\alpha+\beta}$.

Denote by m_1 and m_0 the number of 1 and 0 in the dataset $\mathcal{D}$. The posterior distribution of θ is

$$p(\theta \mid \mathcal{D}) \propto \left[\prod_{i=1}^n P(x_i \mid \theta)\right] p_0(\theta) \propto [\theta^{m_1}(1-\theta)^{m_0}]\, \theta^{\alpha-1}(1-\theta)^{\beta-1}$$
$$\propto \theta^{m_1+\alpha-1}(1-\theta)^{m_0+\beta-1}.$$

Note that $P(x_i \mid \theta)$ is a probability, but $p_0(\theta)$ is a density function. It is ok to mix probability and density function in this way.

Matching the definition of Beta distribution above, we can see that the posterior distribution $p(\theta \mid \mathcal{D})$ is $Beta(m_0 + \alpha,\ \ m_1 + \beta)$, and hence has a posterior mean of

$$\mathbb{E}[\theta \mid \mathcal{D}] = \frac{m_1 + \alpha}{m_1 + \alpha + m_2 + \beta}.$$

■ **Example 4.7 — Posterior of Gaussian.** Assume $\mathcal{D} = \{x_1, \ldots, x_n\}$ are i.i.d. drawn from a Gaussian distribution $\mathcal{N}(\theta,\ \sigma^2)$, where σ is known and given, but θ is unknown. Assume the prior of θ is $\mathcal{N}(\mu_0, \sigma_0^2)$, where μ_0 and σ_0 are given and estimated from some prior knowledge. We are interested in finding the posterior distribution of θ given the observation $\mathcal{D}$.

Writing down the density functions, we have

$$p(x_i \mid \theta) \propto \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x_i - \theta)^2}{2\sigma^2}\right), \qquad p_0(\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\theta - \mu_0)^2}{2\sigma_0^2}\right).$$

The posterior distribution is

$$\begin{aligned}
p(\theta \mid \mathcal{D}) &\propto \left[\prod_{i=1}^{n} p(x_i \mid \theta)\right] p_0(\theta) \\
&\propto \left[\prod_{i=1}^{n} \exp\left(-\frac{(x_i - \theta)^2}{2\sigma^2}\right)\right] \exp\left(-\frac{(\theta - \mu_0)^2}{2\sigma_0^2}\right) \\
&\propto \exp\left(-\left(\frac{n}{2\sigma^2} + \frac{1}{2\sigma_0^2}\right)\theta^2 + \left(\frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma^2}\sum_{i=1}^{n} x_i\right)\theta\right) \quad \text{//dropped terms independent of } \theta \\
&\propto \exp\left(-\frac{1}{2}A\theta^2 + B\theta\right) \qquad \text{//define } A = \frac{n}{\sigma^2} + \frac{1}{\sigma_0^2}, \text{ and } B = \frac{\mu_0}{\sigma_0^2} + \frac{1}{\sigma^2}\sum_{i=1}^{n} x_i \\
&\propto \exp\left(-\frac{1}{2}A\left(\theta - \frac{B}{A}\right)^2\right), \qquad (4.7)
\end{aligned}$$

where we used $A(\theta - B/A)^2 = A\theta^2 - 2B\theta + B^2/A$ and dropped the constant term B^2/A.

Comparing (4.7) with the definition of Gaussian distribution, we can see that the posterior distribution is Gaussian with mean B/A and variance $1/A$, that is, we have $p(\theta \mid \mathcal{D}) = \mathcal{N}(\theta;\ \mu_{poster},\ \sigma^2_{post})$, with

$$\mu_{poster} = \frac{B}{A} = \frac{\frac{1}{\sigma_0^2}\mu_0 + \frac{1}{\sigma^2}\sum_{i=1}^{n} x_i}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}, \qquad \sigma^2_{poster} = \frac{1}{A} = \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}}.$$

The inverse of the variance $1/\sigma^2$ is also called the **precision**. We can see that the posterior mean μ_{poster} is a weighted average of the prior μ_0 (weighted by the prior precision $1/\sigma_0^2$), and the observations (each weighed by the observation precision $1/\sigma^2$). The posterior precision $1/\sigma^2_{poster}$ equals the sum of the prior precision and all the precisions from the observation.

5. Clustering, K-means, Mixture, EM

5.1 Clustering and K-means

Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a *cluster*) are more similar to each other than to those in different groups (clusters). It is an instance of unsupervised learning tasks, which find unknown patterns in dataset without pre-existing labels. The goal of clustering is often to provide exploratory analysis, understanding, and visualization of the data at hand, in contrast with label prediction in supervised learning.

Specifically, in the clustering task, we are giving a dataset $\{\boldsymbol{x}^{(i)}\}_{i=1}^n$ and a positive integer K, where each $\boldsymbol{x}^{(i)}$ represents a feature vector of an instance or objective of interest. The goal is to partition the dataset into K groups, so that data points in the same group are more "similar" to these assigned into different groups. There are several equivalent ways to present clusterings:

1. **Set Representation** A clustering partitions the data into K non-overlapping subsets, $S_1, \ldots, S_K$, such that $S_1 \cup \cdots \cup S_K = \{1, \ldots, n\}$ and $S_\ell \cap S_k = \emptyset$ for every $\ell \neq k$. The subset $\{\boldsymbol{x}_i \colon i \in S_\ell\}$ of data points whose indexes are in S_ℓ forms the ℓ-th cluster.
2. **Index Representation** We want to assign each data point $\boldsymbol{x}^{(i)}$ a cluster ID $z^{(i)} \in \{1, 2, \cdots, K\}$. In this way, S_k forms the subset of data points with cluster ID $z^{(i)} = k$, that is, $S_k = \{i \colon z^{(i)} = k\}$.
3. **One-hot Vectors & Probabilities** Equivalently, we may represent the cluster ID $z^{(i)}$ with an *one-hot* vector $\gamma^{(i)} = [\gamma_{i1}, \ldots, \gamma_{iK}] \in \mathbb{R}^K$, such that

$$\gamma_{ik} = \begin{cases} 1 & z^{(i)} = k \\ 0 & z^{(i)} \neq k. \end{cases}$$

One may view $\gamma^{(i)}$ as the probability of assigning $\boldsymbol{x}^{(i)}$ into different clusters.

Note that the cluster IDs are permutation invariant, meaning that the result does not change if we swap the cluster IDs of any two clusters.

K-means is one of the most basic algorithms for clustering. It jointly optimizes the cluster IDs of the data points, and a *centroid*, denoted by $\mu^{(k)}$, for each cluster S_k. The objective function of

K-means is

$$\min_{\boldsymbol{z}} \min_{\boldsymbol{\mu}} \left\{ F(\boldsymbol{z},\ \boldsymbol{\mu}) := \sum_{i=1}^{n} \left\| \mathbf{x}^{(i)} - \boldsymbol{\mu}^{(z^{(i)})} \right\|^2 \right\}, \tag{5.1}$$

where $\boldsymbol{z} := \{z^{(i)}\}_{i=1}^n$ is the set of cluster IDs of the data points and $\boldsymbol{\mu} := \{\boldsymbol{\mu}^{(k)}\}_{k=1}^K$ is the set of cluster centroids. The idea is that we compare each data point $\boldsymbol{x}^{(i)}$ with the centroid of the cluster it is assigned to (which is $\boldsymbol{\mu}^{(z^{(i)})}$), and minimize the sum of squared distances of the data points to their assigned centroids.

Coordinate Descent K-means can be viewed as a coordinate descent algorithm applied on $F(\boldsymbol{z}, \boldsymbol{\mu})$. It starts from some initialization $(\boldsymbol{z}_0, \boldsymbol{\mu}_0)$, and alternatively update $\boldsymbol{z}$ and $\boldsymbol{\mu}$ at each iteration t until the algorithm converges:

Fixing $\boldsymbol{\mu}$, update $\boldsymbol{z}$:

$$\boldsymbol{z}_{t+1} = \arg\min_{\boldsymbol{z}} F(\boldsymbol{z}, \boldsymbol{\mu}_t), \tag{5.2}$$

Fixing $\boldsymbol{z}$, update $\boldsymbol{\mu}$:

$$\boldsymbol{\mu}_{t+1} = \arg\min_{\boldsymbol{\mu}} F(\boldsymbol{z}_{t+1}, \boldsymbol{\mu}) \tag{5.3}$$

Repeating this process allows us to decrease objective function monotonically.

Theorem 5.1 Following the updates in (5.3) and (5.2), we have

$$F(\boldsymbol{z}_t,\ \boldsymbol{\mu}_t) \geq F(\boldsymbol{z}_{t+1},\ \boldsymbol{\mu}_{t+1}), \qquad \forall t.$$

Proof.

$$\begin{aligned} F(\boldsymbol{z}_t, \boldsymbol{\mu}_t) &\geq F(\boldsymbol{z}_{t+1}, \boldsymbol{\mu}_t) \qquad \text{//following (5.2)} \\ &\geq F(\boldsymbol{z}_{t+1}, \boldsymbol{\mu}_{t+1}) \qquad \text{//following (5.3).} \end{aligned}$$

■

Therefore, repeating this process allows us to find a local optima of $F(\boldsymbol{z}, \boldsymbol{\mu})$. But it is important to note that it does not grantee to find a global optimum, because $F(\boldsymbol{z}, \boldsymbol{\mu})$ may have many local optima. Running coordinate descent with different initialization $(\boldsymbol{z}_0, \boldsymbol{\mu}_0)$ may converge to different local optima of $F(\boldsymbol{z}, \boldsymbol{\mu})$.

In the case of K-means, optimizing $\boldsymbol{z} = \{z^{(i)}\}_{i=1}^n$ with fixed $\boldsymbol{\mu}$ gives

$$z^{(i)} \leftarrow \arg\min_{k} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|^2, \qquad \forall i = 1, \ldots, n. \tag{5.4}$$

This corresponds to assigning each data point $\boldsymbol{x}^{(i)}$ to the cluster with the closest centroid.

To see why (5.4) corresponds to (5.2), we just need to note that the K-means objective function in (5.1) depends on $z^{(i)}$ only through $\left\| \boldsymbol{x}^{(i)} - \mu^{(z^{(i)})} \right\|$ which is the contribution of the loss from data point $\boldsymbol{x}^{(i)}$. Therefore, we just need to solve $\min_{z^{(i)}} \left\| \boldsymbol{x}^{(i)} - \mu^{(z^{(i)})} \right\|$ in order to update $z^{(i)}$.

Optimizing $\boldsymbol{\mu} = \{\boldsymbol{\mu}^{(k)}\}_{i=1}^K$ with fixed $\boldsymbol{z}$ yields

$$\boldsymbol{\mu}^{(k)} \leftarrow \frac{1}{|S_k|} \sum_{i \in S_k} \boldsymbol{x}^{(i)}, \qquad \text{where} \quad S_k := \left\{ i:\ z^{(i)} = k \right\}, \quad \forall k = 1, \ldots, K. \tag{5.5}$$

This corresponds to calculating the centroid of the k-th cluster by averaging the subset S_k of data points that belong to cluster k according to the cluster IDs $\{z^{(i)}\}$.

To see why (5.5) solves (5.3), note that the K-means objective in (5.1) can be rewritten into

$$F(\boldsymbol{z}, \boldsymbol{\mu}) = \sum_{i=1}^{n} \left\| \mathbf{x}^{(i)} - \boldsymbol{\mu}^{(z^{(i)})} \right\|^2 = \sum_{k=1}^{K} \sum_{i \in S_k} \left\| \mathbf{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|^2,$$

where we decompose the sum $\sum_{i=1}^{n}$ over the data points into the sum overall the clusters ($\sum_{k=1}^{K}$), and the data points belong to each cluster ($\sum_{i \in S_k}$). Since the objective is a sum over the clusters, the optimal $\mu^{(k)}$ for each cluster k can be found by

$$\min_{\boldsymbol{\mu}^{(k)}} \sum_{i \in S_k} \left\| \mathbf{x}^{(i)} - \boldsymbol{\mu}^{(k)} \right\|^2,$$

which can be shown to reduce to averaging the data points in S_k as (5.5).

5.2 Gaussian Mixture and EM

Gaussian mixture model (GMM) is a natural probabilistic model for clustering, in which each cluster is represented by a Gaussian distribution. The density function of a GMM is a weighted linear combination of several Gaussian density functions,

$$p(x \mid \theta) = \sum_{k=1}^{K} w_k \times \mathcal{N}(x;\ \mu_k,\ \sigma_k^2), \tag{5.6}$$

where the parameter is $\theta = \{w_k, \mu_k, \sigma_k\}_{k=1}^{K}$, and $\{w_k\}_{k=1}^{K}$ is a set of mixture weights that satisfy $\sum_k w_k = 1$ and $w_k \geq 0$. Here $\mathcal{N}(x;\ \mu_k,\ \sigma_k^2)$ is the density of Gaussian distribution:

$$\mathcal{N}(x;\ \mu_k,\ \sigma_k^2) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2 \right).$$

Each $\mathcal{N}(x;\ \mu_k,\ \sigma_k^2)$ is called a component of $p(x \mid \theta)$, which corresponds to a cluster in the clustering problem.

If a random variable X is drawn from a Gaussian mixture model, it can be equivalently drawn from a randomly picked Gaussian component (or cluster), each with probability w_k. To make this explicit, we can introduce an (unobserved or latent) index variable $Z \in \{1, 2, \ldots, K\}$ to represent which Gaussian component X is drawn from. We can generate (X, Z) using the following procedure:

Draw latent label Z : $P(Z = k \mid \theta) = w_k$

//draw $Z = k$ with probability w_k

Draw observation X : $p(X = x \mid Z = k, \theta) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2)$

//if $Z = k$, X is a Gaussian with mean μ_k, variance σ_k^2.

The joint probability of (X, Z) is obtained by the chain rule:

$$\begin{aligned} p(X = x, Z = k \mid \theta) &= P(Z = k \mid \theta) \times p(X = x \mid Z = k,\ \theta) \\ &= w_k \times \mathcal{N}(x;\ \mu_k,\ \sigma_k^2). \end{aligned}$$

Based on this we can calculate the marginal distribution of X by summing over all the possible values of Z using the law of total probability:

$$p(X = x \mid \theta) = \sum_{k=1}^{K} p(X = x, Z = k \mid \theta) = \sum_{k=1}^{K} w_k \times \mathcal{N}(x;\ \mu_k,\ \sigma_k^2).$$

which is equivalent to the density of GMM we defined in (5.6).

In addition, giving an observation of X, we can infer its cluster ID based on the posterior distribution:

$$P(Z = k \mid X = x,\ \theta) = \frac{P(X = x, Z = k \mid \theta)}{p(X = x \mid \theta)} = \frac{w_k \times \mathcal{N}(x;\ \mu_k, \sigma_k^2)}{\sum_{\ell=1}^{K} w_\ell \times \mathcal{N}(x;\ \mu_\ell, \sigma_\ell^2)}. \tag{5.7}$$

■ **Example 5.1** Assume we have a simple Gaussian mixture model with density function

$$p(x) = 0.2 \times \mathcal{N}(x;\ -1, 1) + 0.8 \times \mathcal{N}(x;\ 1, 1).$$

It has two components $\mathcal{N}(x;\ -1, 1)$ and $\mathcal{N}(x;\ 1, 1)$, with probability of 0.2 and 0.8, respectively. Introducing the cluster ID Z, we have the joint distribution:

$$P(X = x,\ Z = k) = \begin{cases} 0.2 \times \mathcal{N}(x;\ -1, 1) & \text{if } k = 1 \\ 0.8 \times \mathcal{N}(x;\ 1, 1) & \text{if } k = 2. \end{cases}$$

The posterior distribution of the cluster ID is

$$P(Z = k \mid X = x) = \begin{cases} \frac{0.2 \times \mathcal{N}(x;\ -1,1)}{0.2 \times \mathcal{N}(x;\ -1,1) + 0.8 \times \mathcal{N}(x;\ 1,1)} & \text{if } k = 1 \\ \frac{0.8 \times \mathcal{N}(x;\ 1,1)}{0.2 \times \mathcal{N}(x;\ -1,1) + 0.8 \times \mathcal{N}(x;\ 1,1)} & \text{if } k = 2. \end{cases}$$

Plugging in the form of Gaussian density function, we have

$$\begin{aligned} P(Z = 1 \mid X = x) &= \frac{0.2 \times \mathcal{N}(x;\ -1, 1)}{0.2 \times \mathcal{N}(x;\ -1, 1) + 0.8 \times \mathcal{N}(x; 1, 1)} \\ &= \frac{0.2 \frac{1}{\sqrt{2\pi}} \exp(-(x+1)^2/2)}{0.2 \times \frac{1}{\sqrt{2\pi}} \exp(-(x+1)^2/2) + 0.8 \times \frac{1}{\sqrt{2\pi}} \exp(-(x-1)^2/2)} \\ &= \frac{0.2 \exp(-x)}{0.2 \exp(-x) + 0.8 \exp(x)}. \end{aligned}$$

In particular, for $x = -1$, we have

$$P(Z = 1 \mid X = -1) = \frac{0.2 \exp(1)}{0.2 \exp(1) + 0.8 \exp(-1)} \approx 0.65,$$

and hence $P(Z = 2 \mid X = -1) = 1 - P(Z = 1 \mid X = -1) \approx 0.35$. Therefore, $x = -1$ is more likely to be drawn from the first cluster, but also has a decent amount of probability from the second cluster. If we were forced to make a binary prediction of Z, we will go with $Z = 1$, which has larger probability. But it is useful to estimate the probability since it tells the confidence of the prediction.

Clustering as Parameter Estimation of GMM

The clustering problem can be formulated as estimating parameters in GMM. Given a dataset $\{\boldsymbol{x}^{(i)}\}_{i=1}^{n}$, which we assume is drawn from a Gaussian mixture model with parameters $\theta :=$

$\{w_k, \mu_k, \sigma_k\}_{k=1}^K$, where μ_k and σ_k represents the mean and variance of the k-th cluster, and w_k its percentage in the dataset, we want to estimate the parameter θ. If we can estimate θ, we can calculate the posterior probability $P(Z = k \mid X = \boldsymbol{x}^{(i)};\ \theta)$ of the cluster IDs for each data point $\boldsymbol{x}^{(i)}$, hence forming a "soft" or probabilistic clustering of the dataset.

As before, the parameters can be estimated by maximum likelihood estimation:

$$\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log p(\boldsymbol{x}^{(i)} \mid \theta).$$

In the case of Gaussian mixture, we have $p(\boldsymbol{x}^{(i)} \mid \theta) = \sum_{k=1}^K w_k \mathcal{N}(\boldsymbol{x}^{(i)};\ \mu_k, \sigma_k^2)$ and obtain

$$\{\hat{w}_k,\ \hat{\mu}_k,\ \hat{\sigma}_k\} = \arg\max_{\{w_k, \mu_k, \sigma_k\}} \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} w_k \mathcal{N}(\boldsymbol{x}^{(i)};\ \mu_k, \sigma_k^2) \right), \tag{5.8}$$

where the optimization of the weights $\{w_k\}$ should be subject to the constraint that they are valid probabilities, that is, $\sum_{k=1}^K w_k = 1$ and $w_k \geq 0$, and the variance should be positive $\sigma_k \geq 0$.

■ **Example 5.2** Assume we observe a set of one dimensional points $\{x^{(i)}\}_{i=1}^n \subset \mathbb{R}$, which we assume to follow a Gaussian mixture model with two components ($K = 2$),

$$p(x \mid \theta) = w_1 \times \mathcal{N}\left(x;\ \mu_1, \sigma_1^2\right) + w_2 \times \mathcal{N}\left(x; \mu_2, \sigma_2^2\right), \quad \theta = \{w_k, \mu_k, \sigma_k\}_{k=1}^2.$$

Then the likelihood of the parameters $\theta = \{w_k, \mu_k, \sigma_k\}_{k=1}^2$ is

$$\begin{aligned}
\ell(\theta) =& \log p(x^{(1)} \mid \theta) \ + \cdots + \ \log p(x^{(n)} \mid \theta) \\
=& \log \left(w_1 \mathcal{N}\left(x^{(1)};\ \mu_1, \sigma_1^2\right) + w_2 \mathcal{N}\left(x^{(1)}; \mu_2, \sigma_2^2\right) \right) \qquad // = \log p(x^{(1)} \mid \theta) \\
&+ \cdots \\
&+ \log \left(w_1 \mathcal{N}\left(x^{(n)};\ \mu_1, \sigma_1^2\right) + w_2 \mathcal{N}\left(x^{(n)}; \mu_2, \sigma_2^2\right) \right) \qquad // = \log p(x^{(n)} \mid \theta).
\end{aligned}$$

Note that the sum over the data points is outside of the log and the sum over the mixture components (k) is inside the log.

Optimizing the log-likelihood above is a rather complicate optimization problem. It is certainly to solve it with general purpose methods like gradient descent. But there are several hurdles: the constraint on the weights $\{w_i\}$ should be eliminated by either reparameterizing it into an unconstrained form, or addressed using projected gradient descent, and the step sizes should be tuned properly; one might want to use different step sizes for the different types of parameters (mean, variance and weights), since they may have different scales.

It turns out expectation-maximization (EM) provides a much more efficient and convenient method for maximizing the log-likelihood of mixture models. In the sequel, we first introduce the practical, algorithmic idea of EM, and then explain how it provides an optimization method for solving (5.8).

Expectation-Maximization (EM)

Algorithmically, EM can be viewed as a "probabilistic variant" of the K-means algorithm in (5.3)-(5.2). Denote by γ_{ik} to be the posterior probability of the i-th data point is drawn from the k-th component, that is,

$$\gamma_{ik} = P(Z = k \mid X = \boldsymbol{x}^{(i)};\ \theta).$$

EM algorithm starts from an initialization $\theta_0 = \{w_{k;0},\ \mu_{k;0},\ \sigma_{k;0}\}_{k=1}^K$, and performs the following updates until the algorithm converges:
***E-Step**: Fixing θ, update $\{\gamma_{ik}\}$:*

$$\gamma_{ik;t+1} = P(Z = k \mid X = \boldsymbol{x}^{(i)};\ \theta_t) = \frac{w_{k;t} \times \mathcal{N}(\boldsymbol{x}^{(i)};\ \mu_{k;t},\ \sigma_{k;t}^2)}{\sum_{\ell=1}^K w_{\ell;t} \times \mathcal{N}(\boldsymbol{x}^{(i)};\ \mu_{\ell;t},\ \sigma_{\ell;t}^2)},$$

where $\theta_t = \{w_{k;t},\ \mu_{k;t},\ \sigma_{k;t}\}_{k=1}^K$ and $\gamma_{ik;t}$ are the values of θ and γ_{ik} at the t-th iteration, respectively. This calculates the posterior probability of the cluster IDs of each data point following (5.7), assuming θ_t is the true parameter.
***M-Step**: Fixing $\{\gamma_{ik}\}$, update θ*:

$$\begin{aligned}
\text{Updating mean:} \quad & \mu_{k;t+1} = \frac{\sum_{i=1}^n \gamma_{ik;t} \boldsymbol{x}^{(i)}}{\sum_{i=1}^n \gamma_{ik;t}} \\
\text{Updating variance:} \quad & \sigma_{k;t+1}^2 = \frac{\sum_{i=1}^n \gamma_{ik;t} (\boldsymbol{x}^{(i)} - \mu_{k;t+1})^2}{\sum_{i=1}^n \gamma_{ik;t}} \\
\text{Updating weights:} \quad & w_{k;t+1} = \frac{\sum_{i=1}^n \gamma_{ik;t}}{\sum_{i=1}^n \sum_{k=1}^K \gamma_{ik;t}},
\end{aligned}$$

where the mean μ_k and variance σ_k of the k-th cluster equals the empirical mean and variance of the data, when each data point is "*weighted*" by their posterior probability of belong to the k-th cluster, based on the estimation $\gamma_{ik;t}$ from the last iteration. The weight w_k of the k-th cluster is the sum of posterior probabilities assigned to the k-th cluster, normalized by the sum of all posterior probabilities.

Remark

w_k denotes the percentage of data points that belong to the k-th cluster, while γ_{ik} is the posterior probability for the i-th data point belonging to the k-th cluster. Therefore, w_k is the average of γ_{ik} across the dataset. In fact, note that $\sum_{k=1}^K \gamma_{ik;t} = 1$ and hence $\sum_{i=1}^n \sum_{k=1}^K \gamma_{ik;t} = n$. We have by $w_{k;t+1} = \frac{\sum_{i=1}^n \gamma_{ik;t}}{n}$.

EM as Coordinate Descent

Interestingly, the EM procedure above can be viewed as a coordinate descent algorithm that monotonically maximize the function $\ell(\theta)$. The idea is that we can in fact find a function $F(\theta, \gamma)$, where $\{\gamma_{ik}\}$, such that

$$\ell(\theta) = \max_{\gamma \in \Gamma} F(\theta,\ \gamma), \tag{5.9}$$

where Γ is the set of all valid posterior probabilities,

$$\Gamma = \left\{ \gamma := \{\gamma_{ik}\} \colon\ \ \gamma_{ik} \geq 0,\ \forall i, k; \qquad \sum_{k=1}^K \gamma_{ik} = 1,\ \forall i \right\}.$$

If a function F satisfies (5.9), then we have

$$\max_{\theta} \ell(\theta) = \max_{\theta} \max_{\gamma \in \Gamma} F(\theta,\ \gamma),$$

which is obtained by taking $\max_\theta$ on both sides of (5.9). This suggests that optimizing θ on $\ell(\theta)$ is equivalent to optimizing θ and γ jointly on $F(\theta, \gamma)$. Consequently, we can optimize θ and γ

alternatively using coordinate descent similar to the idea used in K-means,

$$\begin{aligned}\gamma_{t+1} &\leftarrow \arg\max_{\gamma\in\Gamma} F(\theta_t,\ \gamma)\\ \theta_{t+1} &\leftarrow \arg\max_{\theta} F(\theta,\ \gamma_{t+1}).\end{aligned} \tag{5.10}$$

It turns out, with the F shown in the sequel, the coordinate descent procedure in (5.12) nicely reduces to the EM algorithm shown above, which provides a justifying EM as an rigours optimization for the likelihood function $\ell(\theta)$.

It is useful to simply the notation a bit. Note that

$$p(x \mid \theta) = \sum_{k=1}^{K} P(Z = k,\ X = x \mid \theta)$$

where $P(Z = k,\ X = x \mid \theta)$ denotes the joint probability of data X and the latent variable Z. In the case of Gaussian mixture, we have

$$P(Z = k,\ X = x \mid \theta) = w_k \mathcal{N}(x \mid \mu_k,\ \sigma_k^2).$$

Therefore, the likelihood function can be written into

$$\ell(\theta) = \sum_{i=1}^{n} \log p(x^{(i)} \mid \theta) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} P(X = x^{(i)},\ \ Z = k \mid \theta) \right).$$

Theorem 5.2 Define

$$F(\theta,\ \gamma) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} \log \left(\frac{P(X = x^{(i)},\ Z = k \mid \theta)}{\gamma_{ik}} \right). \tag{5.11}$$

Then we have

$$\ell(\theta) = \max_{\gamma\in\Gamma} F(\theta, \gamma), \quad \forall\theta.$$

For each θ, the maximum is achieved by

$$\gamma_{ik}^{*} = P(Z = k \mid X = x^{(i)};\ \theta), \quad \forall i, k.$$

Proof. The proof relies on Jensen's inequality.

$$\begin{aligned}\ell(\theta) &= \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} P(X = x^{(i)},\ \ Z = k \mid \theta) \right)\\ &= \sum_{i=1}^{n} \log \left(\sum_{k=1}^{K} \gamma_{ik} \frac{P(X = x^{(i)},\ \ Z = k \mid \theta)}{\gamma_{ik}} \right)\\ &\geq \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} \log \left(\frac{P(X = x^{(i)},\ \ Z = k \mid \theta)}{\gamma_{ik}} \right) \qquad \text{//Jensen inequality}\\ &= F(\theta, \gamma).\end{aligned}$$

This shows that $\ell(\theta) \geq F(\theta, \gamma)$ for $\forall\theta$ and $\forall\gamma \in \Gamma$.

On the other hand, plugging γ^* into $F(\theta, \gamma)$, we have

$$\begin{aligned}
F(\theta, \gamma^*) &= \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma^*_{ik} \log \left(\frac{P(X = x^{(i)},\ Z = k \mid \theta)}{\gamma^*_{ik}} \right) \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} P(Z = k \mid X = x^{(i)};\ \theta) \log \left(\frac{P(X = x^{(i)},\ Z = k \mid \theta)}{P(Z = k \mid X = x^{(i)};\ \theta)} \right) \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} P(Z = k \mid X = x^{(i)};\ \theta) \log P(X = x^{(i)} \mid \theta) \quad //_{P(X = x^{(i)} \mid \theta) = \frac{P(X = x^{(i)},\ Z = k \mid \theta)}{P(Z = k \mid X = x^{(i)};\ \theta)}} \\
&= \sum_{i=1}^{n} \log P(X = x^{(i)} \mid \theta) \qquad //_{\sum_{k=1}^{K} P(Z = k \mid X = x^{(i)};\ \theta) = 1} \\
&= \ell(\theta).
\end{aligned}$$

Therefore, $\ell(\theta) = F(\theta, \gamma^*)$. Combining this with $\ell(\theta) \geq F(\theta, \gamma)$ for $\forall \theta$ and $\forall \gamma \in \Gamma$, we get $\ell(\theta) = \max_{\gamma \in \Gamma} F(\theta, \gamma)$. ■

For the F in (5.11), we can show that the coordinate descent reduces to

$$\begin{aligned}
\gamma_{ik;t+1} &\leftarrow P(Z = k \mid X = x^{(i)};\ \theta_t) \qquad \text{//E-step} \\
\theta_{t+1} &\leftarrow \arg\max_{\theta} \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik;t+1} \log P(Z = k\ ,\ X = x^{(i)};\ \theta) \qquad \text{//M-step.}
\end{aligned} \tag{5.12}$$

For the Gaussian mixture model, one can check that this reduces to the EM procedure that we describe above.

6. Multivariate Normal Distributions

6.1 Multivarate Distributions

Consider a random vector $\boldsymbol{X} = [X_1, \ldots, X_d]^\top \in \mathbb{R}^d$. Its distribution is characterized by its probability density function (PDF), which is a non-negative function $p(\boldsymbol{x})$ on $\mathbb{R}^d$ that integrates to one, that is, $p(\boldsymbol{x}) \geq 0$ and $\int p(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1$. For any function $h(\boldsymbol{x})$, its expectation under $\boldsymbol{X}$ is defined as

$$\mathbb{E}[h(\boldsymbol{X})] = \int h(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}.$$

In particular, the mean vector of $\boldsymbol{X}$ is defined by

$$\mathbb{E}[\boldsymbol{X}] = \begin{bmatrix} \mathbb{E}[X_1] \\ \vdots \\ \mathbb{E}[X_d] \end{bmatrix} = \begin{bmatrix} \int x_1 p(\boldsymbol{x}) d\boldsymbol{x} \\ \vdots \\ \int x_d p(\boldsymbol{x}) d\boldsymbol{x} \end{bmatrix} \in \mathbb{R}^d.$$

Note that $\mathbb{E}[\boldsymbol{X}]$ is a vector of the same size as the random vector $\boldsymbol{X}$. The covariance matrix of $\boldsymbol{X}$ is a $d \times d$ matrix consisting of the pairwise covariance of the coordinates of $\boldsymbol{X}$, that is,

$$\mathrm{Cov}(\boldsymbol{X}) = [\mathrm{Cov}(X_i, X_j)]_{ij} = \begin{bmatrix} \mathrm{Var}(X_1) & \mathrm{Cov}(X_1, X_2) & \cdots & \mathrm{Cov}(X_1, X_d) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Var}(X_2) & \cdots & \mathrm{Cov}(X_2, X_d) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(X_d, X_1) & \mathrm{Cov}(X_d, X_2) & \cdots & \mathrm{Var}(X_d) \end{bmatrix},$$

where $\mathrm{Cov}(X_i, X_j)$ denotes the covariance between X_i and X_j, that is,

$$\begin{aligned}\mathrm{Cov}(X_i, X_j) &= \mathbb{E}[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])] \\ &= \mathbb{E}[X_i X_j] - \mathbb{E}[X_i]\mathbb{E}[X_j],\end{aligned}$$

and

$$\mathrm{Var}(X_i) = \mathrm{Cov}(X_i, X_i) = \mathbb{E}[(X_i - \mathbb{E}[X_i])^2] = \mathbb{E}[X_i^2] - (\mathbb{E}[X_i])^2.s$$

In a compact matrix form, we can simply write the covariance matrix into

$$\begin{aligned}\mathrm{Cov}(\boldsymbol{X}) &= \mathbb{E}\left[(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])^\top\right] \\ &= \mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^\top\right] - \mathbb{E}[\boldsymbol{X}](\mathbb{E}[\boldsymbol{X}])^\top.\end{aligned}$$

We can verify that the ij-th element of this matrix form is indeed $\mathrm{Cov}(X_i, X_j)$.

It is easy to see that the covariance matrix is always symmetric, that is, $\mathrm{Cov}(\boldsymbol{X}) = \mathrm{Cov}(\boldsymbol{X})^\top$. In addition, one can show that it is always positive semi-definite.

Theorem 6.1 $\mathrm{Cov}(\boldsymbol{X})$ is a positive semi-definition matrix.

Proof. Let $\boldsymbol{\Sigma} = \mathrm{Cov}(\boldsymbol{X})$. We just need to show that $\boldsymbol{v}^\top \boldsymbol{\Sigma} \boldsymbol{v} \geq 0$ for any vector $\boldsymbol{v} \in \mathbb{R}^d$.

$$\begin{aligned}\boldsymbol{v}^\top \boldsymbol{\Sigma} \boldsymbol{v} &= \boldsymbol{v}^\top \mathbb{E}\left[(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])^\top\right] \boldsymbol{v} \\ &= \mathbb{E}\left[\boldsymbol{v}^\top (\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])(\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])^\top \boldsymbol{v}\right] \\ &= \mathbb{E}\left[\boldsymbol{b}^\top \boldsymbol{b}\right] \qquad \text{//define } \boldsymbol{b} = (\boldsymbol{X} - \mathbb{E}[\boldsymbol{X}])^\top \boldsymbol{v} \\ &= \mathbb{E}\left[\|\boldsymbol{b}\|^2\right] \geq 0.\end{aligned}$$

■

6.2 Multivariate Normal Distribution

The multivariate normal (or Gaussian) distribution is one of the most basic multivariate distributions. We can construct multivariate normal distributions by linear transforms of simple independent univariate normal distributions.

Recall that a random variable on X on $\mathbb{R}$ is univariate normal $\mathcal{N}(\mu, \sigma^2)$ if its density function is

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

In particular, $\mathcal{N}(0, 1)$ is called the standard normal distribution.

Definition 6.1 A vector-valued random variable $\boldsymbol{X}$ on $\mathbb{R}^d$ is called multivariate normal, if it can be obtained by applying linear transform on a set of independent standard normal random variables, that is,

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{Z} + \boldsymbol{\mu}. \tag{6.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{b} \in \mathbb{R}^d$ are deterministic parameters, and $\boldsymbol{Z} = [Z_1, \ldots, Z_d]^\top$ is a set of independent standard normal random variables, that is, $Z_i \sim \mathcal{N}(0, 1)$ and $Z_i \perp Z_j$ for $i \neq j$.

Theorem 6.2 For the multivariate normal random variable in (6.1), we have

$$\mathbb{E}[\boldsymbol{X}] = \boldsymbol{\mu}, \qquad \mathrm{Cov}(\boldsymbol{X}) = \boldsymbol{A}\boldsymbol{A}^\top.$$

Define $\boldsymbol{\Sigma} = \boldsymbol{A}\boldsymbol{A}^\top$. The multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is denoted by $\mathcal{N}(\boldsymbol{\mu},\ \boldsymbol{\Sigma})$.

Proof. First, because $Z_i,\ \forall i = 1, \ldots, d$ are $\mathcal{N}(0, 1)$ and independent with each other, we have $\mathbb{E}[Z_i] = 0$, $\mathrm{Var}(Z_i) = \mathrm{Cov}(Z_i, Z_i) = 1$ and $\mathrm{Cov}(Z_i, Z_j) = 0$ for $i \neq j$. Therefore,

$$\mathbb{E}[\boldsymbol{Z}] = \boldsymbol{0}, \qquad \mathrm{Cov}(\boldsymbol{Z}) = [\mathrm{Cov}(Z_i, Z_j)]_{ij} = \boldsymbol{I}_{d\times d}.$$

where $I_{d\times d}$ denotes the identity matrix of size $d \times d$. By the linear property of expectation:

$$\mathbb{E}[\boldsymbol{X}] = \mathbb{E}[\boldsymbol{A}\boldsymbol{Z} + \boldsymbol{\mu}] = \boldsymbol{A}\mathbb{E}[\boldsymbol{Z}] + \boldsymbol{\mu} = \boldsymbol{A}\boldsymbol{0} + \boldsymbol{\mu} = \boldsymbol{\mu}.$$

Similarly,

$$\begin{aligned}
\mathrm{Cov}(\boldsymbol{X}) &= \mathbb{E}[(\boldsymbol{X} - \boldsymbol{\mu})(\boldsymbol{X} - \boldsymbol{\mu})^\top] \\
&= \mathbb{E}[(\boldsymbol{A}\boldsymbol{Z} + \boldsymbol{\mu} - \boldsymbol{\mu})(\boldsymbol{A}\boldsymbol{Z} + \boldsymbol{\mu} - \boldsymbol{\mu})^\top] \\
&= \mathbb{E}[(\boldsymbol{A}\boldsymbol{Z})(\boldsymbol{A}\boldsymbol{Z})^\top] \\
&= \mathbb{E}[\boldsymbol{A}\boldsymbol{Z}\boldsymbol{Z}^\top\boldsymbol{A}^\top] \\
&= \boldsymbol{A}\mathbb{E}[\boldsymbol{Z}\boldsymbol{Z}^\top]\boldsymbol{A}^\top \\
&= \boldsymbol{A}\mathrm{Cov}(\boldsymbol{Z})\boldsymbol{A}^\top \\
&= \boldsymbol{A}\boldsymbol{A}^\top,
\end{aligned}$$

where we used the fact that $\mathrm{Cov}(\boldsymbol{Z}) = \mathbb{E}[\boldsymbol{Z}\boldsymbol{Z}^\top]$, because $\mathbb{E}[\boldsymbol{Z}] = \boldsymbol{0}$. ■

Extending the proof above. We have the following more general result.

Theorem 6.3 If $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{\mu}_0,\ \boldsymbol{\Sigma}_0)$ and $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{Z} + \boldsymbol{b}$, then $\boldsymbol{X}$ is also multivariate normal and

$$\mathbb{E}[\boldsymbol{X}] = \boldsymbol{A}\boldsymbol{\mu}_0 + \boldsymbol{b}, \qquad \mathrm{Cov}(\boldsymbol{X}) = \boldsymbol{A}\boldsymbol{\Sigma}_0\boldsymbol{A}^\top.$$

Therefore, $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu}_0 + \boldsymbol{b},\ \boldsymbol{A}\boldsymbol{\Sigma}_0\boldsymbol{A}^\top)$.

It can be shown that if $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, its probability density function (PDF) is

$$p(\boldsymbol{x}) = \frac{1}{D}\exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right), \qquad D = \sqrt{(2\pi)^d \det(\Sigma)},$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean parameter, and $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ is the covariance matrix; D serves as the normalization constant such that the density function is normalized, that is, $\int p(\boldsymbol{x})dx = 1$, because calculus (by Eulur integral) shows that

$$\int \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right) \mathrm{d}\boldsymbol{x} = \sqrt{(2\pi)^d \det(\Sigma)}.$$

In order to make the PDF well defined, we need to require that Σ is symmetric and positive definite.

Marginal Distributions

If $\boldsymbol{X}$ is multivariate normal, then any of its sub-vectors is also multivariate norm. In addition, the covariance matrix of a sub-vector is simply the corresponding sub-matrix of the covariance matrix.

Specifically, assume we partition the vector $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ into two sub-vectors $\boldsymbol{X} = [\boldsymbol{X}_\alpha, \boldsymbol{X}_\beta]^\top$, where α and β is a partition of the index set $\{1, \dots, n\}$, with $\alpha \cap \beta = \emptyset$ and $\alpha \cup \beta = \{1, \dots, n\}$, and $\boldsymbol{X}_\alpha := [X_i]_{i \in \alpha}$ is the sub-vector of $\boldsymbol{X}$ on α. We can partition the mean vector and covariance matrix of $\boldsymbol{X}$ correspondingly:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_\alpha \\ \\ \boldsymbol{X}_\beta \end{bmatrix} \qquad \boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_\alpha \\ \\ \boldsymbol{\mu}_\beta \end{bmatrix}, \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{\alpha\alpha} & \boldsymbol{\Sigma}_{\alpha\beta} \\ \\ \boldsymbol{\Sigma}_{\beta\alpha} & \boldsymbol{\Sigma}_{\beta\beta} \end{bmatrix}, \tag{6.2}$$

where $\boldsymbol{\Sigma}_{\alpha\beta}$ denotes the submatrix of $\boldsymbol{\Sigma}$ on subset α and β, that is, $\boldsymbol{\Sigma}_{\alpha\beta} = [\sigma_{ij}]_{i \in \alpha, j \in \beta}$. Note that $\boldsymbol{\Sigma}_{\alpha\beta} = \boldsymbol{\Sigma}_{\beta\alpha}^\top$. Here σ_{ij} denotes the (ij)-th element of $\boldsymbol{\Sigma}$.

Theorem 6.4 If $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $\boldsymbol{X}_\alpha$ is its sub-vector on $\alpha \subseteq \{1, \dots, n\}$, then $\mathbb{E}[\boldsymbol{X}_\alpha] = \boldsymbol{\mu}_\alpha$ and $\mathrm{Cov}(\boldsymbol{X}_\alpha) = \Sigma_{\alpha\alpha}$, and hence $\boldsymbol{X}_\alpha \sim \mathcal{N}(\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha)$. Therefore, the density function of $\boldsymbol{X}_\alpha$, denoted by $p_{\boldsymbol{X}_\alpha}$, equals

$$p_{\boldsymbol{X}_\alpha}(\boldsymbol{x}_\alpha) = \frac{1}{D_\alpha} \exp\left(-\frac{1}{2} (\boldsymbol{x}_\alpha - \boldsymbol{\mu}_\alpha)^\top \boldsymbol{\Sigma}_{\alpha\alpha}^{-1} (\boldsymbol{x}_\alpha - \boldsymbol{\mu}_\alpha) \right), \qquad D_\alpha = \sqrt{(2\pi)^{d_\alpha} \det(\Sigma_\alpha)},$$

where d_α denotes the dimension of $\boldsymbol{X}_\alpha$.

Proof. The proof is obvious from the element-wise definition of mean vector and covariance matrix. ■

Covariance and Independence

The covariance matrix $\boldsymbol{\Sigma}$ of a multivariate random variable $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ encodes the dependency structure between its elements. If two elements X_i and X_j has zero covariance, that is, $\sigma_{ij} = \mathrm{Cov}(X_i, X_j) = 0$, where σ_{ij} denotes the (ij)-th element of $\boldsymbol{\Sigma}$, then X_i and X_j is in fact independent from each other, as a special property of multivariate normal distributions.

Definition 6.2 Two random variables X_i and X_j are independent with each other, denoted as $X_i \perp X_j$, if their joint density equals the product of their individual density functions, that is,

$$p_{X_i, X_j}(x_i, x_j) = p_{X_i}(x_i) p_{X_j}(x_j), \qquad \forall x_i, x_j,$$

where p_{X_i, X_j} denotes the joint density function of $[X_i, X_j]$, and p_{X_i} and p_{X_j} are the individual density functions of X_i and X_j, respectively.

Theorem 6.5 The following statements regarding two random variables (X_i, X_j) are equivalent:

1. X_i and X_j are independent, $X_i \perp X_j$;
2. $\mathrm{Cov}(h(X_i), h(X_j)) = 0$ for any function h;
3. The joint density of (X_i, X_j) can be written into

$$p_{X_i, X_j}(x_i, x_j) \propto \phi(x_i) \psi(x_j),$$

where ϕ and ψ are two non-negative functions. The notation $\propto$ denotes "proportional to", or "equals upto a constant", that is, there exists a fixed non-zero constant C, such that

$p_{X_i,X_j}(x_i, x_j) = C\phi(x_i)\psi(x_j)$. Note that because $p_{X_i,X_j}(x_i, x_j)$ is a density function, we must have $C = \int \phi(x_i)\psi(x_j)dx_i dx_j$. But it is convenient to use the $\propto$ notation in many cases, so that we do not need to keep track of the exact value of the normalization constant.

Therefore, independence $X_i \perp X_j$ implies zero-covariance $\mathrm{Cov}(X_i, X_j) = 0$, but zero-covariance does not imply independence in general. However, a key property of multivariate normal is that zero-covariance implies independence.

Theorem 6.6 Assume $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is multivariate normal, then

$$\sigma_{ij} = \mathrm{Cov}(X_i, X_j) = 0 \quad \Longleftrightarrow \quad X_i \perp X_j.$$

Proof. Following Theorem 6.4, we have $X_i \sim \mathcal{N}(\mu_i, \sigma_{ii})$ and $X_j \sim \mathcal{N}(\mu_j, \sigma_{jj})$, and

$$\begin{bmatrix} X_i \\ X_j \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_i \\ \mu_j \end{bmatrix}, \begin{bmatrix} \sigma_{ii} & \sigma_{ij} \\ \sigma_{ji} & \sigma_{jj} \end{bmatrix} \right).$$

Because $\sigma_{ij} = \sigma_{ji} = \mathrm{Cov}(X_i, X_j) = 0$, the covariance matrix is a diagonal matrix. Therefore, their joint and individual density functions are

$$\begin{aligned} p_{X_i,X_j}(x_i, x_j) &\propto \exp\left(-\frac{1}{2\sigma_{ii}}(x_i - \mu_i)^2 - \frac{1}{2\sigma_{jj}}(x_j - \mu_j)^2 \right) \\ &\propto \exp\left(-\frac{1}{2\sigma_{ii}}(x_i - \mu_i)^2 \right) \exp\left(-\frac{1}{2\sigma_{jj}}(x_j - \mu_j)^2 \right) \\ &\propto \phi(x_i)\psi(x_j), \end{aligned}$$

where we define $\phi(x_i) := \exp(-\frac{1}{2}\sigma_{ii}^{-1}(x_i - \mu_i)^2)$ and $\psi(x_j) = \exp(-\frac{1}{2}\sigma_{jj}^{-1}(x_j - \mu_j)^2)$. We used $\propto$ notation to avoiding dealing with the normalization constant.

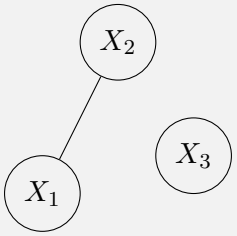
This shows that $X_i \perp X_j$ following Theorem 6.5(3). ■

Covariance Graph

It is sometimes insightful to use a graph, called covariance graph, to visualize the zero pattern of the covariance matrix, which encodes the independence relations between variables. Each node in the covariance graph represents a random variable. For example, $\boldsymbol{X} = [X_1, \dots X_d]^\top$ is represented by a graph with d nodes, whose i-th node represents X_i. The i-th and j-th nodes of the graph is connected iff $\sigma_{ij} = \mathrm{Cov}(X_i, X_j) \neq 0$. As a result, two random variables are not directly connected in the covariance graph if only if they are independent with each other. The covariance graph is undirected.

■ **Example 6.1** Consider a three-dimensional normal random variable $\boldsymbol{X} = [X_1, X_2, X_3]^\top$ with the covariance matrix shown below. From the zero elements of Σ, we can read that $X_1 \perp X_3$ and $X_2 \perp X_3$, but $X_1 \not\perp X_2$, yielding the covariance graph on the right.

$$\Sigma = \begin{bmatrix} 10 & 2 & 0 \\ 2 & 10 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



6.3 Gaussian Graphical Models

Recall that the density function of $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is

$$p(\boldsymbol{x}) = \frac{1}{D} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right), \qquad D = \sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}. \tag{6.3}$$

This is called the **standard form** of multivariate normal distributions.

Alternatively, we can write the density function into the **natural (or information) form**, defined to be

$$p(\boldsymbol{x}) = \frac{1}{C} \exp\left(-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x}\right), \tag{6.4}$$

which we denote by $\overline{\mathcal{N}}(\boldsymbol{b},\ \boldsymbol{Q})$, and $(\boldsymbol{b},\ \boldsymbol{Q})$ are called the *natural parameters* of the distribution. Similar to the constant D in (6.3), the C is a normalization constant decided by $(\boldsymbol{b},\ \boldsymbol{Q})$. It is easy to compared (6.3) and (6.4) and figure out the relation of $(\boldsymbol{b}, \boldsymbol{Q})$ and $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Lemma 6.7 The density functions in (6.3) and (6.4) are equivalent if

$$\boldsymbol{Q} = \boldsymbol{\Sigma}^{-1}, \qquad \boldsymbol{b} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \qquad C = D\exp(\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}/2).$$

That is, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is equivalent to $\bar{\mathcal{N}}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu},\ \boldsymbol{\Sigma}^{-1})$. We can see that $\boldsymbol{Q}$ equals the inverse of the covariance matrix $\boldsymbol{\Sigma}^{-1}$, and hence Q is also called the **inverse covariance matrix, or precision matrix**.

Proof. From (6.3), we have

$$\begin{aligned} p(\boldsymbol{x}) &= \frac{1}{D} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \\ &= \frac{1}{D} \exp\left(-\frac{1}{2}\boldsymbol{x}\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}\Sigma^{-1}\boldsymbol{\mu}\right). \end{aligned}$$

Comparing this with (6.4), gives the result. ■

Why do we want to use the natural form? Because it provides a highly convenient from for studying the *conditional distributions and conditional independence* of multivariate normal distributions. In contrast, the standard form is more convenient for studying the *marginal distributions and marginal independence (and correlation)*, as we see in Section 6.2.

Definition 6.3 — Conditional Independence. Let α, β, γ be three non-overlapping index subsets of $\{1, \ldots, d\}$. We say that $\boldsymbol{X}_\alpha$ and $\boldsymbol{X}_\beta$ are conditionally independent giving $\boldsymbol{X}_\gamma$, denoted by $\boldsymbol{X}_\alpha \perp \boldsymbol{X}_\beta \mid \boldsymbol{X}_\gamma$ if

$$p_{\boldsymbol{X}_{\alpha\cup\beta} \mid \boldsymbol{X}_\gamma}(\boldsymbol{x}_\alpha, \boldsymbol{x}_\beta \mid \boldsymbol{x}_\gamma) = p_{\boldsymbol{X}_\alpha \mid \boldsymbol{X}_\gamma}(\boldsymbol{x}_\alpha \mid \boldsymbol{x}_\gamma) \times p_{\boldsymbol{X}_\beta \mid \boldsymbol{X}_\gamma}(\boldsymbol{x}_\beta \mid \boldsymbol{x}_\gamma), \qquad \forall \boldsymbol{x}_\alpha, \boldsymbol{x}_\beta, \boldsymbol{x}_\gamma,$$

where $p_{\boldsymbol{X}_\alpha \mid \boldsymbol{X}_\gamma}(\boldsymbol{x}_\alpha \mid \boldsymbol{x}_\gamma)$ denotes the density function of $\boldsymbol{X}_\alpha$ conditioning on $\boldsymbol{X}_\gamma = \boldsymbol{x}_\gamma$, and

$p_{\boldsymbol{X}_{\alpha\cup\beta}\mid\boldsymbol{X}_\gamma}(\boldsymbol{x}_\alpha,\boldsymbol{x}_\beta\mid\boldsymbol{x}_\gamma)$ the joint density function of $[\boldsymbol{X}_\alpha,\boldsymbol{X}_\beta]$ conditioning on $\boldsymbol{X}_\gamma=\boldsymbol{x}_\gamma$.

Lemma 6.8 Let $\boldsymbol{X}$ be a random variable in $\mathbb{R}^d$, and α,β,γ are non-overlapping subsets of $\{1,\dots,d\}$. Let $p(\boldsymbol{x}_\alpha,\boldsymbol{x}_\beta,\boldsymbol{x}_\gamma)$ be the joint density function of $[\boldsymbol{X}_\alpha,\boldsymbol{X}_\beta,\boldsymbol{X}_\gamma]$. Then we have $\boldsymbol{X}_\alpha\perp\boldsymbol{X}_\beta\mid\boldsymbol{X}_\gamma$ iff we can write $p(\boldsymbol{x}_\alpha,\boldsymbol{x}_\beta,\boldsymbol{x}_\gamma)$ into a form of

$$p(\boldsymbol{x}_\alpha,\boldsymbol{x}_\beta,\boldsymbol{x}_\gamma)\propto\phi(\boldsymbol{x}_\alpha,\boldsymbol{x}_\gamma)\psi(\boldsymbol{x}_\beta,\boldsymbol{x}_\gamma),$$

where ϕ and ψ are two non-negative functions.

Proof. This is a direct result of Theorem 6.5(3), if we treat $\boldsymbol{x}_\gamma$ as a constant (since it is the variable that is conditioned on). ■

Therefore, in order to have $\boldsymbol{X}_\alpha\perp\boldsymbol{X}_\beta\mid\boldsymbol{X}_\gamma$, we must be able to decompose the joint density function to a product of two terms (called "factors") that only involve $\boldsymbol{x}_\alpha$ and $\boldsymbol{x}_\beta$, respectively.

Consider the special case of $X_i\perp X_j\mid\boldsymbol{X}_{\neg ij}$, where $i,j\in\{1,\dots,d\}$, $i\neq j$ and $\neg ij$ denotes the set of all the other indexes, that is, $\neg ij=\{k\colon k\notin\{i,j\},\ k=1,\dots,d\}$. The condition in Lemma 6.8 reduces to

$$p(\boldsymbol{x})\propto\phi(x_i,\boldsymbol{x}_{\neg ij})\psi(x_j,\boldsymbol{x}_{\neg ij}).$$

■ **Example 6.2** Assume $\boldsymbol{X}=[X_1,X_2,X_3]$ has the following simple joint density function

$$p(\boldsymbol{x})\propto\exp\left(-x_1^2-x_2^2-x_3^2+x_1x_2+x_2x_3\right). \tag{6.5}$$

We rewrite the density into $p(\boldsymbol{x})\propto\phi(x_1,x_2)\psi(x_3,x_2)$, with

$$\phi(x_1,x_2)=\exp\left(-x_1^2-x_2^2+x_1x_2\right),\qquad\psi(x_2,x_3)=\exp\left(-x_3^2+x_2x_3\right).$$

Therefore, we have $X_1\perp X_3\mid X_2$.

However, it is impossible to write the density into $p(\boldsymbol{x})\propto\phi(x_1,x_3)\psi(x_2,x_3)$, because there is a crossing term $\exp(x_1x_2)$ inside the density function, which prevents us from *separating* x_1 and x_2. We have $X_1\not\perp X_2\mid X_3$ in this case. Similarly, we have $X_2\not\perp X_3\mid X_1$.

More generally, it turns out we can read out all the $X_i\perp X_j\mid\boldsymbol{X}_{\neg ij}$ relations from the zero-elements of the inverse covariance matrix $\boldsymbol{Q}$.

Theorem 6.9 Assume $\boldsymbol{X}$ is multivariate normal with inverse covariance matix $\boldsymbol{Q}$. Denote by q_{ij} the ij-th element of the inverse covariance matrix $\boldsymbol{Q}$, that is, $\boldsymbol{Q}=[q_{ij}]_{ij=1}^d$. We have

$$q_{ij}=0,\qquad\Longleftrightarrow\qquad X_i\perp X_j\mid\boldsymbol{X}_{\neg ij}.$$

Comparing this with Theorem 6.6, we can see that **the zero elements of the covariance matrix encodes the (marginal) independence, while the zero elements of the inverse covariance encodes the conditional independence.**

Proof. Following (6.4), the joint density function of $\boldsymbol{X}$ is

$$\begin{aligned}
p(\boldsymbol{x}) &\propto \exp\left(-\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x}\right) \\
&\propto \exp\left(-\frac{1}{2}\sum_{k\ell} x_k q_{k\ell} x_\ell + \sum_k b_k x_k\right) \\
&\propto \exp\left(-\frac{1}{2}q_{ij}x_i x_j - \frac{1}{2}\sum_{k\neq i,j} q_{ik}x_i x_k - \frac{1}{2}\sum_{k\neq i,j} q_{jk}x_j x_k + b_i x_i + b_j x_j + \sum_{k\neq i,j} b_k x_k\right) \\
&\propto \exp\left(-\frac{1}{2}q_{ij}x_i x_j\right)\phi(x_i, \boldsymbol{x}_{\neg ij})\psi(x_j, \boldsymbol{x}_{\neg ij}),
\end{aligned}$$

where we define

$$\begin{aligned}
\phi(x_i, \boldsymbol{x}_{\neg ij}) &= \exp\left(-\frac{1}{2}\sum_{k\neq i,j} q_{ik}x_i x_k + b_i x_i + sum_{k\neq i,j} b_k x_k\right), \\
\psi(x_j, \boldsymbol{x}_{\neg ij}) &= \exp\left(-\frac{1}{2}\sum_{k\neq i,j} q_{jk}x_j x_k + b_j x_j\right).
\end{aligned}$$

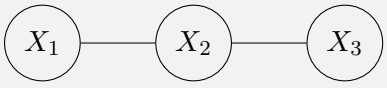
Therefore, if $q_{ij} = 0$, we have $p(\boldsymbol{x}) \propto \phi(x_i, \boldsymbol{x}_{\neg ij})\psi(x_j, \boldsymbol{x}_{\neg ij})$, which implies that $X_i \perp X_j \mid \boldsymbol{X}_{\neg ij}$ by Lemma 6.8. The idea is that $\exp(-\frac{1}{2}q_{ij}x_i x_j)$ is the only cross term that involves both x_i and x_j in the density function. If $q_{ij} = 0$, there is no cross term of x_i and x_j, which implies conditional independence. ■

Markov Graph

Similar to covariance graph, we can represent the zero pattern of the covariance matrix using a graph to visualize the *conditional independence* relations between variables. Such graphs are called Markov graphs. Each variable X_i is represented by a node in the Markov graph, and X_i and X_j are connected in the Markov graph iff $q_{ij} \neq 0$. Two random variables are not directly connected in the Markov graph if only if they are conditional independent with all the other variables fixed.

■ **Example 6.3** Comparing the density function (6.5) in Example 6.2 with the natural form (6.4), we can see that $\boldsymbol{X} = [X_1, X_2, X_3]$ in Example 6.2 is multivariate normal with the inverse covariance matrix shown below, from which we can read $X_1 \perp X_3 \mid X_2$ but $X_1 \not\perp X_2 \mid X_3$ and $X_2 \not\perp X_3 \mid X_1$. It yields the Markov graph shown on the right.

$$\boldsymbol{Q} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$$



The covariance graph and Markov graph represent different information of multivariate normal distributions. They are usually very different from each other.

■ **Example 6.4** By matrix inversion, we can calcualte the covariance matrix of the random

variable in Example 6.3,

$$\boldsymbol{\Sigma} = \boldsymbol{Q}^{-1} = \begin{bmatrix} 3/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 3/4 \end{bmatrix}.$$

Since all its elements are non-zero, the covariance graph is fully connected, even though its Markov graph is a chain.

7. Kernel Method

The standard linear regression assumes the true models are linear functions of the input variables:

$$f(\boldsymbol{x};\ \boldsymbol{\theta}) = \sum_{\ell=1}^{d} \theta_\ell x_\ell = \boldsymbol{\theta}^\top \boldsymbol{x}.$$

But linear models can not capture nonlinear relations. A basic approach to capture nonlinear relations is to assume the models are linear functions of a set of nonlinear basis functions:

$$f(\boldsymbol{x};\ \boldsymbol{\theta}) = \sum_{\ell=1}^{m} \theta_\ell \phi_\ell(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{x}),$$

where $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \ldots, \phi_m(\boldsymbol{x})]^\top$ is a set of basis functions that is believed to capture important nonlinear pattern regarding the input, and $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_m]^\top$ is a coefficient vector applied on $\boldsymbol{\phi}(\boldsymbol{x})$. For example, in the case of polynomial regression, we assume $\phi_\ell(\boldsymbol{x}) = x^{\ell-1}$ and hence

$$f(\boldsymbol{x};\ \boldsymbol{\theta}) = \sum_{\ell=0}^{m-1} \theta_\ell x^\ell.$$

For a fixed $\boldsymbol{\phi}(\boldsymbol{x})$, the coefficient $\boldsymbol{\theta}$ is estimated the same way as linear regression, and hence has a simple closed form solution.

However, it would be much more desirable to be able to automatically and adaptively construct the basis functions in a data-driven way to achieve the best performance. We will discussion two methods to achieve *adaptive basis functions*, including kernel method in this chapter and neural networks in Chapter 8.

7.1 Kernel Regression

Let $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}')$ be a similarity measure between two points $\boldsymbol{x}$ and $\boldsymbol{x}'$. We assume it is symmetric in that $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{k}(\boldsymbol{x}', \boldsymbol{x})$ and call it a kernel function. A typical example of kernel function is the

Gaussian radial basis function (RBF) kernel:

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2h^2}\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|_2^2\right),$$

where h is a positive parameter called the *bandwidth*. Given a dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$, we may construct a kernel representation of a point $\boldsymbol{x}$ by comparing it with each observed data point in the dataset:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}_n) \end{bmatrix},$$

where $\{\boldsymbol{x}_i\}$ are the fixed observed data points, and $\boldsymbol{x}$ is the variable of the function.

Defining $\boldsymbol{\phi}$ this way allows us to represent each data point using its relative similarity with the other data points in the dataset. This yields a simple yet powerful adaptive representation of the data points.

We can use these feature to construct linear function class:

$$f(\boldsymbol{x};\, \boldsymbol{\theta}) = \sum_i \theta_i \phi_i(\boldsymbol{x}) = \sum_i \theta_i \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}^{(i)}),$$

where θ_i and is the coefficient of $\phi_i(\boldsymbol{x}) = \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}^{(i)})$.

Note that as the size n of the dataset increases, the dimension of the feature $\boldsymbol{\phi}(\boldsymbol{x})$ also increases. Therefore, if we have an infinite number of data points, we expect to get an infinite dimensional representation of the feature. The more data we have, the larger representation power of $\boldsymbol{\phi}$ is.

The parameter $\boldsymbol{\theta}$ can be estimated by minimizing the empirical loss function,

$$\begin{aligned} L(\boldsymbol{\theta}) &= \sum_{i=1}^n (y_i - f(\boldsymbol{x}^{(i)};\, \boldsymbol{\theta}))^2 \\ &= \sum_{i=1}^n \left(y_i - \sum_{j=1}^n \theta_j \boldsymbol{k}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})\right)^2 \\ &= \|\boldsymbol{Y} - \boldsymbol{K}\boldsymbol{\theta}\|_2^2, \end{aligned}$$

where $\boldsymbol{Y} = [y_1, \ldots, y_n]^\top \in \mathbb{R}^{n\times 1}$, and $\boldsymbol{K} = [\boldsymbol{k}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})]_{ij=1}^n \in \mathbb{R}^{n\times n}$. The $n \times n$ matrix $\boldsymbol{K}$ is often called the gram matrix. Then estimating $\boldsymbol{\theta}$ reduces to

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{Y} - \boldsymbol{K}\boldsymbol{\theta}\|_2^2. \tag{7.1}$$

Assume matrix $\boldsymbol{K}$ is invertiable. Solving this equation gives

$$\boldsymbol{\theta} = \boldsymbol{K}^{-1}\boldsymbol{Y}.$$

This is fitting the curve exactly (or interpolation), because we have n data points and n parameters.

Since (7.1) exactly fits all the data and may have the risk of overfitting, we usually introduce a regularization term in kernel-based regression:

$$\min_{\boldsymbol{\theta}} \|Y - \boldsymbol{K}\boldsymbol{\theta}\|_2^2 + \alpha\Phi(\boldsymbol{\theta}), \tag{7.2}$$

where $\Phi(\boldsymbol{\theta})$ is some regularization term and α the regularization coefficient. A natural choice would be the L2 norm $\Phi(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$. However, it is more typical to use a different regularization,

$$\Phi(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{K} \boldsymbol{\theta}, \tag{7.3}$$

which incorporates the kernel information into the regularization.

Remark 7.1 In order to make the regularization in (7.3) non-negative all the time, $\boldsymbol{K}$ need to be positive semi-definite. The kernel $\boldsymbol{k}(\cdot,\cdot)$ is called a *positive definite kernel* if the gram matrix $\boldsymbol{K} = [\boldsymbol{k}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})]_{ij=1}^n \in \mathbb{R}^{n\times n}$ is a *positive semi-definite matrix* for any n and any points $\{\boldsymbol{x}^{(i)}\} \subset \mathbb{R}^d$, that is, $\boldsymbol{v}^\top \boldsymbol{K} \boldsymbol{v} \geq 0$ for any $\boldsymbol{v} \in \mathbb{R}^n$ and $\boldsymbol{v} \neq 0$. Note that the use of "positive definite" and "positive semi-definite" are inconsistent for the gram matrix $\boldsymbol{K}$ and its kernel $\boldsymbol{k}(\cdot,\cdot)$, due to terminology convention. If matrix $\boldsymbol{K}$ is positive definite for all $\{\boldsymbol{x}^{(i)}\}$, kernel $\boldsymbol{k}(\cdot,\cdot)$ is called to be a strictly positive definite kernel. ∎

Remark 7.2 Why is it more reasonable to use $\Phi(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{K} \boldsymbol{\theta}$ over the typical L2 norm $\Phi(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$? To draw understanding, consider the case when we have three data points $[\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}]$, such that $\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(2)} \neq \boldsymbol{x}^{(3)}$. The kernel function class is

$$\begin{aligned} f(\boldsymbol{x}) &= \theta_1 \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}^{(1)}) + \theta_2 \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}^{(2)}) + \theta_3 \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}^{(3)}) \\ &= (\theta_1 + \theta_2) \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}^{(1)}) + \theta_3 \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}^{(3)}). \end{aligned}$$

Therefore, in this case, it is more reasonable to use $(\theta_1 + \theta_2)^2 + \theta_3^2$ as the penalty, instead of the standard L2 norm $\|\boldsymbol{\theta}\|_2^2 = \theta_1^2 + \theta_2^2 + \theta_3^2$.

It turns out that $\Phi(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{K} \boldsymbol{\theta}$ exactly does what we want. To see this, assume we use the Gaussian RBF kernel $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 / h)$, for which we have $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}) = 1$ for $\forall \boldsymbol{x}$. If $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(3)}$ are very different, we have $\boldsymbol{k}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(3)}) \approx 0$. Then the kernel gram matrix looks like

$$\boldsymbol{K} \approx \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which suggests that

$$\Phi(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{K} \boldsymbol{\theta} \approx (\theta_1 + \theta_2)^2 + \theta_3^2.$$

∎

Problem 7.1 Why is it more reasonable to make $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}')$ a similarity measure rather than a distance measure (which is larger when $\boldsymbol{x}$ and $\boldsymbol{x}'$ are more different from each other)?

7.2 Kernel as Infinite Dimensional Features (Optional)

Assume $\{\psi_\ell(\boldsymbol{x}) : \ell = 1, \cdots, \infty\}$ is an infinite number of "basis functions" whose linear combination can approximate arbitrary "nice" or "well-behaved" functions (such as continuous functions, or integral functions), that is, for any nice function f, we can find a set of coefficients $\{w_\ell\}_{\ell=1}^\infty$, such that

$$f(\boldsymbol{x}) = \sum_{\ell=1}^{\infty} w_\ell \psi_\ell(\boldsymbol{x}).$$

For example, in the case when $\boldsymbol{x} \in \mathbb{R}$, ψ_ℓ can be ℓ-th order polynomials so that $f(\boldsymbol{x})$ can be approximated by Taylor series, or Fourier series $\psi_{2\ell}(\boldsymbol{x}) = \cos(2\pi\ell\boldsymbol{x})$ and $\psi_{2\ell+1}(\boldsymbol{x}) = \sin(2\pi\ell\boldsymbol{x})$.

Therefore, we can consider the following *infinite dimensional* regression problem:

$$\min_{\{w_\ell\}_{\ell=1}^{\infty}} \sum_{i=1}^{n} \left(y^{(i)} - \sum_{\ell=1}^{\infty} w_\ell \psi_\ell(\boldsymbol{x}^{(i)}) \right)^2 + \sum_{\ell=1}^{\infty} \alpha_\ell w_\ell^2 \tag{7.4}$$

where $\sum_{\ell=1}^{\infty} \alpha_\ell w_\ell^2$ is a regularization term that prevents the norm of $\{w_\ell\}$ from being too large. It is like a "weighted" L2 norm, where each w_ℓ^2 term is weighted by a positive coefficient α_ℓ. So α_ℓ controls how important feature $\psi_\ell(\boldsymbol{x})$ is in the regression.

A remarkable fact is that this optimization is solvable, despite being infinite dimensional. And it exactly reduces to learning a kernel representation in Section 7.1.

Theorem 7.1 Given ψ_ℓ and $\lambda_\ell := 1/\alpha_\ell > 0$, define

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\ell=1}^{\infty} \lambda_\ell \psi_\ell(\boldsymbol{x}) \psi_\ell(\boldsymbol{x}').$$

Assume the infinite sum above converges absolutely. Let $\{\hat{w}_\ell\}_{\ell=1}^{\infty}$ be the optimal solution of (7.4) and $\hat{f}(\boldsymbol{x}) = \sum_{\ell=1}^{\infty} \hat{w}_\ell \psi_\ell(\boldsymbol{x})$ the corresponding function, then $\hat{f}(\boldsymbol{x})$ can be represented in the following kernel form:

$$\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_i \boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}^{(i)})$$

where $\theta_i = y^{(i)} - \hat{f}(\boldsymbol{x}^{(i)})$, for which we have

$$\boldsymbol{\theta}^\top \boldsymbol{K} \boldsymbol{\theta} = \sum_{\ell=1}^{\infty} \alpha_\ell \hat{w}_\ell^2.$$

Therefore, (7.4) is equivalent to (7.2) with $\Phi(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{K} \boldsymbol{\theta}$ and $\alpha = 1$.

Proof. At the optimal point, we have from the zero gradient condition:

$$-2 \sum_{i=1}^{n} \theta_i \psi_\ell(\boldsymbol{x}^{(i)}) + 2\alpha_\ell \hat{w}_\ell = 0.$$

This gives

$$\hat{w}_\ell = \frac{1}{\alpha_\ell} \sum_{i=1}^{n} \theta_i \psi_\ell(\boldsymbol{x}^{(i)}) = \sum_{i=1}^{n} \lambda_\ell \theta_i \psi_\ell(\boldsymbol{x}^{(i)}) \tag{7.5}$$

This suggests that $\hat{w}_\ell$ is a linear combination of $\psi_\ell(\boldsymbol{x}^{(i)})$, $i = 1, \ldots, n$. Therefore,

$$\begin{aligned}
\hat{f}(\boldsymbol{x}) &= \sum_{\ell=1}^{\infty} \hat{w}_\ell \psi_\ell(\boldsymbol{x}) \\
&= \sum_{\ell=1}^{\infty} \sum_{i=1}^{n} \lambda_\ell \theta_i \psi_\ell(\boldsymbol{x}^{(i)}) \psi_\ell(\boldsymbol{x}) \\
&= \sum_{i=1}^{n} \sum_{\ell=1}^{\infty} \lambda_\ell \theta_i \psi_\ell(\boldsymbol{x}^{(i)}) \psi_\ell(\boldsymbol{x}) \\
&= \sum_{i=1}^{n} \theta_i \sum_{\ell=1}^{\infty} \lambda_\ell \psi_\ell(\boldsymbol{x}^{(i)}) \psi_\ell(\boldsymbol{x}) \\
&= \sum_{i=1}^{n} \theta_i \boldsymbol{k}(\boldsymbol{x}^{(i)}, \boldsymbol{x}).
\end{aligned}$$

Further, plugging (7.5)

$$\begin{aligned}
\sum_{\ell=1}^{\infty} \alpha_\ell \hat{w}_\ell^2 &= \sum_{\ell=1}^{\infty} \alpha_\ell \left(\sum_{i=1}^{n} \lambda_\ell \theta_i \psi_\ell(\boldsymbol{x}^{(i)}) \right)^2 \\
&= \sum_{\ell=1}^{\infty} \alpha_\ell \left(\sum_{ij=1}^{n} \lambda_\ell \theta_i \psi_\ell(\boldsymbol{x}^{(i)}) \lambda_\ell \theta_j \psi_\ell(\boldsymbol{x}^{(j)}) \right) \\
&= \sum_{ij=1}^{n} \theta_i \theta_j \left(\sum_{\ell=1}^{\infty} \lambda_\ell \psi_\ell(\boldsymbol{x}^{(i)}) \psi_\ell(\boldsymbol{x}^{(j)}) \right) \\
&= \sum_{ij=1}^{n} \theta_i \theta_j \boldsymbol{k}(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \boldsymbol{\theta}^\top \boldsymbol{K} \boldsymbol{\theta}.
\end{aligned}$$

■

Here, we connect the kernel and basis function perspectives via

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\ell} \lambda_\ell \psi_\ell(\boldsymbol{x}) \psi_\ell(\boldsymbol{x}'). \tag{7.6}$$

In fact, a stronger claim can be made: any continuous positive definite kernel can be represented by (7.6) with $\lambda_\ell \geq 0$. This result is known as Mercer's theorem.

Let us verify this for the Gaussian RBF kernel on $\mathbb{R}$. We have for $\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|^2)$,

$$\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\ell=0}^{\infty} \lambda_\ell \psi_\ell(\boldsymbol{x}) \psi_\ell(\boldsymbol{x}'),$$

where

$$\psi_\ell(\boldsymbol{x}) = \exp(-\gamma \boldsymbol{x}^2) \boldsymbol{x}^\ell,$$

and

$$\lambda_\ell = \frac{(2\gamma)^\ell}{\ell!}.$$

This can be derived by applying Taylor expansion on $\exp(\cdot)$.

8. Neural Networks

Recall that a basic approach for construct nonlinear function classes is using basis functions,

$$f(\boldsymbol{x};\ \boldsymbol{a}) = \sum_{\ell=1}^{m} a_\ell \phi_\ell(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{\phi}(\boldsymbol{x}),$$

where $\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \ldots, \phi_m(\boldsymbol{x})]^\top$ is a set of basis functions that is believed to capture important information regarding the input, and $\boldsymbol{a} = [a_1, \ldots, a_m]^\top$ is a coefficient vector applied on $\boldsymbol{\phi}(\boldsymbol{x})$. For example, in the case of polynomial regression, we assume $\phi_\ell(x) = x^{\ell-1}$ and hence

$$f(x; \boldsymbol{a}) = \sum_{\ell=0}^{m-1} a_\ell x^\ell.$$

For a fixed $\boldsymbol{\phi}(\boldsymbol{x})$, the coefficient $\boldsymbol{a}$ is estimated the same way as linear regression, by minimizing mean square error.

The key question, however, is how to generate basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ adaptively based on the observed data to achieve the best result.

Neural networks is a simple approach to enable adaptive basis functions. The idea is embarrassingly simple. We simply assume the basis functions $\boldsymbol{\phi}(\boldsymbol{x})$ has some parameter and jointly estimate them with $\boldsymbol{a}$ by minimizing the loss function. Specifically, in neural networks, each basis function $\phi(\boldsymbol{x})$ is assumed to have a form of

$$\phi(\boldsymbol{x};\ \boldsymbol{w}) = \sigma\left(\sum_{i=1}^{d} w_i x_i + w_0\right),$$

where $\boldsymbol{w} := \{w_i\}_{i=0}^d$ is a set of coefficients that will be estimated from the data, and $\sigma(\cdot)$ is a one-dimensional nonlinear function called *activation function*. So here we take a linear combination of the features (like what we did in linear regression), and push it into a nonlinear activation function $\sigma(\cdot)$. Such a basis function is called a *neuron*, and the parameters $\boldsymbol{w}$ is called the weights of the neuron.

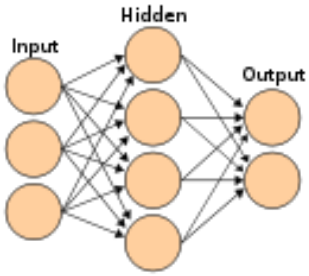


Figure 8.1: A Two-Layer Neural Network

The are several different common choices of activation functions:

$$\begin{aligned} \textbf{Rectified linear unit (ReLU):} \quad & \sigma(t) = \max(0, t) \\ \textbf{Sigmoid:} \quad & \sigma(t) = \frac{\exp(t)}{1+\exp(t)} \\ \textbf{Thresholding:} \quad & \sigma(t) = \mathbb{I}(t \geq 0). \end{aligned}$$

A neural network consists of several number of neurons, each of which has it own weight,

$$f(\boldsymbol{x};\ [\boldsymbol{a}, \boldsymbol{W}]) = \sum_{\ell=1}^{m} a_\ell \sigma\left(\sum_{\ell=1}^{d} w_{\ell,i} x_i + w_{\ell,0}\right) = \sum_{\ell=1}^{m} a_\ell \sigma(\boldsymbol{w}_\ell^\top [1; \boldsymbol{x}]),$$

where $\boldsymbol{W} = \{\boldsymbol{w}_\ell\} = \{w_{\ell,i}\}_{\ell,i}$ is a matrix that consists of the weights of all the neurons and $\boldsymbol{w}_\ell = \{w_{\ell,i}\}_i$ is the weight of the ℓ-th neuron , and m is the number of neurons which should be decided by the user. The coefficients $\{w_{\ell,0}\}_\ell$ are called the bias terms. This function consists of two sets of parameters, $\boldsymbol{a}$ and $\boldsymbol{W} = \{\boldsymbol{w}_\ell\}_{\ell=1}^m$, which can be estimated from data by minimizing the loss function,

$$\min_{\boldsymbol{a},\boldsymbol{W}} \left\{ L(\boldsymbol{a}, \boldsymbol{W}) := \mathbb{E}_{\mathcal{D}} \left[(y - f(\boldsymbol{x};\ [\boldsymbol{a}, \boldsymbol{W}]))^2 \right] \right\}. \tag{8.1}$$

Here we need to jointly optimize $[\boldsymbol{a}, \boldsymbol{W}]$ to minimize the loss, which can be solved with gradient descent or stochastic gradient descent.

Neural Networks Yield Non-convex Optimization

A key fact is that the optimization in (8.1) almost always yield non-convex optimization, except very simple cases such as when the activation function is linear (i.e. $\sigma(t) = t$).

A main reason for this is that the different neurons can be exchanged with each other without changing the function that the neural network represents. So if there is one optimum of the loss, then by exchanging the neurons, we can obtain multiple symmetric optima that yields the sample loss functions. Typically these optima obtained by permutation are separated by points with higher losses, and hence yield a multi-modal, non-convex loss landscape (see Figure 8.2).

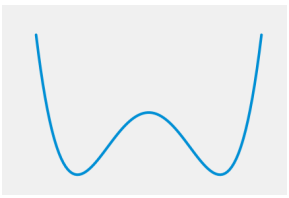


Figure 8.2: A loss function with two symmetric local optima.

To be more concrete, consider a simple neuron network with two neurons,

$$f(x; [\boldsymbol{w}_1, \boldsymbol{w}_2]) = \sigma(\boldsymbol{w}_1^\top \boldsymbol{x}) + \sigma(\boldsymbol{w}_2^\top \boldsymbol{x}).$$

Let us assume that $[\boldsymbol{w}_1^*, \boldsymbol{w}_2^*]$ is an optimal point and the optimal weights of the two neurons are different, that is, $\boldsymbol{w}_1^* \neq \boldsymbol{w}_2^*$ (otherwise the network only has a single neuron). Note that the order of the two neurons can be exchanged without changing the output the neural network:

$$f(x, [\boldsymbol{w}_1, \boldsymbol{w}_2]) = f(x, [\boldsymbol{w}_2, \boldsymbol{w}_1]).$$

Therefore, $[\boldsymbol{w}_2^*, \boldsymbol{w}_1^*]$ yields the same value as $[\boldsymbol{w}_1^*, \boldsymbol{w}_2^*]$ and is hence also an optimal point. If the loss is convex, then the middle point of $[\boldsymbol{w}_1^*, \boldsymbol{w}_2^*]$ and $[\boldsymbol{w}_2^*, \boldsymbol{w}_1^*]$, which is $[\bar{\boldsymbol{w}}^*, \bar{\boldsymbol{w}}^*]$ with $\bar{\boldsymbol{w}}^* = (\boldsymbol{w}_1^* + \boldsymbol{w}_2^*)/2$, must be also an optimal point. But this is unlikely the case, because $[\bar{\boldsymbol{w}}^*, \bar{\boldsymbol{w}}^*]$ only utilizes a single neuron, and is less flexible than the case when two neurons are different.

Exercise 8.1 Consider a simple neuron network

$$f(x; [w_1, w_2]) = \sigma(x - w_1) + \sigma(x - w_2),$$

where $x, w_1, w_2 \in \mathbb{R}$. Simulate a random dataset $\mathcal{D} := \{x^{(i)}, y^{(i)}\}_{i=1}^n$, and plot the contour of the loss function of $[w_1, w_2]$ on a 2D plane. Show that this loss function is non-convex. Repeat this for different activation functions, including ReLU, Sigmoid, and Binary. ■

Because of the non-convexity, finding the global optimum of (8.1) is impossible. Fortunately, in practice, people find that is it is often sufficient to find a local optima using stochastic gradient descent (see Section 2.3).

A. Appendix

A.1 Probability Background

A random variable is a variable whose values depend on outcomes of a random phenomenon. The behavior of a random variable is described by its probability distribution, which specifies the probability of its values. Random variables can be discrete, that is, taking any of a specified finite or countable list of values, endowed with a probability mass function characteristic of the random variable's probability distribution; or continuous, taking any numerical value in an interval or collection of intervals, whose behavior is characterized by a probability density function; or a mixture of both types.

Discrete Random Variables

A random variable is discrete if it takes values from a finite or countable list of values. For example, for random variable X that takes values in $\{a_1, \ldots, a_k\}$, it is behavior is characterized by the probability $\Pr(X = a_i)$ that it equals each value a_i in the list. These probabilities, which form a distribution, should satisfy

$$\Pr(X = a_i) \geq 0 \quad \text{(Non-negativity)}, \qquad \sum_{i=1}^{k} \Pr(X = a_i) = 1 \quad \text{(Sum-to-one)}.$$

The function $f(a_i) := \Pr(X = a_i)$ is called the probability mass function of X. The expectation or mean of X is the weighted average of the possible values that X can take, each value being weighted according to the probability of that event occurring.

$$\textbf{Expectation}: \qquad \mathbb{E}[X] = \sum_{i=1}^{k} a_i \Pr(X = a_i),$$

The variance of X is a measurement of how spread of the possible values are, defined to be the expectation of the squared deviation of X from its expectation $\mathbb{E}[X]$.

$$\textbf{Variance}: \qquad \mathrm{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \sum_{i=1}^{k} (a_i - \mathbb{E}[X])^2 \Pr(X = a_i).$$

Theorem A.1

$$\mathrm{Var}[X] = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2 \geq 0.$$

Proof.

$$\begin{aligned}
\mathrm{Var}[X] &= \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] \\
&= \mathbb{E}\left[X^2 - 2\mathbb{E}[X]X + (\mathbb{E}[X])^2\right] \\
&= \mathbb{E}[X^2] - 2\mathbb{E}[X]\mathbb{E}[X] + (\mathbb{E}[X])^2 \\
&= \mathbb{E}[X^2] - (\mathbb{E}[X])^2.
\end{aligned}$$

■

The square root of variance is called the standard deviation:

$$\textbf{Standard deviation:} \qquad \mathrm{Std}(X) := \sqrt{\mathrm{Var}(X)} = \sqrt{\mathbb{E}\left[(X - \mathbb{E}[X])^2\right]}.$$

■ **Example A.1** Assume X is a random variable taking values in $\{0, 1\}$, with $\Pr(X = 1) = \alpha$, where $0 \leq \alpha \leq 1$. Then

$$\Pr(X = 0) = 1 - \Pr(X = 1) = 1 - \alpha.$$

$$\mathbb{E}[X] = 0 \times \Pr(X = 0) + 1 \times \Pr(X = 1) = 0 + 1 \times \alpha = \alpha.$$

$$\begin{aligned}
\mathrm{Var}(X) &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\
&= \mathbb{E}[(X - \alpha)^2] \\
&= (0 - \alpha)^2 \Pr(X = 0) + (1 - \alpha)^2 \Pr(X = 1) \\
&= \alpha^2(1 - \alpha) + (1 - \alpha)^2 \alpha \\
&= \alpha(1 - \alpha).
\end{aligned}$$

Joint Distributions

Two random variables X and Y, taking values in $\{a_1, \ldots, a_k\}$ and $\{b_1, \ldots, b_\ell\}$, respectively, can form a joint distribution on all the possible pairs of values of (X, Y):

$$p_{ij} = \Pr(X = a_i,\ Y = b_j).$$

We can think p_{ij} as forming a two dimensional table. As a distribution, it should also satisfy the non-negativity and normalization properties:

$$p_{ij} \geq 0 \quad \text{(Non-negativity)}, \qquad \sum_{i=1}^{k} \sum_{j=1}^{\ell} p_{ij} = 1 \quad \text{(Sum-to-one)}.$$

The marginal distribution of X is

$$\Pr(X = a_i) = \sum_{j=1}^{\ell} \Pr(X = a_i,\ Y = b_j) = \sum_{j=1}^{\ell} p_{ij}.$$

The conditional distribution is

$$\Pr(X = a_i \mid Y = b_j) = \frac{\Pr(X = a_i,\ Y = b_j)}{\Pr(Y = b_j)} = \frac{\Pr(X = a_i,\ Y = b_j)}{\sum_{j=1}^{\ell} \Pr(X = a_i,\ Y = b_j)}.$$

For notation convenience, we can drop a_i and b_j and simply write the formula above as

$$\Pr(X \mid Y) = \frac{\Pr(X, Y)}{\Pr(Y)}.$$

Theorem A.2 — Chain Rule.

$$\Pr(X, Y) = \Pr(X \mid Y)\Pr(Y).$$

Note that the chain rule can be applied starting from either X or Y:

$$\Pr(X, Y) = \Pr(X \mid Y)\Pr(Y) = \Pr(Y \mid X)\Pr(X).$$

This yields the important Bayesian rule.

Theorem A.3 — Bayesian Rule.

$$\Pr(Y \mid X) = \frac{\Pr(X \mid Y)\Pr(Y)}{\Pr(X)}.$$

The power of Bayesian rule is that it allows us to invert the direction of the conditional probability, transforming $\Pr(X \mid Y)$ to $\Pr(Y \mid X)$.

■ **Example A.2** Assume we know that it rains in Austin with probability α_1; if it rains, there will be traffic jam with probability α_2 and if it does not rain, there will be traffic jam with probability α_3.

Question: if you experience a traffic jam, what is the probability that it rains today?

Continuous Random Variables

When random variable X is continuous, e.g., taking values from $\mathbb{R}$, its distribution should be characterized using its **probability density functions** (PDF).

The idea is that we can partition the real line to small bins and count the probability divided by the length of the bin and take the limit as the density function:

$$p(x) = \lim_{h \to 0} \frac{\Pr(X \in [x - h,\ x + h])}{2h}.$$

The density function should satisfy

$$\begin{aligned} &\text{Normalization :} && \int p(x)dx = 1 \\ &\text{Non-negativity :} && p(x) \geq 0. \end{aligned}$$

The expectation of a function $f(x)$ is defined by integration (instead of summation):

$$\mathbb{E}[f(X)] = \int p(x)f(x)dx.$$

Related, the expectation and variance are defined as

$$\mathbb{E}[X] = \int_{-\infty}^{+\infty} xp(x)dx$$
$$\mathrm{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \int (x - \mathbb{E}[X])^2 p(x)dx = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

■ **Example A.3 — Uniform Distribution.** Consider uniform distribution:

$$f(x) = \begin{cases} 1/a & \text{if } x \in [0, a] \\ 0 & \text{if otherwise} \end{cases}$$

Calculate its mean:

$$\mathbb{E}[X] = \int xf(x)dx = \int_0^a \frac{x}{a}dx = \frac{a^2}{2a} = \frac{a}{2}.$$
$$\mathbb{E}[X^2] = \int x^2 f(x)dx = \int_0^a \frac{x^2}{a}dx = \frac{a^3}{3a} = \frac{a^2}{3}.$$

From this we can calculate the variance:

$$\mathrm{Var}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = \frac{a^2}{3} - \left(\frac{a}{2}\right)^2 = \frac{a^2}{12}.$$

Theorem A.4 — Mean and Variance Under Linear Transform. Let X be a random variable, and a, b two deterministic constants. We have

$$\mathbb{E}[aX + b] = a\mathbb{E}[X] + b, \qquad \mathrm{Var}(aX + b) = a^2\mathrm{Var}(X).$$

Proof. Let $f(x)$ be the density function of X.

$$\mathbb{E}[aX + b] = \int (ax + b)f(x)dx = a\int xf(x)dx \; + \; b = a\mathbb{E}[X] + b.$$

$$\begin{aligned}\mathrm{Var}(aX + b) &= \mathbb{E}[(aX + b - \mathbb{E}[aX + b])^2] \\ &= \mathbb{E}[(aX + b - a\mathbb{E}[X] - b)^2] \\ &= \mathbb{E}[a^2(X - \mathbb{E}[X])^2] = a^2\mathbb{E}[(X - \mathbb{E}[X])^2] = a^2\mathrm{Var}(X).\end{aligned}$$

■

Theorem A.5 — Expectation of Sum of Two Random Variables. Let (X, Y) is a pair of random variables on the same sample space. We have

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y].$$

Proof. Let $f_{XY}(x,y)$ be the joint density function of (X,Y). Related, let $f_X(x)$ and $f_Y(y)$ the marginal distributions of X and Y, respectively. We have

$$f_X(x) = \int f_{XY}(x,y)dy, \qquad f_Y(y) = \int f_{XY}(x,y)dx.$$

$$\begin{aligned}
\mathbb{E}[X+Y] &= \int (x+y) f_{XY}(x,y)dxdy \\
&= \int x f_{XY}(x,y)dxdy \; + \; \int y f_{XY}(x,y)dxdy \\
&= \int x f_X(x)dx + \int y f_Y(y)dy \\
&= \mathbb{E}[X] + \mathbb{E}[Y].
\end{aligned}$$

■

Theorem A.6 If a random variable X has density $p_X(x)$, Let $Y = aX + b$, where $a \neq 0$. Then the density of Y is

$$p_Y(y) = \frac{1}{a} p_X\left(\frac{y-b}{a}\right).$$

Proof. Recall that for any function h, we have

$$\mathbb{E}[h(Y)] = \int p_Y(y)h(y)dy.$$

Reversely, if we can find a function $f(y)$, such that $\mathbb{E}[h(Y)] = \int f(y)h(y)dy$ for any function h, then we can claim $p_Y(y) = f(y)$ (see Lemma A.7 below). We use this fact to derive p_Y.

For any h, we have

$$\mathbb{E}[h(Y)] = \mathbb{E}[h(aX+b)] = \int p_X(x)h(ax+b)dx = \int \frac{1}{a} p_X\left(\frac{y-b}{a}\right) h(y)dy.$$

Since this holds for any function h, from this we can claim that $p_Y(y) = \frac{1}{a} p_X\left(\frac{y-b}{a}\right)$. ■

Lemma A.7 Let X be a random variable. Assume there exists a function f, such that

$$\mathbb{E}[h(X)] = \int f(x)h(x)dx,$$

for any function h (for which the expectation and integral are defined), then f equals the probability density function of X.

Proof. This can be viewed as one way to define probability density function mathematically. In fact, taking $h(x) = \mathbb{I}(x \in A)$ for any set A, we have

$$\Pr(X \in A) = \mathbb{E}[\mathbb{I}(X \in A)] = \int \mathbb{I}(x \in A) f(x)dx = \int_{x \in A} f(x)dx.$$

This shows that the probability $\Pr(X \in A)$ of X belong to any set A equals the integration of $f(x)$ on A, and hence $f(x)$ is the density function of X ■

■ **Example A.4 — Gaussian Distribution.** A random variable Z is called standard normal (or Gaussian) if it has a density of standard Bell shape curve:

$$p(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right).$$

This distribution is denoted by $\mathcal{N}(0,1)$. Using calculus, we can verify that $\mathbb{E}[Z] = 0$ and $\mathrm{Var}(Z) = 1$. The factor $1/\sqrt{2\pi}$ arises because $\int \exp(-z^2/2)dz = 1/\sqrt{2\pi}$; it is used to ensure that the density integrates to one, that is, $\int p(z)dz = 1$.

A random variable X is Gaussian if it can be expressed as

$$X = \mu + \sigma Z,$$

where $Z \sim \mathcal{N}(0,1)$. Following Theorem A.11, we have $\mathbb{E}[X] = \mu$ and $\mathrm{Var}(X) = \sigma^2$. Therefore, the distribution of X is denoted by $\mathcal{N}(\mu, \sigma^2)$. Derivation shows that the density of X is a Bell curve, center at μ and scaled with σ:

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

Joint Distributions

The joint distribution of a pair of continuous random variables (X, Y) is characterized by a joint density function, which we denote by $p_{XY}(x, y)$. In this case, the joint density function is a bivariate function that should satisfy

$$p_{XY}(x,y) \geq 0, \qquad \int\int p_{XY}(x,y)dxdy = 1.$$

Related, the marginal distribution of X and Y can be obtained by integration:

$$p_X(x) = \int p_{XY}(x,y)dy, \qquad p_Y(y) = \int p_{XY}(x,y)dx.$$

■ **Example A.5** Assume (X, Y) is a random variable on $[-a, a] \times [-b, b]$. This means that

$$p_{XY}(x,y) = \begin{cases} \frac{1}{4ab} & \text{if } (x,y) \in [-a,a] \times [-b,b] \\ 0 & \text{if otherwise.} \end{cases}$$

For two random variables (X, Y) with joint density $p_{XY}(x, y)$, their covariance is defined by

Covariance:
$$\begin{aligned} \mathrm{Cov}(X,Y) &= \mathbb{E}\left[(X-\mu_X)(Y-\mu_Y)\right] \\ &= \int p_{XY}(x,y)(x-\mu_X)(y-\mu_Y)dxdy, \end{aligned}$$

where $\mu_X := \mathbb{E}[X]$ and $\mu_Y := \mathbb{E}[Y]$ denote the mean of X and Y, respectively. The covariance describes the degree to which X and Y tend to deviate from their expected values in similar ways.

Note that the covariance $\mathrm{Cov}(X, X)$ of X with itself equals its variance, that is,

$$\mathrm{Cov}(X,X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \mathrm{Var}(X).$$

Theorem A.8 For two random variables (X, Y) on the same sample space, we have

$$\mathrm{Cov}(X, Y) = \mathbb{E}\left[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\right] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y].$$

Obviously, this result generalizes the property of variance in Theorem A.1.

Proof.

$$\begin{aligned}
\mathrm{Cov}(X, Y) &= \mathbb{E}\left[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\right] \\
&= \mathbb{E}\left[XY - X\mathbb{E}[Y] - \mathbb{E}[X]Y + \mathbb{E}[X]\mathbb{E}[Y]\right] \\
&= \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y] - \mathbb{E}[X]\mathbb{E}[Y] + \mathbb{E}[X]\mathbb{E}[Y] \\
&= \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y].
\end{aligned}$$

■

The correlation between random variable (X, Y) is a scaled variant of the covaiance:

$$\textbf{Correlation:} \qquad \mathrm{Corr}(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}},$$

where we scale the covariance with the standard deviation of X and Y. This ensures that the correlation is always in interval $[-1, 1]$.

Theorem A.9 For any two random variables (X, Y) on the same sample space, we have

$$-1 \leq \mathrm{Corr}(X, Y) \leq 1.$$

Proof. Define $\bar{X} = X - \mathbb{E}[X]$ and $\bar{Y} = Y - \mathbb{E}[Y]$. We have $\mathrm{Var}(X) = \mathbb{E}[\bar{X}^2]$, $\mathrm{Var}(Y) = \mathbb{E}[\bar{Y}^2]$ and $\mathrm{Cov}(X, Y) = \mathbb{E}[\bar{X}\bar{Y}]$. The result then follows the Cauchy–Schwarz inequality shown in Lemma A.10 below:

$$\mathrm{Cov}(X, Y) = \mathbb{E}[\bar{X}\bar{Y}] \leq \sqrt{\mathbb{E}[\bar{X}^2]\mathbb{E}[\bar{Y}^2]} = \sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}.$$

■

Lemma A.10 — Cauchy–Schwarz Inequality. For any two random variables (X, Y) on the same sample space, we have

$$\mathbb{E}\left[XY\right] \leq \sqrt{\mathbb{E}\left[X^2\right]\mathbb{E}\left[Y^2\right]}.$$

Proof. Obviously, we have $\mathbb{E}\left[(X - \alpha Y)^2\right] \geq 0$ for any number $\alpha \in \mathbb{R}$. Therefore,

$$\mathbb{E}\left[(X - \alpha Y)^2\right] = \mathbb{E}\left[X^2\right] - 2\alpha\mathbb{E}\left[XY\right] + \alpha^2\mathbb{E}\left[Y^2\right] \geq 0, \quad \alpha \in \mathbb{R}.$$

Since this holds for any number α, we can take $\alpha = \sqrt{\frac{\mathbb{E}[X^2]}{\mathbb{E}[Y^2]}}$ in particular. In this case,

$$\begin{aligned}
\mathbb{E}\left[X^2\right] - 2\alpha\mathbb{E}\left[XY\right] + \alpha^2\mathbb{E}\left[Y^2\right] &= \mathbb{E}\left[X^2\right] - 2\sqrt{\frac{\mathbb{E}\left[X^2\right]}{\mathbb{E}\left[Y^2\right]}}\mathbb{E}\left[XY\right] + \mathbb{E}\left[X^2\right] \\
&= 2\sqrt{\frac{\mathbb{E}\left[X^2\right]}{\mathbb{E}\left[Y^2\right]}}\left(\sqrt{\mathbb{E}\left[X^2\right]\mathbb{E}\left[Y^2\right]} - \mathbb{E}\left[XY\right]\right) \\
&\geq 0,
\end{aligned}$$

which is the result we want. ■

Theorem A.11 — Covariance of Sum of Two Random Variables. Let (X, Y) is a pair of random variables on the same sample space. We have

$$\mathrm{Var}(X+Y) = \mathrm{Var}(X) + 2\mathrm{Cov}(X,Y) + \mathrm{Var}(Y).$$

Proof. Define $\bar{X} = X - \mathbb{E}[X]$ and $\bar{Y} = Y - \mathbb{E}[Y]$. Then we have $\mathrm{Var}(X) = \mathbb{E}[\bar{X}^2]$, $\mathrm{Var}(Y) = \mathbb{E}[\bar{Y}^2]$ and $\mathrm{Cov}(X,Y) = \mathbb{E}[\bar{X}\bar{Y}]$. We have

$$\begin{aligned}
\mathrm{Var}(X+Y) &= \mathbb{E}[(\bar{X}+\bar{Y})^2] \\
&= \mathbb{E}[\bar{X}^2 + 2\bar{X}\bar{Y} + \bar{Y}^2] \\
&= \mathbb{E}[\bar{X}^2] + 2\mathbb{E}[\bar{X}\bar{Y}] + \mathbb{E}[\bar{Y}^2] \\
&= \mathrm{Var}(X) + 2\mathrm{Cov}(X,Y) + \mathrm{Var}(Y).
\end{aligned}$$

■

A.2 Gradient of Multivariate Functions

Gradient of Single Variable Functions

For a single variable function $f(a)$, it is gradient $\nabla f(a)$ is defined to be

$$\nabla f(a) = \lim_{\epsilon \to 0} \frac{f(a+\epsilon) - f(a)}{\epsilon}.$$

Alternatively, we can write this as

$$f(a+\epsilon) = f(a) + \nabla f(a)\epsilon + o(\epsilon),$$

where $o(\epsilon)$ denotes a term that is smaller in magnitude than ϵ, that is $R(\epsilon) = o(\epsilon)$ if $\lim_{\epsilon \to 0} R(\epsilon)/\epsilon = 0$. In fact, the formula above is the first order Taylor approximation, which approximates a function locally with a linear function (w.r.t. ϵ).

Related, the second order Taylor approximation locally approximates a function with a quadratic function:

$$f(a+\epsilon) = f(a) + \nabla f(a)\epsilon + \frac{1}{2}\nabla^2 f(a)\epsilon^2 + o(\epsilon^2),$$

where $\nabla^2 f(a) := \nabla(\nabla f(a))$ denotes the second order derivative of $f(a)$.

Gradient of Multivariate Functions

We now consider multivariate functions $f(\boldsymbol{a})$ which maps a vector-valued input $\boldsymbol{a} = [a_1, \ldots, a_d]^\top$ to a real value. In this case, the gradient $\nabla f(\boldsymbol{a})$ is a vector of the same size as $\boldsymbol{a}$, consisting of all the partial derivatives:

$$\nabla f(\boldsymbol{a}) = \begin{bmatrix} \dfrac{\partial f(\boldsymbol{a})}{\partial a_1} \\ \vdots \\ \dfrac{\partial f(\boldsymbol{a})}{\partial a_d} \end{bmatrix} \tag{A.1}$$

where $\frac{\partial f(\boldsymbol{a})}{\partial a_i}$ denotes the partial derivative w.r.t. a_i,

$$\frac{\partial f(\boldsymbol{a})}{\partial a_i} = \lim_{\epsilon \to 0} \frac{f([a_1, \ldots, a_i + \epsilon, \ldots a_d]^\top) - f([a_1, \ldots, a_i, \ldots, a_d]^\top)}{\epsilon}.$$

We can derive the partial derivative by assuming all the other variables are fixed to be constant.

Similar to the single-variable case, the gradient $\nabla f(\boldsymbol{a})$ also yields a first order Taylor approximation:

$$\begin{aligned} f(\boldsymbol{a}+\epsilon\boldsymbol{\delta}) &= f(\boldsymbol{a}) \ + \ \epsilon\nabla f(\boldsymbol{a})^\top\boldsymbol{\delta} \ + \ o(\epsilon), \\ &= f(\boldsymbol{a}) \ + \ \epsilon\sum_i \frac{\partial f(\boldsymbol{a})}{\partial a_i}\delta_i \ + \ o(\epsilon) \end{aligned} \tag{A.2}$$

where ϵ is a small number and $\boldsymbol{\delta} = [\delta_1, \ldots, \delta_d]^\top$ is any vector of the same size as $\boldsymbol{a}$. This suggests that the difference $f(\boldsymbol{a}+\epsilon\boldsymbol{\delta}) - f(\boldsymbol{a})$ approximately equals the sum of the partial derivatives $\frac{\partial f(\boldsymbol{a})}{\partial a_i}$ multiplying the perturbation $\epsilon\delta_i$ on the corresponding dimension (e.g., the inner product of $\nabla f(\boldsymbol{a})$ and $\epsilon\boldsymbol{\delta}$).

Deriving Gradient of Multivariate Functions (Optional)

Following the definition (A.1), we can derive the gradient of a multivariate function by calculating all the partial derivatives and put them together into a vector form. This, however, is cumbersome to do sometimes. Another approach is to leverage (A.2), deriving linear approximation of $f(\boldsymbol{a}+\epsilon\boldsymbol{\delta})$ and extract ∇f by comparing with the left hand side of (A.2).

Let us illustrate this method using the loss function of linear regression:

Theorem A.12 For the loss function of linear regression $f(\boldsymbol{a}) = \|X\boldsymbol{a} - Y\|^2$, we have

$$f(\boldsymbol{a}+\epsilon\boldsymbol{\delta}) = f(\boldsymbol{a}) \ + \ 2\epsilon(X\boldsymbol{a}-Y)^\top X\boldsymbol{\delta} \ + \ O(\epsilon^2).$$

Comparing this with (A.2), we have

$$\nabla f(\boldsymbol{a}) = 2((X\boldsymbol{a}-Y)^\top X)^\top = 2X^\top(X\boldsymbol{a}-Y).$$

Proof. We have

$$\begin{aligned} f(\boldsymbol{a}+\epsilon\boldsymbol{\delta}) &= \|X(\boldsymbol{a}+\epsilon\boldsymbol{\delta}) - Y\|^2 \\ &= \|X\boldsymbol{a} - Y + \epsilon X\boldsymbol{\delta}\|^2 \\ &= (X\boldsymbol{a} - Y + \epsilon X\boldsymbol{\delta})^\top(X\boldsymbol{a} - Y + \epsilon X\boldsymbol{\delta}) \\ &= (\boldsymbol{b} + \epsilon X\boldsymbol{\delta})^\top(\boldsymbol{b} + \epsilon X\boldsymbol{\delta}) \qquad \text{//define } \boldsymbol{b} = X\boldsymbol{a} - Y \text{ for notation} \\ &= \boldsymbol{b}^\top\boldsymbol{b} + \epsilon\left(\boldsymbol{b}^\top X\boldsymbol{\delta} + (X\boldsymbol{\delta})^\top\boldsymbol{b}\right) + \epsilon^2(X\boldsymbol{\delta})^\top(X\boldsymbol{\delta}) \qquad \text{//expand the quadratic} \\ &= \boldsymbol{b}^\top\boldsymbol{b} + 2\epsilon\boldsymbol{b}^\top X\boldsymbol{\delta} + \epsilon^2(X\boldsymbol{\delta})^\top(X\boldsymbol{\delta}) \qquad \text{//}\boldsymbol{b}^\top(X\boldsymbol{\delta}) = (X\boldsymbol{\delta})^\top\boldsymbol{b} \\ &= \boldsymbol{b}^\top\boldsymbol{b} + 2\epsilon\boldsymbol{b}^\top X\boldsymbol{\delta} + o(\epsilon) \qquad \text{//}o(\epsilon) \text{ represents the term with } \epsilon^2 \\ &= (X\boldsymbol{a} - Y)^\top(X\boldsymbol{a} - Y) + 2\epsilon(X\boldsymbol{a} - Y)^\top X\boldsymbol{\delta} + o(\epsilon) \\ &= f(\boldsymbol{a}) + 2\epsilon(X\boldsymbol{a} - Y)^\top X\boldsymbol{\delta}. \end{aligned}$$

■

Hessian Matrix (Optional)
Similarly, the second order Taylor approximation is

$$f(\boldsymbol{a}+\epsilon\boldsymbol{\delta}) = f(\boldsymbol{a}) \ + \ \epsilon\nabla f(\boldsymbol{a})^\top\boldsymbol{\delta} \ + \ \frac{\epsilon^2}{2}\boldsymbol{\delta}^\top H(\boldsymbol{a})\boldsymbol{\delta} \ + \ o(\epsilon^2)$$
$$= f(\boldsymbol{a}) \ + \ \epsilon\sum_i \frac{\partial f(\boldsymbol{a})}{\partial a_i}\delta_i \ + \ \epsilon^2\sum_{ij=1}^{d}\delta_i\frac{\partial^2 f(\boldsymbol{a})}{\partial a_i \partial a_j}\delta_j \ + \ o(\epsilon^2)$$

Here $H(\boldsymbol{a}) := \nabla(\nabla f(\boldsymbol{a})) := \left[\frac{\partial^2 f(\boldsymbol{a})}{\partial a_i \partial a_j}\right]_{ij=1}^{d}$ is the second order derivative (or Hessian matrix) of $f(a)$, which is a $d \times d$ matrix (because it is the vector derivative of a vector-valued function):

$$H(\boldsymbol{a}) = \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{a})}{\partial a_1 \partial a_1} & \cdots & \frac{\partial^2 f(\boldsymbol{a})}{\partial a_1 \partial_d} \\ \frac{\partial^2 f(\boldsymbol{a})}{\partial a_2 \partial a_1} & \ddots & \frac{\partial^2 f(\boldsymbol{a})}{\partial a_2 \partial a_d} \\ \frac{\partial^2 f(\boldsymbol{a})}{\partial a_d \partial a_1} & \cdots & \frac{\partial^2 f(\boldsymbol{a})}{\partial a_d \partial a_d}, \end{bmatrix}$$

where each $\frac{\partial^2 f(\boldsymbol{a})}{\partial a_i \partial a_j}$ is a second order partial derivative (or cross derivative),

A.3 Eigenvalues and EigenVectors

Eigenvectors of a square matrix is non-zero vectors that, when multiplied with the matrix, does not change direction. Specifically, a vector $\boldsymbol{x} \in \mathbb{R}^d$ is said to be an eigenvector of a square matrix $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ if there exists a scalar value λ (called an eigenvalue), such that

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}.$$

If $\boldsymbol{x}$ is an eigenvector, then $c\boldsymbol{x}$ is also an eigenvector for any non-zero scalar $c \in \mathbb{R}$. Therefore, we always assume scale $\boldsymbol{x}$ such that $\|\boldsymbol{x}\| = 1$. If $\boldsymbol{A}$ is a symmetric matrix, it must have a d pairs of eigenvectors/eigenvalues. We can sort the eigenvalues

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_d.$$

λ_1 is the minimum eigenvalue and λ_d the maximum eigenvalue.

Rayleigh quotient provides a good way to understand eigenvalues and eigenvectors. The Rayleigh quotient of $\boldsymbol{A}$ is a function defined

$$R_{\boldsymbol{A}}(\boldsymbol{x}) = \frac{\boldsymbol{x}^\top\boldsymbol{A}\boldsymbol{x}}{\|\boldsymbol{x}\|^2}.$$

Calculating its gradient yields

$$\nabla_{\boldsymbol{x}} R_{\boldsymbol{A}}(\boldsymbol{x}) = \frac{2}{\|\boldsymbol{x}\|^2}\left(\boldsymbol{A}\boldsymbol{x} - R_A(\boldsymbol{x})\boldsymbol{x}\right).$$

Note that $R_{\boldsymbol{A}}(\boldsymbol{x})$ is a scalar. Comparing this with the definition of eigenvector directly yields the following connection.

Theorem A.13 $\boldsymbol{x}$ is an eigenvalue of $\boldsymbol{A}$ iff it is a stationary point of the Rayleigh quotient (i.e., $\nabla_{\boldsymbol{x}} R_{\boldsymbol{A}}(\boldsymbol{x}) = 0$). In this case, $\lambda = R_{\boldsymbol{A}}(\boldsymbol{x})$ is its associated eigenvalue.

Eigenvectors of the maximum / minimum eigenvalues are hence related to the global maxima / minima of $R_{\boldsymbol{A}}(\boldsymbol{x})$. All the other eigenvectors are in fact saddle points of $R_{\boldsymbol{A}}(\boldsymbol{x})$.

Remark A.1 We can use (stochastic) gradient descent or other numerical optimization algorithms to find the maximum / minimum eigenvectors of a matrix. This is what people do in practice for very large scale matrices. ■