# Chapter 1

# Background

Welcome to ECE 595 Machine Learning I. Machine learning is an interesting subject. In order to prepare readers for our course, in this chapter we go over a few basic mathematical concepts that will be used frequently. Since this is a review rather than introduction, our discussion will be brief. Readers interested in details are encouraged to check out the references listed at the end of the chapter.

## 1.1 Linear Algebra

We denote $\boldsymbol{x} \in \mathbb{R}^n$ an $n$-dimensional vector taking real numbers as its entries. An $m$-by-$n$ matrix is denoted as $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. The matrix $\boldsymbol{A}$ can be viewed according to its columns and rows via either $\boldsymbol{A}$ or its transpose $\boldsymbol{A}^T$:

$$\boldsymbol{A} = \begin{bmatrix} | & | & & | \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & \dots & \boldsymbol{a}_m \\ | & | & & | \end{bmatrix}, \quad \text{and} \quad \boldsymbol{A}^T = \begin{bmatrix} — & \boldsymbol{a}_1^T & — \\ — & \boldsymbol{a}_2^T & — \\ & \vdots & \\ — & \boldsymbol{a}_m^T & — \end{bmatrix}.$$

Here, $\boldsymbol{a}_i$ is the $i$-th column of $\boldsymbol{A}$. The $(i,j)$-th element of $\boldsymbol{A}$ is denoted as $a_{ij}$ or $[\boldsymbol{A}]_{ij}$. The identity matrix is denote as $\boldsymbol{I}$. The $i$-th column of $\boldsymbol{I}$ is denoted as $\boldsymbol{e}_i = [0, \dots, 1, \dots, 0]^T$, and is called the $i$-th **standard basis vector**. An all-zero vector is denoted as $\boldsymbol{0} = [0, \dots, 0]^T$.

### Norm

The **norm** of $\boldsymbol{x}$ is denoted as $\|\boldsymbol{x}\|$, and it is the *length* of the vector $\boldsymbol{x}$. We are mostly interested in the $\ell_p$-norm of a vector, defined as

$$\|\boldsymbol{x}\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \tag{1.1}$$

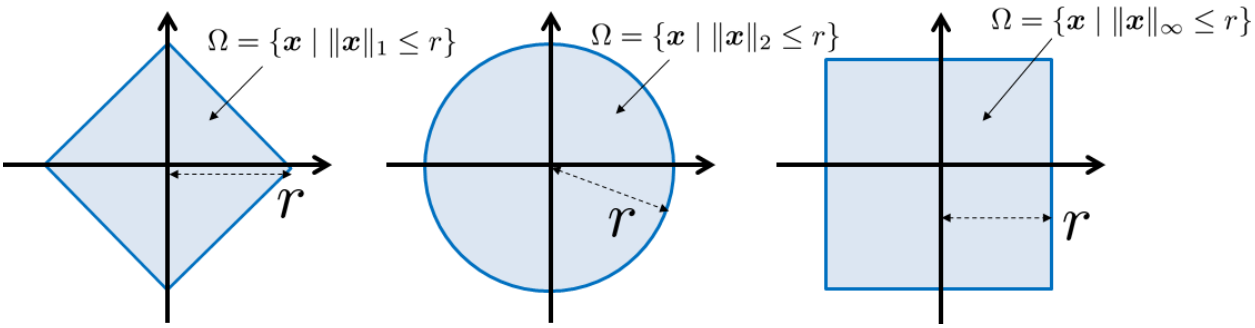


Figure 1.1: The shapes of $\Omega$ defined using different $\ell_p$-norms.

for $p \geq 1$. Let us take a look at a few of the most common choices of $p$.

**The $\ell_2$-norm**: When $p = 2$, $\|\boldsymbol{x}\|_2$ is called the **Euclidean norm** of $\boldsymbol{x}$, defined as

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}. \tag{1.2}$$

If we consider a set $\Omega = \{\boldsymbol{x} \mid \|\boldsymbol{x}\|_2 \leq r\}$, then all $\boldsymbol{x} \in \Omega$ are points inside the sphere with radius $r$. For example, if the dimension of $\boldsymbol{x}$ is $n = 2$, then $\Omega$ is equivalent to

$$\Omega = \{\boldsymbol{x} \mid \|\boldsymbol{x}\|_2 \leq r\} = \{(x_1, x_2) \mid x_1^2 + x_2^2 \leq r^2\},$$

which is a circle and all points $\boldsymbol{x} = (x_1, x_2)$ should live inside the circle. See Figure 1.1 for an illustration.

Note that $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_2$ is not the same as $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_2^2$. The former is the Euclidean norm, and hence the triangle inequality holds:

$$\|\boldsymbol{x} + \boldsymbol{y}\|_2 \leq \|\boldsymbol{x}\|_2 + \|\boldsymbol{y}\|_2.$$

In contrast, $\|\boldsymbol{x}\|^2$ is the Euclidean norm square and the triangle inequality does not hold. Another difference is that $\|\boldsymbol{x}\|_2$ not differentiable at $\boldsymbol{x} = \boldsymbol{0}$ whereas $\|\boldsymbol{x}\|^2$ is differentiable everywhere.

**The $\ell_1$-norm**: When $p = 1$, the norm becomes the **$\ell_1$-norm**, defined as

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|. \tag{1.3}$$

Different from the $\ell_2$-norm which is the sum of squares, the $\ell_1$-norm is the sum of absolute values. Geometrically, if we define as set $\Omega = \{\boldsymbol{x} \mid \|\boldsymbol{x}\|_1 \leq r\}$ then $\Omega$ is a diamond as shown

 

in Figure 1.1. Why is it a diamond? Consider the 2D case when $n = 2$. The condition $\|\boldsymbol{x}\|_1 = r$ is equivalent to

$$\|\boldsymbol{x}\|_1 = |x_1| + |x_2| = r.$$

If $x_1 > 0$ and $x_2 > 0$, then the absolute values have no effect to the sign and so the above equation reduces to $x_1 + x_2 = r$. This is a straight line connecting the coordinates $(r, 0)$ and $(0, r)$. If $x_1 < 0$ and $x_2 > 0$, then $|x_1| + |x_2| = r$ becomes a straight line in the 4th quadrant. Continuing this argument we will construct the diamond shape.

In MATLAB (or Python), you may determine the norm of a vector using

```
% MATLAB Code:
x = randn(100,1);
norm1_x = norm(x, 1); %l1 norm
norm2_x = norm(x, 2); %l2 norm
```

or equivalently by using the definition

```
% MATLAB Code:
x = randn(100,1);
norm1_x = sum(abs(x(:)));     %l1 norm
norm2_x = sqrt(sum(x(:).^2)); %l2 norm
```

**The $\ell_\infty$-norm**: When $p = \infty$, we define $\|\boldsymbol{x}\|_\infty$ as

$$\|\boldsymbol{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|. \tag{1.4}$$

Therefore, $\|\boldsymbol{x}\|_\infty$ picks the entry with the maximum magnitude. The set $\Omega = \{\boldsymbol{x} \mid \|\boldsymbol{x}\|_\infty \leq r\}$ is a square as shown in Figure 1.1. Readers are encouraged to check geometry in 2D.

It is not difficult to show that

$$\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1. \tag{1.5}$$

Without going through the formal proof, we can see why this is true from the figures. In Figure 1.1, the diamond is the smallest shape among the three, because its vertices touches the boundary of the circle and all its edges are inside the circle. Similarly, the circle is completely enclosed by the square.

The following inequality, called the Holder's Inequality, will become useful in this course.

**Theorem 1** (Holder's Inequality)**.** *Let $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^n$. Then,*

$$|\boldsymbol{x}^T \boldsymbol{y}| \leq \|\boldsymbol{x}\|_p \|\boldsymbol{y}\|_q \tag{1.6}$$

*for any $p$ and $q$ such that $\frac{1}{p} + \frac{1}{q} = 1$, where $p \geq 1$. Equality holds if and only if $|x_i|^p = \alpha |y_i|^q$ for some scalar $\alpha$ and for all $i = 1, \ldots, n$.*

 

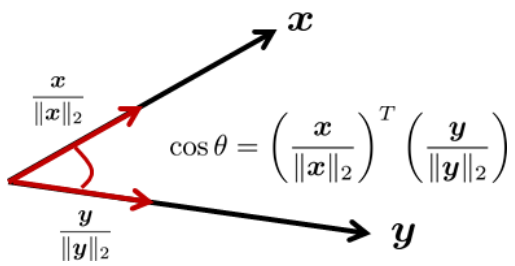


Figure 1.2: Pictorial interpretation of Cauchy-Schwarz inequality. The inner product defines the cosine angle, which by definition must be less than 1.

As a special case of the Holder's Inequality, we can substitute $p = q = 2$ to obtain the Cauchy-Schwarz inequality.

**Corollary 1** (Cauchy-Schwarz Inequality)**.** *Let $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^n$. Then,*

$$|\boldsymbol{x}^T\boldsymbol{y}| \leq \|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2, \tag{1.7}$$

*where the equality holds if and only if $\boldsymbol{x} = \alpha\boldsymbol{y}$ for some scalar $\alpha$.*

Intuitively, $\boldsymbol{x}^T\boldsymbol{y}/(\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2)$ defines the cosine angle between the two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Since cosine is always less than 1, we have $\boldsymbol{x}^T\boldsymbol{y}/(\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2) \leq 1$ and hence the Cauchy inequality. The equality holds if and only if the two vectors are parallel (positively or negatively because $\alpha$ can be any real scalar.) See Figure 1.2 for an illustration.

**Exercise 1.1**. Prove that the $\ell_1$-norm $\|\cdot\|_1$ and the $\ell_2$-norm $\|\cdot\|_2$ are equivalent. That is, show that there exists two real positive constants $c$ and $C$ such that

$$c\|\boldsymbol{v}\|_1 \leq \|\boldsymbol{v}\|_2 \leq C\|\boldsymbol{v}\|_1, \; \forall \boldsymbol{v} \in \mathbb{R}^n \tag{1.8}$$

**Hint**: Hölder's inequality can be useful at some point.

**Sparsity**. Among the different $\ell_p$ norms, the $\ell_1$-norm has a special property that it promotes sparsity. By sparse we meant that a vector $\boldsymbol{x}$ has most of its entries being 0 except a few. Sparsity plays an important role in modern statistics, as it allows us to handle a large set of variables with very few activations. For example, when modeling crime rate of a city, we may have hundreds of variables that could be the potential causes of the crime rate. Yet, if we dig into the data we may find that only a few of them are the true influential factors.

Why does the $\ell_1$-norm promotes sparsity? In many machine learning problems we need to solve optimization. For any convex function (we shall discuss what convex functions are later), minimizing over an $\ell_1$-ball will return one of the vertices. When touching a vertex, only that vertex will have a non-zero value and all the rest will be zero. Consequently, we have a sparse solution. Figure 1.3 shows an intuitive explanation and we shall discuss in details when we review convex optimization.

 

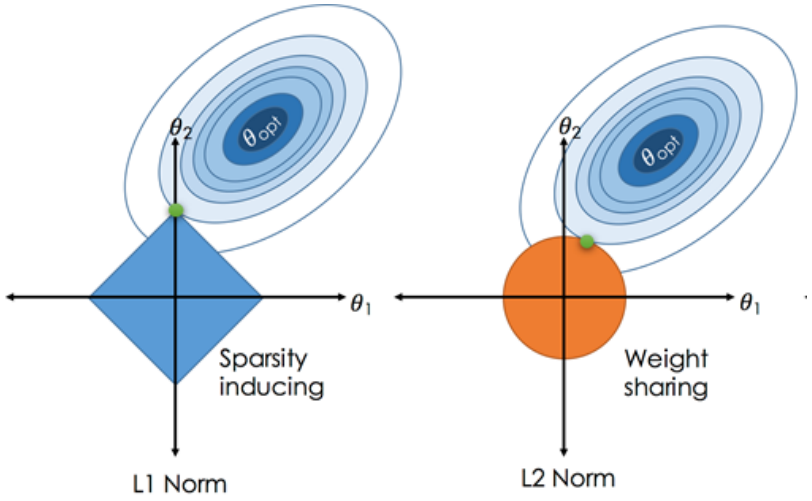


Figure 1.3: $\ell_1$-norm promotes sparsity whereas $\ell_2$-norm leads to weight sharing. Figure is taken from `http://www.ds100.org/`

## Eigenvalues and Eigenvectors

When applying a matrix $\boldsymbol{A}$ to a vector $\boldsymbol{x}$, what $\boldsymbol{x}$ would be invariant to $\boldsymbol{A}$? More precisely, for what $\boldsymbol{x}$ we can make sure that $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$, for some scalar $\lambda$? If we can find such vector $\boldsymbol{x}$, we say that $\boldsymbol{x}$ is the eigenvector of $\boldsymbol{A}$. Eigenvectors are useful for seeking principal components of datsets, or finding efficient signal representations. We shall start with the definition of eigenvectors.

**Definition 1.** *Given a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, the vector $\boldsymbol{u} \in \mathbb{R}^n$ (with $\boldsymbol{u} \neq \boldsymbol{0}$) is called the **eigenvector** of $\boldsymbol{A}$ if*

$$\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{u}, \tag{1.9}$$

*for some $\lambda \in \mathbb{R}$. The scalar $\lambda$ is called the **eigenvalue** associated with $\boldsymbol{u}$.*

An $n \times n$ matrix has $n$ eigenvectors and $n$ eigenvalues. Therefore, the above equation can be generalized to

$$\boldsymbol{A}\boldsymbol{u}_i = \lambda_i\boldsymbol{u}_i,$$

for $i = 1, \ldots, n$, or more compactly as $\boldsymbol{A}\boldsymbol{U} = \Lambda\boldsymbol{U}$. The eigenvalues $\lambda_1, \ldots, \lambda_n$ are not necessarily distinct. There are matrices with identical eigenvalues; The identity matrix is a trivial example. On the other hand, not all square matrices have eigenvectors. For example, the matrix $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ does not have an eigenvalue. Matrices that have eigenvalues must be **diagonalizable**.

There are a number of equivalent conditions for $\lambda$ to be an eigenvalue:

- There exists $\boldsymbol{u} \neq 0$ such that $\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{u}$;

 

- There exists $\boldsymbol{u} \neq 0$ such that $(\boldsymbol{A} - \lambda \boldsymbol{I})\boldsymbol{u} = \boldsymbol{0}$;
- $(\boldsymbol{A} - \lambda \boldsymbol{I})$ is not invertible;
- $\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$;

**Exercise 1.2**. Prove the equivalence of the above conditions.

In this course, we are mostly interested in symmetric matrices. If $\boldsymbol{A}$ is symmetric, then all the eigenvalues are real, and the following result holds.

**Theorem 2.** *If $\boldsymbol{A}$ is symmetric, then all the eigenvalues are real, and there exists $\boldsymbol{U}$ such that $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$:*

$$\underbrace{\begin{bmatrix} | & | & & | \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & \dots & \boldsymbol{a}_n \\ | & | & & | \end{bmatrix}}_{\boldsymbol{A}} = \underbrace{\begin{bmatrix} | & | & & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_n \\ | & | & & | \end{bmatrix}}_{\boldsymbol{U}} \underbrace{\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}}_{\boldsymbol{\Lambda}} \underbrace{\begin{bmatrix} — & \boldsymbol{u}_1^T & — \\ — & \boldsymbol{u}_2^T & — \\ & \vdots & \\ — & \boldsymbol{u}_n^T & — \end{bmatrix}}_{\boldsymbol{U}^T}. \tag{1.10}$$

We call such decomposition as the **eigen-decomposition**. For proof, please refer to, e.g., `https://cseweb.ucsd.edu/~dasgupta/291-unsup/lec7.pdf`.

In MATLAB / Python, we can compute the eigevalues of a matrix by using the `eig` command:

```
% MATLAB Code:
A = randn(100,100);
A = (A + A')/2;     % symmetrize because A is not symmetric
[U,S] = eig(A);     % eigen-decomposition
s = diag(S);        % extract eigen-value
```

The condition that $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ is the result of an **orthonormal** matrix. It is equivalent to say that $\boldsymbol{u}_i^T\boldsymbol{u}_j = 1$ if $i = j$ and $\boldsymbol{u}_i^T\boldsymbol{u}_j = 0$ if $i \neq j$. Since $\{\boldsymbol{u}_i\}_{i=1}^n$ is orthonormal, it can be served as a basis of any vector in $\mathbb{R}^n$:

$$\boldsymbol{x} = \sum_{j=1}^{n} \alpha_j \boldsymbol{u}_j, \tag{1.11}$$

where $\alpha_j = \boldsymbol{u}_j^T\boldsymbol{x}$ is called the **basis coefficient**. Basis vectors are useful in the sense that it can provide alternative **representation** of a vector.

 

Figure 1.4: The extended Yale Face Database B.

## Example: Eigenface

As a concrete example of how eigen-decompositions are used in practice, we consider a classical vision problem called the **eigenface** problem. Consider a dataset consisting of $N$ data points $\{\boldsymbol{x}_j\}_{j=1}^N$ with $\boldsymbol{x}_j \in \mathbb{R}^d$. Treat each data point as a vectorized version of a $\sqrt{d}\times\sqrt{d}$ image. We can estimate the covariance matrix of the data by computing

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{j=1}^{N}(\boldsymbol{x}_j - \boldsymbol{\mu})(\boldsymbol{x}_j - \boldsymbol{\mu})^T, \tag{1.12}$$

where $\boldsymbol{\mu} = \frac{1}{N}\sum_{j=1}^N \boldsymbol{x}_j$ is the mean vector. Note that the size of $\boldsymbol{\Sigma}$ is $d \times d$.

Once we obtain an estimate of the covariance matrix, we can perform an eigen-decomposition to get

$$\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T. \tag{1.13}$$

The columns of $\boldsymbol{U}$, i.e., $\{\boldsymbol{u}_i\}_{i=1}^d$ are the eigenvectors of $\boldsymbol{\Sigma}$. These eigenvectors are the **basis** of a testing face image.

Now, suppose that the dataset consists of two classes so that we have two corresponding covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, e.g., class 1 for male and class 2 for female. Since $\boldsymbol{\Sigma}_1$ is different from $\boldsymbol{\Sigma}_2$, we have two different eigen-decompositions $\boldsymbol{\Sigma}_1 = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ and $\boldsymbol{\Sigma}_2 = \boldsymbol{V}\boldsymbol{S}\boldsymbol{V}^T$. Denote the columns of $\boldsymbol{U}$ as $\{\boldsymbol{u}_i\}_{i=1}^d$ and columns of $\boldsymbol{V}$ as $\{\boldsymbol{v}_i\}_{i=1}^d$. Given a testing image $\boldsymbol{y}$, we can approximate $\boldsymbol{y}$ using the respective basis vectors to obtain the closest approximations using the leading $p$ vectors ($p < d$):

$$\widehat{\boldsymbol{y}} \approx \sum_{i=1}^{p}\alpha_i\boldsymbol{u}_i, \quad \text{and} \quad \widetilde{\boldsymbol{y}} \approx \sum_{i=1}^{p}\beta_i\boldsymbol{v}_i.$$

where $\alpha_i = \boldsymbol{u}_i^T\boldsymbol{y}$ and $\beta_i = \boldsymbol{v}_i^T\boldsymbol{y}$ are the basis coefficients for $\boldsymbol{U}$ and $\boldsymbol{V}$ respectively. If the approximation error (e.g., the sum squared error) $\|\boldsymbol{y} - \widehat{\boldsymbol{y}}\|^2$ is less than $\|\boldsymbol{y} - \widetilde{\boldsymbol{y}}\|^2$, then we claim that $\boldsymbol{y}$ belongs to $\boldsymbol{\Sigma}_1$; Otherwise we claim that $\boldsymbol{y}$ belongs to $\boldsymbol{\Sigma}_2$.

The above procedure of using the eigen-decomposition of the covariance is called the **principle component analysis** (PCA). PCA is a specific data analysis technique which applies eigen-decomposition to the covariance matrix. It is also a useful technique to reduce

 

the dimensionality of the input vector $\boldsymbol{y}$ by inspecting the first few projected coefficients $\boldsymbol{u}_i^T\boldsymbol{y}$. Also because of the projection, noise in the dataset can better be handled by PCA.

The MATLAB code below is taken from Wikipedia `https://en.wikipedia.org/wiki/Eigenface`, trying to extract the eigenfaces of the Extended Yale Database (`http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html`).

```
% MATLAB Code obtained from https://en.wikipedia.org/wiki/Eigenface
clear all; close all;
load yalefaces                              % load data
[h,w,n] = size(yalefaces); d = h*w;         % compute size
x = double(reshape(yalefaces,[d n]));       % vectorize images
mean_matrix = mean(x,2);                    %
x = bsxfun(@minus, x, mean_matrix);         % subtract mean
s = cov(x');                                % calculate covariance
[V,D] = eig(s);                             % obtain eigenvalue & eigenvector
eigval = diag(D);                           %
eigval = eigval(end:-1:1);                  % sort eigenvalues
V = fliplr(V);                              % sort eigenvectors
```

**Exercise 1.2**.

(a) Try the above code and comment on the visual appearance of the eigenfaces.

(b) Build a simple classifier by using the eigenface idea.

(c) Should we remove the mean of the testing image $\boldsymbol{y}$ when doing PCA? Why and why not?

## Positive Semi-Definite

Given a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, it is important to check the positive semi-definiteness of $\boldsymbol{A}$. Why? Here are two practical scenarios where we need positive semi-definiteness. (1) If we are estimating the covariance matrix $\boldsymbol{\Sigma}$ from a dataset, we need to ensure that $\boldsymbol{\Sigma} = \mathbb{E}[(\boldsymbol{X}-\boldsymbol{\mu})(\boldsymbol{X}-\boldsymbol{\mu})^T]$ is positive semi-definite because all covariance matrices are positive semi-definite. Otherwise the matrix we estimate is not a legitimate covariance matrix. (2) If we are solving an optimization problem involving a function $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$, then having $\boldsymbol{A}$ being positive semi-definite we can guarantee that the problem is convex. Convex problems ensure a local minimum is also global, and convex problems can be solved efficiently using known algorithms.

 

**Definition 2** (Positive Semi-Definite)**.** *A matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is positive semi-definite if*

$$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \geq 0 \tag{1.14}$$

*for any $\boldsymbol{x} \in \mathbb{R}^n$. $\boldsymbol{A}$ is positive definite if $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$ for any $\boldsymbol{x} \in \mathbb{R}^n$.*

Using eigen-decomposition, it is not difficult to show that positive semi-definiteness is equivalent to having non-negative eigenvalues.

**Theorem 3.** *A matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is positive semi-definite if and only if*

$$\lambda_i(\boldsymbol{A}) \geq 0 \tag{1.15}$$

*for all $i = 1, \ldots, n$, where $\lambda_i(\boldsymbol{A})$ denotes the $i$-th eigenvalue of $\boldsymbol{A}$.*

*Proof.* By definition of eigenvalue and eigenvector, we have that $\boldsymbol{A}\boldsymbol{u}_i = \lambda_i\boldsymbol{u}_i$ where $\lambda_i$ is the eigenvalue and $\boldsymbol{u}_i$ is the corresponding eigenvector. If $\boldsymbol{A}$ is positive semi-definite then $\boldsymbol{u}_i^T\boldsymbol{A}\boldsymbol{u}_i \geq 0$ since $\boldsymbol{u}_i$ is a particular vector in $\mathbb{R}^n$. So we have $0 \leq \boldsymbol{u}_i^T\boldsymbol{A}\boldsymbol{u}_i = \lambda\|\boldsymbol{u}_i\|^2$ and hence $\lambda_i \geq 0$. Conversely, if $\lambda_i \geq 0$ for all $i$, then since $\boldsymbol{A} = \sum_{i=1}^n \lambda_i\boldsymbol{u}_i\boldsymbol{u}_i^T$ we can conclude that $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^T\left(\sum_{i=1}^n \lambda_i\boldsymbol{u}_i\boldsymbol{u}_i^T\right)\boldsymbol{x} = \sum_{i=1}^n \lambda_i(\boldsymbol{u}_i^T\boldsymbol{x})^2 \geq 0$. □

The following corollary shows that if $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is positive definite, then it must be invertible. Being invertible also means that columns of $\boldsymbol{A}$ are linearly independent.

**Corollary 2.** *If a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is positive definite (not semi-definite), then $\boldsymbol{A}$ must be invertible, i.e., there exist $\boldsymbol{A}^{-1} \in \mathbb{R}^{n\times n}$ such that*

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}. \tag{1.16}$$

**Exercise 1.3**. Prove that for any $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, the matrix $\boldsymbol{A}^T\boldsymbol{A}$ is always positive semi-definite. You may prove using both the original definition or using the eigenvalue.

## Matrix Calculus

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a scalar field. The gradient of $f$ with respect to $\boldsymbol{x} \in \mathbb{R}^n$ is defined as

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix}. \tag{1.17}$$

 

To see how this works consider a few examples.

**Example 1**. $f(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x}$. In this case, the gradient is

$$\nabla_{\boldsymbol{x}}\Big(\boldsymbol{a}^T\boldsymbol{x}\Big) = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1}\sum_{j=1}^n a_j x_j \\ \vdots \\ \frac{\partial}{\partial x_n}\sum_{j=1}^n a_j x_j \end{bmatrix} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} = \boldsymbol{a}. \tag{1.18}$$

**Example 2**. $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$. Then,

$$\begin{aligned}\nabla_{\boldsymbol{x}}\Big(\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}\Big) = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix} &= \begin{bmatrix} \frac{\partial}{\partial x_1}\sum_{i,j=1}^n a_{ij}x_i x_j \\ \vdots \\ \frac{\partial}{\partial x_n}\sum_{i,j=1}^n a_{ij}x_i x_j \end{bmatrix} \\ &= \begin{bmatrix} \sum_{j=1}^n a_{1,j}x_j \\ \vdots \\ \sum_{j=1}^n a_{n,j}x_j \end{bmatrix} + \begin{bmatrix} \sum_{i=1}^n a_{i,1}x_i \\ \vdots \\ \sum_{i=1}^n a_{i,n}x_i \end{bmatrix} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{A}^T\boldsymbol{x}. \end{aligned} \tag{1.19}$$

If $\boldsymbol{A}$ is symmetric so that $\boldsymbol{A} = \boldsymbol{A}^T$ then $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 2\boldsymbol{A}\boldsymbol{x}$.

**Example 3**. $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$. The gradient is

$$\begin{aligned}\nabla_{\boldsymbol{x}}\Big(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2\Big) &= \nabla_{\boldsymbol{x}}\Big(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{y}^T\boldsymbol{y}\Big) \\ &= \nabla_{\boldsymbol{x}}\Big(\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}\Big) - 2\nabla_{\boldsymbol{x}}\Big(\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{x}\Big) + \nabla_{\boldsymbol{x}}\Big(\boldsymbol{y}^T\boldsymbol{y}\Big) \\ &= 2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{A}^T\boldsymbol{y} + 0 = 2\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}). \end{aligned} \tag{1.20}$$

The Hessian of $f$ with respect to $\boldsymbol{x} \in \mathbb{R}^n$ is defined as

$$\nabla^2_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n^2} \end{bmatrix}. \tag{1.21}$$

A quick way of obtaining the Hessian is compute $\nabla_{\boldsymbol{x}}\Big(\nabla_{\boldsymbol{x}} f\Big)$, where the outer gradient is

 

applied to each element of $(\nabla_{\boldsymbol{x}} f)_i$. Let us consider the following example.

**Example 4.** $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$. The Hessian is

$$\begin{aligned}\nabla^2_{\boldsymbol{x}}\Big(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2\Big) &= \nabla_{\boldsymbol{x}}\Big(2\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})\Big) \\ &= \Big[\nabla_{\boldsymbol{x}}\Big(\boldsymbol{c}_1^T\boldsymbol{x} - b_1\Big) \quad \nabla_{\boldsymbol{x}}\Big(\boldsymbol{c}_2^T\boldsymbol{x} - b_2\Big) \quad \dots \quad \nabla_{\boldsymbol{x}}\Big(\boldsymbol{c}_n^T\boldsymbol{x} - b_n\Big)\Big], \end{aligned} \tag{1.22}$$

where $\boldsymbol{c}_i$ is the $i$-th column of $2\boldsymbol{A}^T\boldsymbol{A}$ such that $2\boldsymbol{A}^T\boldsymbol{A} = [\boldsymbol{c}_1, \dots, \boldsymbol{c}_n]$, and $b_i$ is the $i$-th element of $2\boldsymbol{A}^T\boldsymbol{y}$. Continuing the calculation, we show that

$$\nabla^2_{\boldsymbol{x}}\Big(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2\Big) = \begin{bmatrix}\boldsymbol{c}_1 & \boldsymbol{c}_2 & \dots & \boldsymbol{c}_n\end{bmatrix} = 2\boldsymbol{A}^T\boldsymbol{A}. \tag{1.23}$$

# 1.2 Probability

## Basic Concept

A probability space contains three basic elements: (1) Sample space $\Omega$, (2) Event Space $\mathcal{E}$, and (3) Probability Law $\mathbb{P}$, as depicted in Figure 1.5. Given an experiment, the **sample space** $\Omega$ contains all the possible outcomes of the experiment. For example, if we throw a dice then the sample space is $\{1, 2, 3, 4, 5, 6\}$. An event $E$ is a collection of outcomes. For example, if we throw a dice, an event could be $E = \{1, 2\}$ or $E = \emptyset$. The collection all possible events is the **event space** $\mathcal{E}$. If $\Omega$ is finite, then $\mathcal{E}$ is also the power set of $\Omega$. Every event in the event space can be measured by a function called the **probability law** $\mathbb{P}$. The probability law is always applied to an event. Thus, $\mathbb{P}[E]$ is the probability that an event $E$ happens, which is also the size of the set if we consider $E$ as a set. For example, when throwing a dice $\mathbb{P}[\{1, 2\}]$ should be larger than $\mathbb{P}[\{1\}]$ because the former is a bigger set. All probability laws should satisfy the three **Axioms of Probability**. Readers interested in details can refer to an undergraduate probability textbook.

A random variable $X$ is a mapping from $\mathcal{E}$ to $\mathbb{R}$. Such mapping is necessary because events in $\mathcal{E}$ are not always numbers, e.g., democrats and republicans, male and female, etc. A random variable can be considered as an encoding scheme that converts a verbal description of an event to a number on the real line. The exact mapping is not important, e.g., encoding male = 1 and female = 0 is the probabilistically the same as encoding male = 0 and female = 1. As long as we clearly state the mapping and keep consistent, there will not be any issue. We denote a random variable as $X$, and the state it takes as $x$.

Each random variable has an underlying distribution $F_X(x)$ called the **cumulative distribution function** (CDF), defined as

$$F_X(x) = \mathbb{P}[X \leq x]. \tag{1.24}$$

 

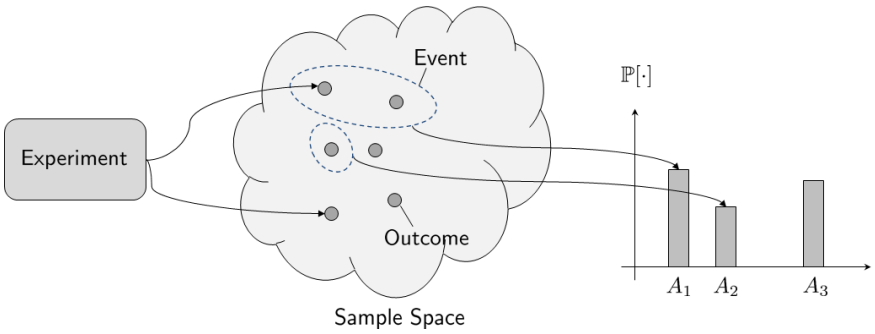


Figure 1.5: A probability space is composed of three elements: (1) Sample space $\Omega$, (2) Event Space $\mathcal{E}$, and (3) Probability Law $\mathbb{P}$.

Here, $\mathbb{P}[X \leq x]$ is the measure of the probability of the event $\{X \leq x\}$. Since CDF measures the probability of $\{X \leq x\}$, the cumulative probability grows as $x$ increases. So $F_X$ is a **non-decreasing** function. When $x \to \infty$, $F_X(\infty) = 1$, and when $x \to -\infty$, $F_X(-\infty) = 0$. The derivative of the CDF is called the **probability density function** (PDF):

$$p_X(x) = \frac{d}{dx} F_X(x). \tag{1.25}$$

PDF can be considered as the **histogram** of the random variable $X$. Therefore, the height of $p_X(x)$ can be (roughly) treated as probability of having a state $x$.

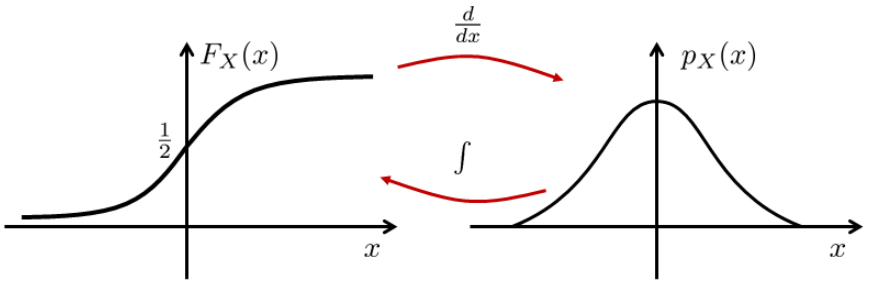


Figure 1.6: The relationship between CDF and PDF; Valid for continuous CDF only.

**Example 1**. Let $X$ be an exponential random variable with CDF

$$F_X(x) = 1 - e^{-\lambda x},$$

for $x \geq 0$ and $\lambda > 0$. The PDF is given by

$$p_X(x) = \frac{d}{dx}\left(1 - e^{-\lambda x}\right) = \lambda e^{-\lambda x}, \quad x \geq 0. \tag{1.26}$$

Note that $p_X(x)$ exists only when $F_X$ is continuously differentiable at $x$. If the CDF is not continuous differentiable, e.g., it is a step function, then the associated PDF becomes a

 

probability mass, e.g., a delta function.

**Example 2**. Let $X$ be a random variable with

$$p_X(x) = \begin{cases} 0, & x < 0, \\ \frac{1}{2}, & x = 0, \\ \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{x^2}{2}\right\}, & x > 0. \end{cases}$$

Then, the CDF of $X$ is

$$F_X(x) = \begin{cases} 0, & x < 0, \\ \frac{1}{2} + \int_0^x \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{s^2}{2}\right\} ds, & x \geq 0. \end{cases} \tag{1.27}$$

The **expectation** of $X$ is defined as

$$\mathbb{E}[X] = \int x \, dF_X(x) = \int x \, p_X(x) dx, \tag{1.28}$$

where the second equality is valid only when $p_X$ exists. Not all random variables has an expectation. Expectation $\mathbb{E}[X]$ exists only when $\mathbb{E}[|X|] < \infty$. For example, a Cauchy distribution does not have an expectation.

For any function $g$, the expectation $\mathbb{E}[g(X)]$ is defined as

$$\mathbb{E}[g(X)] = \int g(x) \, dF_X(x) = \int g(x) \, p_X(x) dx. \tag{1.29}$$

For example, if $g(x) = x^2$ then $\mathbb{E}[g(X)] = \mathbb{E}[X^2]$ is the **second moment** of $X$. The **variance** of $X$ is defined as $\text{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$.

**Exercise 2.1**. Prove that the expectation and the variance of a Gaussian random variable is $\mathbb{E}[X] = \mu$ and $\text{Var}[X] = \sigma^2$. The PDF of $X$ is given by

$$p_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}. \tag{1.30}$$

Hint: When proving the mean, use the fact that $p_X(x)$ is an even function and $xp_X(x)$ is an odd function.

Generating random variables in computer is typically done by built-in pseudo-number generators. Here is an example of generating Gaussian random variables.

```
% MATLAB Code: Generate random numbers
n = 1000;
```

 

```
x = randn(n,1);        % generate n random numbers from Gaussian(0,1)
y = 0.5*randn(n,1)+2; % generate n random numbers from Gaussian(2,0.5)
muX  = mean(x);        % find mean
muY  = mean(Y);
varX = var(x);         % find variance
varY = var(Y);
hist(x,30);            % plot the histogram of x using 30 bins
```

If we let $g(X) = e^{sX}$, then $\mathbb{E}[g(X)]$ is defined as the **moment generating function**:

$$M_X(s) = \mathbb{E}[e^{sX}]. \tag{1.31}$$

For example, the moment generating function of a Gaussian is $M_X(s) = \exp\{\mu s + \frac{s^2\sigma^2}{2}\}$. From this we can show that if $X \sim \mathcal{N}(\mu, \sigma^2)$, then any affine transformation of $X$ remains a Gaussian random variable: Let $Y = aX + b$ for some constants $a$ and $b$. The moment generating function of $Y$ is

$$\begin{aligned} M_Y(s) &= \mathbb{E}[e^{sY}] = \mathbb{E}[e^{s(aX+b)}] = e^{sb}\mathbb{E}[e^{asX}] \\ &= \exp\left\{sb + \mu(as) + \frac{(as)^2\sigma^2}{2}\right\} \\ &= \exp\left\{(a\mu + b)s + \frac{s^2(a\sigma)^2}{2}\right\}. \end{aligned}$$

Therefore, $Y$ remains a Gaussian, and its mean and variance are $a\mu+b$ and $a^2\sigma^2$, respectively. This results is summarized in the theorem below.

**Theorem 4** (Affine Transformation of Gaussian)**.** *Let $X \sim \mathcal{N}(\mu, \sigma^2)$ be a Gaussian random variable with mean $\mu$ and variance $\sigma^2$. If $Y = aX + b$ for some constants $a$ and $b$, then $Y$ is also a Gaussian, with $s$*

$$\mathbb{E}[Y] = a\mu + b, \quad \textit{and} \quad \mathrm{Var}[Y] = a^2\sigma^2. \tag{1.32}$$

For additional formula of moment generating functions, readers can refer to `https://engineering.purdue.edu/ChanGroup/ECE302/`.

In general $\mathbb{E}[g(X)] \neq g(\mathbb{E}[X])$. However, if $g$ is convex, we can prove Jensen's Inequality.

**Theorem 5** (Jensen's Inequality)**.** *Let $X$ be a random variable and $g$ be a convex function. Then*

$$g(\mathbb{E}[X]) \leq \mathbb{E}[g(X)] \tag{1.33}$$

 

**Exercise 2.2**. Prove Jensen's Inequality for a finite discrete random variable $X$.
Assume that $X$ is a discrete random variable with $n$ states $\{x_1, \ldots, x_n\}$. The probability of getting these states are $\{p_1, \ldots, p_n\}$, with $\sum_{i=1}^n p_i = 1$.

(a) Prove that if $n = 2$, $g(\mathbb{E}[X]) \leq \mathbb{E}[g(X)]$. Hint: If $n = 2$, $\mathbb{E}[X] = p_1 x_1 + p_2 x_2$.

(b) Assume that $g\left(\sum_{i=1}^n p_i x_i\right) \leq \sum_{i=1}^n p_i g(x_i)$. Prove that

$$g\left(\sum_{i=1}^{n+1} p_i x_i\right) \leq \sum_{i=1}^{n+1} p_i g(x_i).$$

Hence by induction conclude that $g(\mathbb{E}[X]) \leq \mathbb{E}[g(X)]$.

Beside Jensen's Inequality, the other very useful inequality is the Markov inequality.

**Theorem 6.** *For any $X > 0$ and $\epsilon > 0$,*

$$\mathbb{P}[X \geq \epsilon] \leq \frac{\mathbb{E}[X]}{\epsilon}. \tag{1.34}$$

*Proof.* Consider $\epsilon\mathbb{P}[X \geq \epsilon]$. It holds that $\epsilon\mathbb{P}[X \geq \epsilon] = \int_\epsilon^\infty \epsilon\, p_X(x)dx \leq \int_\epsilon^\infty x p_X(x)dx$, where the inequality holds because for any $x \geq \epsilon$, the integrand which is non-negative will always increase. It then follows that $\int_\epsilon^\infty x p_X(x)dx \leq \int_0^\infty x p_X(x)dx = \mathbb{E}[X]$. □

**Exercise 2.3**. Use Markov inequality to prove the following two inequalities.

(a) Chebyshev inequality. Let $X$ be any random variable and let $\epsilon > 0$, then

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq \epsilon] \leq \frac{\text{Var}[X]}{\epsilon^2}. \tag{1.35}$$

(b) Chernoff inequality. Let $s > 0$, then

$$\mathbb{P}[X \geq \epsilon] \leq e^{-s\epsilon \log M_X(s)}, \tag{1.36}$$

where $M_X(s) \stackrel{\text{def}}{=} \mathbb{E}[e^{sX}]$ is the moment generating function of $X$.

## Joint Distributions

Let $X$ and $Y$ be two random variables. The joint CDF of $X$ and $Y$ is defined as

$$F_{X,Y}(x, y) = \mathbb{P}[X \leq x, Y \leq y]. \tag{1.37}$$

 

We can generalize this idea to $N$ random variables $X_1, \ldots, X_N$. For notational simplicity we denote $\boldsymbol{X} = [X_1, \ldots, X_N]^T$, and write $F_{\boldsymbol{X}}(\boldsymbol{x}) = F_{X_1,\ldots,X_N}(x_1, \ldots, x_N)$. The joint PDF of $X$ and $Y$ is defined as

$$p_{X,Y}(x,y) = \frac{\partial}{\partial x \partial y} F_{X,Y}(x,y). \tag{1.38}$$

For $N$ random variables, the joint PDF is denoted as $p_{\boldsymbol{X}}(\boldsymbol{x})$. The **joint expectation** of $X$ and $Y$ is the **correlation**

$$\mathbb{E}[XY] = \int xy \, dF_{X,Y}(x,y) = \int xy \, p_{X,Y}(x,y) dx dy.$$

The **covariance** of $X$ and $Y$ is $\text{Cov}(X,Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$.

**Exercise 2.4**. Let $X \sim \text{Uniform}[0,1]$ be a uniform random variable on $[0,1]$. Let $Y = X^2$.

(a) Find $\text{Cov}(2X, Y)$.

(b) Find the CDF of $Z = 2X + Y$.

Two random variables $X$ and $Y$ are said to be **independent** if $F_{X,Y}(x,y) = F_X(x)F_Y(y)$, which implies that $p_{X,Y}(x,y) = p_X(x)p_Y(y)$. The density functions $p_X(x)$ and $p_Y(y)$ are called the marginal density functions. A sequence of random variables $X_1, \ldots, X_N$ is said to be independently and identically distributed (**i.i.d.**) if they are independent and all have the same distribution. That is, $F_{X_1,\ldots,X_N}(x, \ldots, x) = (F_{X_1}(x))^N$.

**Exercise 2.5**.

(a) Show that if $X$ and $Y$ are independent, then $\text{Cov}(X,Y) = 0$.

(b) Show that the converse is not true by constructing a counter example.

The **conditional CDF** of $X$ given $Y$ is denoted as

$$F_{X|Y}(x|y) = \frac{F_{X,Y}(x,y)}{F_Y(y)}, \tag{1.39}$$

and similarly the condition PDF is $p_{X|Y}(x|y) = \frac{p_{X,Y}(x,y)}{p_Y(y)}$. If $X$ and $Y$ are independent, then $p_{X|Y}(x|y) = p_X(x)$. The **Bayes Theorem** states that

$$p_{Y|X}(y|x) = \frac{p_{X|Y}(x|y)p_Y(y)}{p_X(x)}, \tag{1.40}$$

which allows us to switch the roles between $Y|X$ and $X|Y$. If $X$ represents certain observation and $Y$ represents an underlying model parameter, then $p_{Y|X}$ is called the **posterior**

 

density, and $p_{X|Y}$ is called the **likelihood**. The distribution $p_Y(y)$ is called the **prior**. By **Law of Total Probability**, we can further write the Bayes Theorem as

$$p_{Y|X}(y|x) = \frac{p_{X|Y}(x|y)p_Y(y)}{\int p_{X|Y}(x|y)p_Y(y)dy}, \tag{1.41}$$

where the denominator holds because $p_X(x) = \int p_{X|Y}(x|y)p_Y(y)dy$. Since $p_X(x)$ is independent of the model parameter $Y$, it is called the **partition function**, and can be removed in some of the machine learning problems (e.g., in some linear classifiers).

The **conditional expectation** $\mathbb{E}[X|Y=y]$ is defined as

$$\mathbb{E}[X|Y=y] = \int x\, dF_{X|Y}(x|y) = \int x p_{X|Y}(x|y)dx, \tag{1.42}$$

and it holds that $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X|Y]]$, due to the **law of total expectation**. To see this, note that

$$\begin{aligned}
\mathbb{E}[X] &= \int x p_X(x)dx = \int x \left( \int p_{X,Y}(x,y)dy \right) dx \\
&= \int \int x p_{X|Y}(x|y)p_Y(y)dydx \\
&= \int \left( \int x p_{X|Y}(x|y)dx \right) p_Y(y)dy = \int \mathbb{E}[X|Y=y]p_Y(y)dy = \mathbb{E}[\mathbb{E}[X|Y]].
\end{aligned}$$

**Exercise 2.6**. Let $X \sim \mathcal{N}(\mu, \sigma)$ and let $Y|X \sim \mathcal{N}(X, X/10)$.

(a) Find $\mathbb{E}[Y|X=x]$ and $\mathbb{E}[Y^2|X=x]$.

(b) Find $\mathbb{E}[Y]$ and $\text{Var}[Y]$.

What can conditional expectation be used for? Suppose that we have a pair of random variables $(X, Y)$ and there is some unknown process which maps $X$ to $Y$. Let us assume that we have observed $Y$. Can we design a function $g$ such that the error between the unknown variable $X$ and the predicted variable $g(Y)$ is minimized:

$$\min_g \ \mathbb{E}[(X - g(Y))^2].$$

The optimal $g$ is called the **minimum mean squared error** (MMSE) estimator. It turns out that the MMSE estimator is the condition expectation of $X$ given $Y$.

**Theorem 7.** *The MMSE estimator is a function $g^*$ which minimizes the mean squared error:*

$$g^* = \underset{g}{\text{argmin}} \ \mathbb{E}[(X - g(Y))^2],$$

*and is given by*

$$g^*(y) = \mathbb{E}[X|Y=y]. \tag{1.43}$$

 

## Law of Large Number

The **sample mean** or the **empirical average** of a sequence of random variables $X_1, \ldots, X_N$ is defined as

$$M_N = \frac{1}{N}\sum_{i=1}^{N} X_i. \tag{1.44}$$

Since all $X_i$'s are random variables, the average $M_N$ is also a random variable. As a result, $M_N$ has its own CDF and PDF. If $X_i$ are i.i.d. with mean $\mathbb{E}[X_i] = \mu$, then the mean of $M_N$ is

$$\mathbb{E}[M_N] = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} X_i\right] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[X_i] = \mu. \tag{1.45}$$

For finite number of samples $N$, the deviation between the sample mean and the true mean is bounded according to a probability inequality. For example, by Chebyshev inequality we have

$$\mathbb{P}[|M_N - \mu| > \epsilon] \leq \frac{\sigma^2}{N\epsilon^2}, \tag{1.46}$$

where $\sigma^2 = \text{Var}[X_i]$. Therefore, as $N \to \infty$, $M_N \xrightarrow{p} \mu$. This result is known as the **weak law of large number**.

**Theorem 8** (Weak Law of Large Number)**.** *Let $X_1, \ldots, X_N$ be a sequence of i.i.d. random variables with mean $\mathbb{E}[X_i] = \mu$ and variance $\text{Var}[X_i] = \sigma^2$. Then,*

$$\lim_{N\to\infty}\mathbb{P}\left[\left|\frac{1}{N}\sum_{i=1}^{N} X_i - \mu\right| > \epsilon\right] = 0. \tag{1.47}$$

Law of Large Number plays a critical role in understanding the theoretical limit of a machine learning algorithm. It will help us ask questions such as: What is the minimum of samples we need to train an algorithm? How well will training? How well can we generalize?

**Example 3**. Let $X_i$ be a Bernoulli random variable with $p = 1/2$. Define

$$M_N = \frac{1}{N}\sum_{i=1}^{N} X_i.$$

By varying $N$, we like to plot $M_N$ as a function of $N$ and see if $M_N \to p$. The MATLAB code of this example is shown below, with the plot shown in Figure 1.7. As we can see from the plot, the average of $M_N$ is converging to the true value $p$. The variance of $M_N$ is also reducing as $N$ grows.

 

```
% MATLAB code to demonstrate weak law of large number
prob_true = 1/2;
prob_est  = zeros(1,100);
prob_std  = zeros(1,100);
Nset = round(logspace(2,5,100));
for n = 1:100
    N = Nset(n);
    out = (rand(N,1)>0.5);
    prob_est(n) = mean(out);
    prob_std(n) = std(out);
end
semilogx(Nset, prob_est, 'o', 'LineWidth', 2); hold on;
semilogx(Nset, prob_true*ones(size(Nset)), 'k-', 'LineWidth', 2);
semilogx(Nset, prob_true+prob_std./sqrt(Nset), 'r', 'LineWidth', 2);
semilogx(Nset, prob_true-prob_std./sqrt(Nset), 'r', 'LineWidth', 2);
xlabel('number of trials');
ylabel('probability');
```

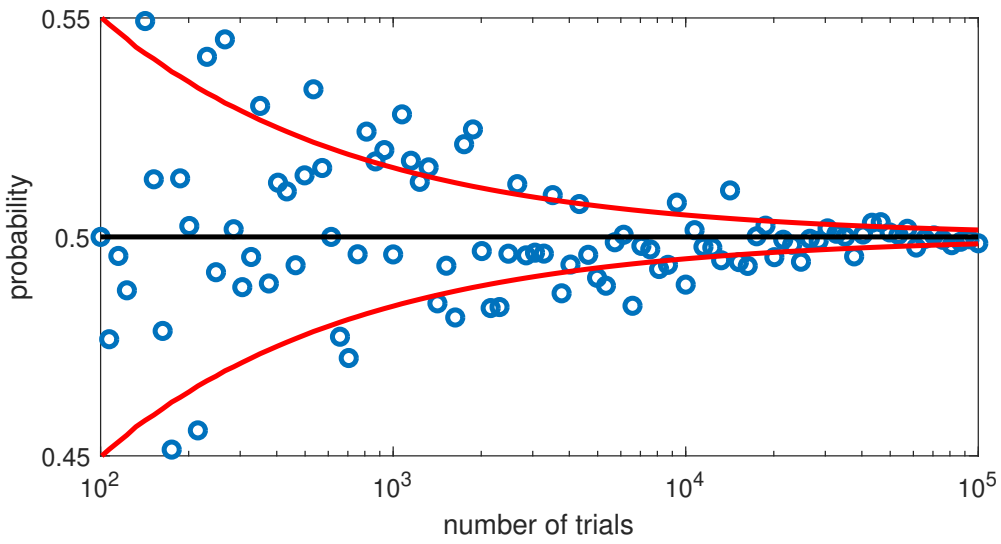


Figure 1.7: Weak Law of Large Number: As $N$ grows, the empirical average $M_N$ converges to the true probability $p$.

## Multi-dimensional Gaussian

Among the many different distributions, the most commonly used distribution in this course is the joint Gaussian. A joint Gaussian distribution concerns about a vector of random

 

variables $\boldsymbol{X} = [X_1, \ldots, X_d]^T$, where we call $d$ the dimensionality of the random vector $\boldsymbol{X}$. Entries of the random vector $\boldsymbol{X}$ are not necessarily independent (in fact, we like them to be correlated so that we can model complex structures).

The expectation of $\boldsymbol{X}$ is

$$\mathbb{E}[\boldsymbol{X}] = \begin{bmatrix} \mathbb{E}[X_1] \\ \vdots \\ \mathbb{E}[X_d] \end{bmatrix} = \boldsymbol{\mu}, \tag{1.48}$$

and the covariance matrix is

$$\mathbb{E}[(\boldsymbol{X} - \boldsymbol{\mu})(\boldsymbol{X} - \boldsymbol{\mu})^T] = \begin{bmatrix} \mathrm{Var}[X_1] & \mathrm{Cov}(X_1, X_2) & \ldots & \mathrm{Cov}(X_1, X_d) \\ \mathrm{Cov}(X_2, X_1) & \mathrm{Var}[X_2] & \ldots & \mathrm{Cov}(X_2, X_d) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(X_d, X_1) & \mathrm{Cov}(X_N, X_2) & \ldots & \mathrm{Var}[X_d] \end{bmatrix} = \boldsymbol{\Sigma}. \tag{1.49}$$

It is not difficult to show that $\boldsymbol{\Sigma}$ is symmetric positive semi-definite. (Why?)

On computer, computing the mean vector and the covariance matrix can be done as follows. Consider a data matrix $\boldsymbol{X} \in \mathbb{R}^{d \times N}$, where $d$ is the dimensionality of the feature, and $N$ is the number of samples. The sample mean vector and the sample covariance matrix are respectively:

$$\widehat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{j=1}^{N} \boldsymbol{X}_j \quad \text{and} \quad \widehat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{j=1}^{N} (\boldsymbol{X}_j - \boldsymbol{\mu})(\boldsymbol{X}_j - \boldsymbol{\mu})^T, \tag{1.50}$$

and the code to generate them is shown below.

```
% MATLAB code to compute mean and covariance matrix.
d = 64; N = 5000;
X     = randn(d,N);
mu    = mean(X,2);
Sigma = cov(X');
```

Let us define the PDF of a $d$-dimensional Gaussian.

**Definition 3.** *An $d$-dimensional **Gaussian** has a PDF*

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left\{ -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right\}, \tag{1.51}$$

*where $d$ denotes the dimensionality of the vector $\boldsymbol{x}$.*

**Special Case: Diagonal Covariance**. The $d$-dimensional Gaussian is a generalization of the 1D Gaussian(s). Suppose that $X_i$ and $X_j$ are independent for all $i \neq j$. Then,

 

$\mathbb{E}[X_iX_j] = \mathbb{E}[X_i]\mathbb{E}[X_j]$ and hence $\text{Cov}(X_i, X_j) = 0$. Consequently, the covariance matrix $\boldsymbol{\Sigma}$ is a diagonal matrix:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_d^2 \end{bmatrix},$$

where $\sigma_i^2 = \text{Var}[X_i]$. When this happens, exponential in the Gaussian PDF is

$$\begin{aligned}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) &= \begin{bmatrix} x_1-\mu_1 \\ \vdots \\ x_n-\mu_n \end{bmatrix}^T \begin{bmatrix} \sigma_1^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_d^2 \end{bmatrix}^{-1} \begin{bmatrix} x_1-\mu_1 \\ \vdots \\ x_n-\mu_d \end{bmatrix} \\ &= \begin{bmatrix} x_1-\mu_1 \\ \vdots \\ x_n-\mu_d \end{bmatrix}^T \begin{bmatrix} \frac{x_1-\mu_1}{\sigma_1^2} \\ \vdots \\ \frac{x_n-\mu_n}{\sigma_d^2} \end{bmatrix} = \sum_{i=1}^n \frac{(x_i-\mu_i)^2}{\sigma_i^2}.\end{aligned}$$

Moreover, the determinant $|\boldsymbol{\Sigma}|$ is

$$|\boldsymbol{\Sigma}| = \left| \begin{bmatrix} \sigma_1^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_d^2 \end{bmatrix} \right| = \prod_{i=1}^d \sigma_i^2.$$

Substituting these results into the joint Gaussian PDF we obtain

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = \prod_{i=1}^n \frac{1}{\sqrt{(2\pi)\sigma_i^2}} \exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\}, \tag{1.52}$$

which is a product of individual Gaussians.

The **negative log-likelihood** of a joint Gaussian is defined as

$$\begin{aligned}-\log p_{\boldsymbol{X}}(\boldsymbol{x}) &= -\log\left(\frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}\exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}\right) \\ &= -\log\left(\frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}\right) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \\ &= \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) - \frac{n}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{\Sigma}|. \end{aligned} \tag{1.53}$$

The first term $\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$ is a quadratic term. Since $\boldsymbol{\Sigma}$ is positive semi-definite, it holds that

$$\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \geq 0,$$

 

for any $\boldsymbol{x}$. The square root $\sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$ is called the **Mahalanobis distance**, which is a measure of how far away $\boldsymbol{x}$ is from $\boldsymbol{\mu}$.

Generating random numbers from a multi-dimension Gaussian can be done by calling built-in commands. In MATLAB and Python, we can use the following code.

```
% MATLAB code: Generate random numbers from multivariate Gaussian
mu    = [0 0];
Sigma = [.25 .3; .3 1];
x     = mvnrnd(mu,Sigma,1000);
```

To display the data points and overlay with the contour, we can call commands such as `contour`. The resulting plot looks like the one shown in Figure 1.8.

```
% MATLAB code: Overlay random numbers with the Gaussian contour.
x1 = -2.5:.01:2.5;
x2 = -3.5:.01:3.5;
[X1,X2] = meshgrid(x1,x2);
F = mvnpdf([X1(:) X2(:)],mu,Sigma);
F = reshape(F,length(x2),length(x1));
figure(1);
scatter(x(:,1),x(:,2),'rx', 'LineWidth', 1.5); hold on;
contour(x1,x2,F,[.001 .01 .05:.1:.95 .99 .999], 'LineWidth', 2);
xlabel('x'); ylabel('y');
set(gcf, 'Position', [100, 100, 600, 300]);
```

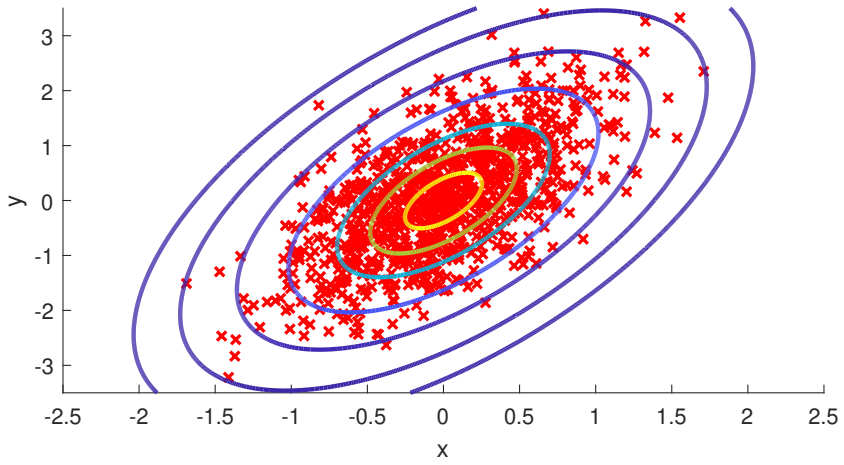


Figure 1.8: Example of generating 1000 random numbers from a 2D Gaussian and plotting the contour.

**Exercise 2.7**. Show that the marginal distribution of a joint Gaussian remains a Gaussian. Verify this result by modifying the code above.

## Geometry of Multi-dimensional Gaussian

The geometry of the joint Gaussian is determined by its eigenvalues and eigenvectors. Consider the eigen-decomposition of $\boldsymbol{\Sigma}$:

$$\begin{aligned}\boldsymbol{\Sigma} &= \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \\ &= \begin{bmatrix} | & | & & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \lambda_n \end{bmatrix} \begin{bmatrix} - & \boldsymbol{u}_1^T & - \\ - & \boldsymbol{u}_2^T & - \\ & \vdots & \\ - & \boldsymbol{u}_n^T & - \end{bmatrix}. \end{aligned} \tag{1.54}$$

for some unitary matrix $\boldsymbol{U}$ and diagonal matrix $\boldsymbol{\Lambda}$. The columns of $\boldsymbol{U}$ are called the eigenvectors, and the entries of $\boldsymbol{\Lambda}$ are called the eigenvalues. Since $\boldsymbol{\Sigma}$ is symmetric, all $\lambda_i$'s are real. In addition, since $\boldsymbol{\Sigma}$ is positive semi-definite, all $\lambda_i$'s are non-negative. As such, the volume defined by the multi-dimensional Gaussian is always a convex object, e.g., an ellipse in 2D or an ellipsoid in 3D.

**Exercise 2.8**. For what $\boldsymbol{U}$ and $\boldsymbol{\Lambda}$ can we ensure that the covariance matrix $\boldsymbol{\Sigma}$ is a diagonal matrix with diagonal elements $[\boldsymbol{\Sigma}]_{ii} = \sigma_i^2$?

The orientation of the axes are defined by the column vectors $\boldsymbol{u}_i$. In case of $n = 2$, the major axis is defined by $\boldsymbol{u}_1$ and the minor axis is defined by $\boldsymbol{u}_2$. The corresponding radii of each axis is specified by the eigenvalues $\lambda_1$ and $\lambda_2$. Figure 1.9 shows an illustration.

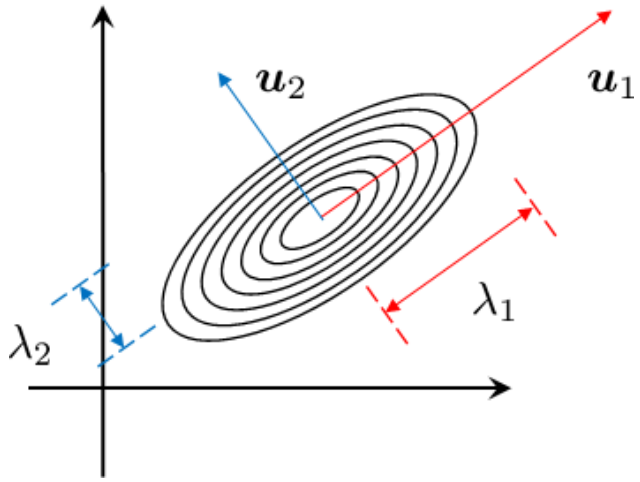


Figure 1.9: The center and the radius of the ellipse is determined by $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

One typical problem we encounter in simulation is how to generate random samples according to some Gaussian distributions. If we are given zero-mean unit-variance Gaussian

 

$\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, it is possible to generate $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for a known $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. To do so, we let $\boldsymbol{Y} = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{X} + \boldsymbol{\mu}$, where $\boldsymbol{\Sigma}^{\frac{1}{2}} = \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{U}^T$. Then, the mean of $\boldsymbol{Y}$ is

$$\mathbb{E}[\boldsymbol{Y}] = \mathbb{E}[\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{X} + \boldsymbol{\mu}] = \boldsymbol{\Sigma}^{\frac{1}{2}}\mathbb{E}[\boldsymbol{X}] + \boldsymbol{\mu} = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{0} + \boldsymbol{\mu} = \boldsymbol{\mu}.$$

and the covariance matrix is

$$\begin{aligned}\mathbb{E}[(\boldsymbol{Y}-\boldsymbol{\mu})(\boldsymbol{Y}-\boldsymbol{\mu})^T] &= \mathbb{E}[(\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{X} + \boldsymbol{\mu} - \boldsymbol{\mu})(\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{X} + \boldsymbol{\mu} - \boldsymbol{\mu})^T] \\ &= \mathbb{E}[(\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{X})(\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{X})^T] = \boldsymbol{\Sigma}^{\frac{1}{2}}\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T]\boldsymbol{\Sigma}^{\frac{1}{2}} \\ &= \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{I}\boldsymbol{\Sigma}^{\frac{1}{2}} = \boldsymbol{\Sigma}.\end{aligned}$$

**Theorem 9.** *Let $\boldsymbol{X}$ be a zero-mean unit-variance Gaussian $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Consider a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$ with eigen-decomposition $\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$. If*

$$\boldsymbol{Y} = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{X} + \boldsymbol{\mu}, \tag{1.55}$$

*where $\boldsymbol{\Sigma}^{\frac{1}{2}} = \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{U}^T$, then $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.*

**Exercise 3.8**. Show the reverse direction of the previous theorem. That is, given $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, find an affine transformation $(\boldsymbol{A}, \boldsymbol{b})$ such that the random vector $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{Y} + \boldsymbol{b}$ is a zero-mean unit-variance Gaussian: $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The relationship is shown in Figure 1.10.

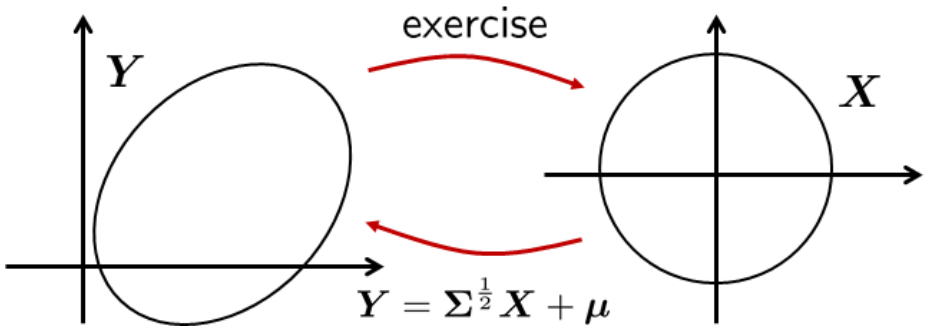


Figure 1.10: Given a Gaussian $\boldsymbol{Y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, there exists an affine transformation that can convert $\boldsymbol{Y}$ to a standard Gaussian $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

## 1.3 Optimization

### Unconstrained Optimization

Consider an unconstrained optimization over a subset $\mathcal{X} \subseteq \mathbb{R}^n$:

$$\underset{\boldsymbol{x}\in\mathcal{X}}{\text{minimize}} \quad f(\boldsymbol{x}), \tag{1.56}$$

 

where $f : \mathbb{R}^n \to \mathbb{R}$ is a scalar function. We say the a point $\boldsymbol{x}^* \in \mathcal{X}$ is a **global minimizer** if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathcal{X}$, and a **local minimizer** if $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})$ for any $\boldsymbol{x}$ in a neighborhood $\mathcal{B}_\delta(\boldsymbol{x}^*) = \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}^*\|_2 \leq \delta\}$. Being a global minimizer only means that $f(\boldsymbol{x}^*)$ is unique; It does not mean that $\boldsymbol{x}^*$ is unique. To say that a global minimizer is unique, we need to be state it explicitly that it is a unique global minimizer.

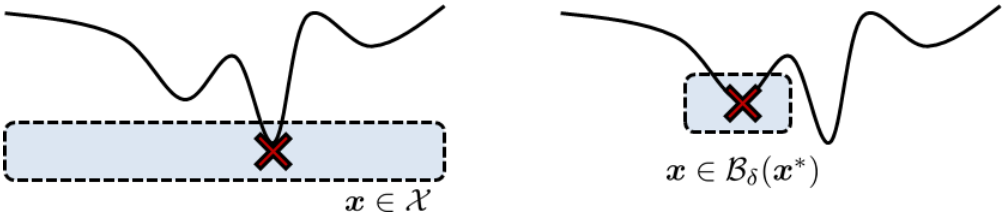


Figure 1.11: [Left] A global minimizer $\boldsymbol{x}^*$ attains the minimum value throughout the entire $\mathcal{X}$. [Right] A local minimizer attains minimum only in a local neighborhood.

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, it would be useful if we can claim local (or global) optimality using the gradient information of $f$.

**Theorem 10.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. Assume that $\nabla f(\boldsymbol{x})$ and $\nabla^2 f(\boldsymbol{x})$ exist for all $\boldsymbol{x} \in \mathcal{B}_\delta(\boldsymbol{x}^*)$ (or $\boldsymbol{x} \in \mathbb{R}^n$). Then, $\boldsymbol{x}^*$ is a local (or global) minimizer if and only if*

*(i) First Order Condition: $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$.*

*(ii) Second Order Condition: $\nabla^2 f(\boldsymbol{x}^*) \succeq 0$. ("$\succeq$" for necessity, and "$\succ$" for sufficiency.)*

Here is a hand-waving argument of why the above first order and second order conditions are true. Suppose $\boldsymbol{x}^*$ is a local (or global) minimizer, then by the definition of optimal point, we should expect to get $f(\boldsymbol{x}^* + \epsilon\boldsymbol{d}) \geq f(\boldsymbol{x}^*)$, for any $\boldsymbol{d}$ and $\epsilon > 0$ such that $\boldsymbol{x}^* + \epsilon\boldsymbol{d} \in \mathcal{B}_\delta(\boldsymbol{x}^*)$. The mean value theorem suggests that

$$f(\boldsymbol{x}^* + \epsilon\boldsymbol{d}) = f(\boldsymbol{x}^*) + \epsilon\nabla f(\boldsymbol{x}^*)^T\boldsymbol{d} + \frac{\epsilon^2}{2}\boldsymbol{d}^T\nabla^2 f(\widehat{\boldsymbol{x}})\boldsymbol{d}, \tag{1.57}$$

for some $\widehat{\boldsymbol{x}} \in \mathcal{B}_\delta(\boldsymbol{x}^*)$. Moving terms around and taking the limit of $\epsilon \to 0$, we can obtain the first order condition:

$$0 \leq \lim_{\epsilon\to 0}\frac{1}{\epsilon}\Big[f(\boldsymbol{x}^* + \epsilon\boldsymbol{d}) - f(\boldsymbol{x}^*)\Big] = \nabla f(\boldsymbol{x}^*)^T\boldsymbol{d} + \lim_{\epsilon\to 0}\left[\frac{\epsilon}{2}\boldsymbol{d}^T\nabla^2 f(\widehat{\boldsymbol{x}})\boldsymbol{d}\right], \tag{1.58}$$

Since $\boldsymbol{d}$ is arbitrary and the inequality has to hold for all $\boldsymbol{d}$, the only possibility is that "$\leq$" is replaced by "=". Thus, $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$. The second order condition is obtained by a similar manner:

$$0 \leq \lim_{\epsilon\to 0}\frac{1}{\epsilon^2}\Big[f(\boldsymbol{x}^* + \epsilon\boldsymbol{d}) - f(\boldsymbol{x}^*)\Big] = \underbrace{\nabla f(\boldsymbol{x}^*)^T\boldsymbol{d}}_{=0} + \frac{1}{2}\boldsymbol{d}^T\nabla^2 f(\boldsymbol{x}^*)\boldsymbol{d} + \lim_{\epsilon\to 0}\left[\frac{\epsilon}{6}\mathcal{O}(\|\boldsymbol{d}\|^3)\right], \tag{1.59}$$

 

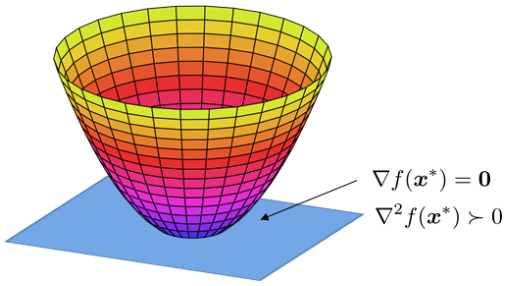


Figure 1.12: Pictorial illustration of the first and second order optimality. Figure credit to `https://en.wikipedia.org/`

which implies that $\nabla^2 f(\boldsymbol{x}^*) \succeq 0$.

**Example 1**. Let us consider an example. Suppose $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$. The first order condition implies that

$$\mathbf{0} = \nabla f(\boldsymbol{x}^*) = \nabla\Big(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2\Big) = 2\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{A}^T\boldsymbol{y}.$$

Rearranging the terms yields

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{y},$$

which is known as the **normal equation**. The solution is given by $\boldsymbol{x}^* = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y}$, assuming that $\boldsymbol{A}^T\boldsymbol{A}$ is invertible.

If the matrix $\boldsymbol{A}^T\boldsymbol{A}$ in the normal equation is not invertible, one standard trick is to append a regularization constant $\lambda$ so that the solution becomes $\boldsymbol{x}^* = (\boldsymbol{A}^T\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}\boldsymbol{A}^T\boldsymbol{y}$. The magnitude of $\lambda$ should be carefully chosen, for too large will make $\boldsymbol{x}^* \to \mathbf{0}$, and too small will not alleviate the non-invertible issue of $\boldsymbol{A}^T\boldsymbol{A}$.

In MATLAB / Python, solving the normal equation can be done by either the followings.

```
% MATLAB Code
n = 100;
A = randn(n,n);
y = randn(n,1);
x1 = inv(A'*A)*(A'*y);
x2 = A \ y;
```

Here, the first equation is the normal equation, and in case $\boldsymbol{A}^T\boldsymbol{A}$ is not invertible we can replace the inverse "`inv`" by pseudo-inverse "`pinv`" (which only inverts the non-zero eigenvalues.) The second equation using backslash "\" directly solves a system of linear equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y},$$

 

which is an over-determined system if $\boldsymbol{A}$ has more rows than columns.

**Exercise 3.1**.

(i) What is the objective function $f(\boldsymbol{x})$ for $\boldsymbol{x}^* = (\boldsymbol{A}^T\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}\boldsymbol{A}^T\boldsymbol{y}$?

(ii) The normal equation is trying to solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$. Is it meaningful to append multiple $\boldsymbol{A}^T\boldsymbol{A}$, e.g. $(\boldsymbol{A}^T\boldsymbol{A})^2\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{A}^T\boldsymbol{y}$?

One handy way to solve unconstrained optimization in MATLAB / Python is to use built-in packages. Of particular interest to our course is the convex optimization package `cvx` by Boyd et al. Here is an example trying to minimize $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$.

```
% MATLAB Code. Please install CVX package from http://cvxr.com/cvx/
n = 100;
A = randn(2*n,n);
y = randn(2*n,1);
cvx_begin
   variable x(n)
   minimize( norm( A*x-y ) )
cvx_end
```

**Example 2**. $f(\boldsymbol{x}) = \log\left(\sum_{i=1}^m \exp(\boldsymbol{a}_i^T\boldsymbol{x} + b_i)\right)$. The first order condition implies that

$$\boldsymbol{0} = \nabla f(\boldsymbol{x}^*) = \frac{1}{\sum_{j=1}^m \exp(\boldsymbol{a}_j^T\boldsymbol{x}^* + b_j)} \sum_{i=1}^m \exp(\boldsymbol{a}_i^T\boldsymbol{x}^* + b_i)\boldsymbol{a}_i.$$

Unfortunately this system of nonlinear equation does not have a closed-form solution, and so we need to resort to iterative algorithms to solve the problem.

As example in MATLAB / Python, we will minimize $f(\boldsymbol{x}) = \log\left(\sum_{i=1}^m \exp(\boldsymbol{a}_i^T\boldsymbol{x} + b_i)\right) + \lambda\|\boldsymbol{x}\|^2$, where $\lambda$ is a parameter.

```
% MATLAB Code. Please install CVX package from http://cvxr.com/cvx/
n = 100;
A = randn(n,n);
b = randn(n,1);
lambda = 0.1;
cvx_begin
```

 

```
   variable x(n)
   minimize( log_sum_exp( A*x + b  ) + lambda * sum_square(x) )
cvx_end
```

## Convexity

**Definition 4** (Convex function). *Let $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{X}$. Let $0 \leq \lambda \leq 1$. A function $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** over $\mathcal{X}$ if*

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}). \tag{1.60}$$

*The function is called strictly convex if "$\leq$" is replaced by "$<$".*

Geometrically, we can consider $\boldsymbol{x}$ and $\boldsymbol{y}$ as two end points of a set $\mathcal{X}$. The scalar parameter $\lambda$ measures the swing from $\boldsymbol{y}$ to $\boldsymbol{x}$ as it goes from 0 to 1. When $\lambda = 0$, the combination is $\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y} = \boldsymbol{y}$; When $\lambda = 1$, the combination is $\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y} = \boldsymbol{x}$. For anything in between, $\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y} = \boldsymbol{y}$ represents an intermediate point from $\boldsymbol{y}$ to $\boldsymbol{x}$. Therefore, $f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y})$ is the function evaluated at this intermediate point. The right hand side of the definition is $\lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$, which is the linear combination of the output values. Pictorially $\lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$ is the line connecting the two end points $f(\boldsymbol{x})$ and $f(\boldsymbol{y})$. If $f$ is convex, then this straight line must be higher than the actual functional value evaluated at the intermediate point. Figure 1.13 illustrates the idea.

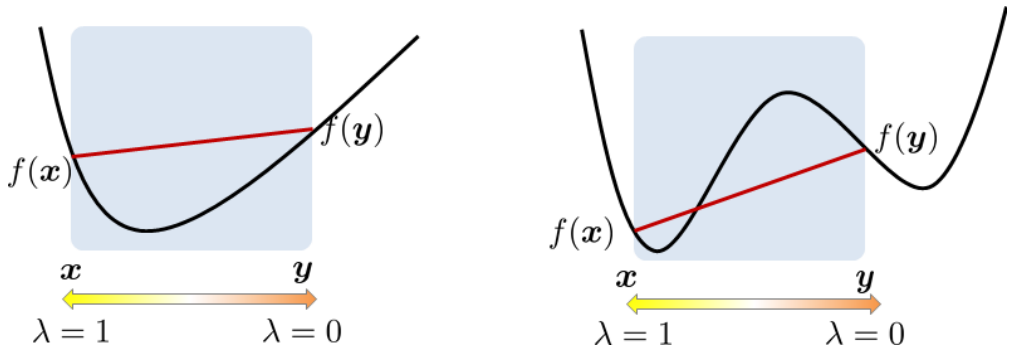


Figure 1.13: [Left] A convex function has a line above the function. [Right] A non-convex function we can find a pair of points $(\boldsymbol{x}, \boldsymbol{y})$ such that the line is not always above the curve.

 

**Theorem 11.** *Let $\mathcal{X} \in \mathbb{R}^n$ be a convex subset and let $f : \mathbb{R}^n \to \mathbb{R}$.*

*(i) First Order Convexity: $f$ is convex over $\mathcal{X}$ if and only if*

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}), \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}. \tag{1.61}$$

*(ii) Second Order Convexity: $f$ is convex over $\mathcal{X}$ if and only if*

$$\nabla^2 f(\boldsymbol{x}) \succeq 0 \quad \forall \boldsymbol{x} \in \mathcal{X}.$$

The proof of these two results can be found in Bertsekas-Nedic-Ozdaglar Proposition 1.2.5 and 1.2.6, or Boyd-Vandenberghe Section 3.1.3 and 3.1.4.

**Exercise 3.2**. Let $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$. Show that $f$ is convex by using three different approaches:

(i) $f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y})$ for all $\boldsymbol{x}$ and $\boldsymbol{y} \in \mathbb{R}^n$, and $0 \leq \lambda \leq 1$.

(ii) $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})$ for all $\boldsymbol{x}$ and $\boldsymbol{y} \in \mathbb{R}^n$.

(iii) $\nabla^2 f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{R}^n$.

If a function is convex, then any local minimizer is also a global minimizer. If the function is strictly convex, then the global minimizer is **unique**. Convexity of $f$ is a very powerful condition, for it allows us to use any local algorithm to find a global minimizer.

**Exercise 3.3**. Let $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_1$.

(i) Show that $f$ is convex but not straightly convex.

(ii) Show that the minimizer of $f$ is not unique.

## Gradient Descent

One of the commonly used algorithms in this course is the gradient descent algorithm. The reason of studying gradient descent, despite the availability of off-the-shelf convex optimization packages such as `CVX`, is that gradient descent is transparent. We can make exact claims about whether the algorithm is converging, how fast it converges, and where it is converging to. This is also part of the reasons why many deep learning packages use gradient descent as their core optimization engine. But most importantly, the algorithm is just simple.

 

Gradient descent is an iterative algorithm. Given an initial estimate $\boldsymbol{x}^{(0)}$, the algorithm defines the $t$-th iteration using the definition below.

**Definition 5** (Gradient Descent)**.** *The gradient descent algorithm is an iterative algorithm, with the $t$-th iteration being*

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \alpha^{(t)} \nabla f(\boldsymbol{x}^{(t)}), \qquad t = 0, 1, 2, \ldots, \tag{1.62}$$

*where $\alpha^{(t)}$ is called the step size.*

Gradient descent works because at any point $\boldsymbol{x}^{(t)}$, the negative gradient $-\nabla f(\boldsymbol{x}^{(t)})$ is guaranteed to point to a direction where the objective value will decrease. Perhaps less obvious, but Equation (1.58) actually suggests that

$$\nabla f(\boldsymbol{x}^*)^T \boldsymbol{d} \geq 0, \tag{1.63}$$

for any $\boldsymbol{d} \in \mathbb{R}^n$. This implies that at the optimal point $\boldsymbol{x}^*$, and potential search direction $\boldsymbol{d}$ must make $\nabla f(\boldsymbol{x}^*)^T \boldsymbol{d}$ positive. However, if we are not at the optimal point yet, say we are at $\boldsymbol{x}^{(t)}$, then we want reduction in cost so that $f(\boldsymbol{x}^{(t)} + \epsilon \boldsymbol{d}) \leq f(\boldsymbol{x}^{(t)})$. Thus, we will have

$$\underbrace{\lim_{\epsilon \to 0} \frac{1}{\epsilon} \Big[ f(\boldsymbol{x}^{(t)} + \epsilon \boldsymbol{d}) - f(\boldsymbol{x}^{(t)}) \Big]}_{\leq 0} = \nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d}$$

Therefore, if we are not at the optimal point yet, then $\nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d} \leq 0$.

So what $\boldsymbol{d}$ we should pick? Apparently we should simply pick a $\boldsymbol{d}$ that can make $\nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d}$ as small as possible. One possible option is to consider

$$\boldsymbol{d}^{(t)} = \operatorname*{argmin}_{\boldsymbol{d}} \quad \nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d}.$$

However, this problem is not valid because the objective function is unbounded below. In order to constrain $\boldsymbol{d}$ so that $\boldsymbol{d}$ cannot be arbitrarily large, we restrict ourselves to $\boldsymbol{d}$'s such that $\|\boldsymbol{d}\|_2 = \delta$ for some constant $\delta$. This gives us the following problem.

**Theorem 12.** *The gradient descent search direction $\boldsymbol{d}$ is the solution of the problem*

$$\boldsymbol{d}^{(t)} = \operatorname*{argmin}_{\|\boldsymbol{d}\|_2 = \delta} \quad \nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d}, \tag{1.64}$$

*in which solution is given by $\boldsymbol{d}^{(t)} = -\nabla f(\boldsymbol{x}^{(t)})$.*

*Proof.* To prove this result, we first notice that the optimization is equivalent to

$$\boldsymbol{d}^{(t)} = \operatorname*{argmin}_{\boldsymbol{d} \neq \boldsymbol{0}} \quad \frac{\delta \, \nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d}}{\|\boldsymbol{d}\|_2}, \tag{1.65}$$

 

because at optimal point we must have $\|\boldsymbol{d}\|_2 = \delta$. Now, by Cauchy inequality, we have that

$$\nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d} \geq -\|\nabla f(\boldsymbol{x}^{(t)})\|_2 \|\boldsymbol{d}\|_2,$$

with equality holds if and only if $\boldsymbol{d} = -\nabla f(\boldsymbol{x}^{(t)})$. Therefore, the minimum of Equation (1.65) is attainable when $\boldsymbol{d} = -\nabla f(\boldsymbol{x}^{(t)})$, and hence $\boldsymbol{d} = -\nabla f(\boldsymbol{x}^{(t)})$ is the optimal solution. □

Geometrically, $-\nabla f(\boldsymbol{x}^{(t)})$ is the direction perpendicular to the tangent line of the contour at $\boldsymbol{x}^{(t)}$. If the surface is convex, then $-\nabla f(\boldsymbol{x}^{(t)})$ is guaranteed to point to a place where the objective value is lower. And since $-\nabla f(\boldsymbol{x}^{(t)})$ is the solution to Equation (1.64), it is the steepest descent direction. As we continue to iterate, the negative gradient will eventually point towards the minimum location which is the ultimate goal. Pictorially, Figure 1.14 explains the ideas of gradient descent.

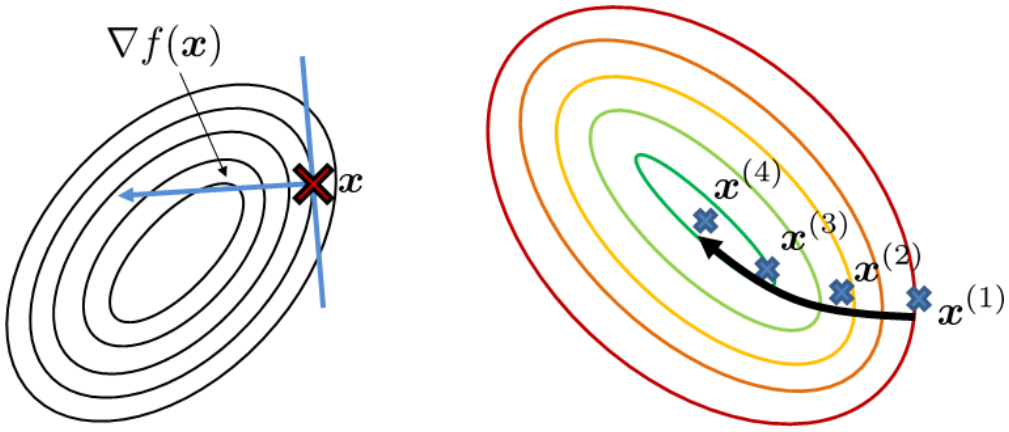


Figure 1.14: [Left] Gradient descent picks the orthogonal direction to the tangent of the contour at $\boldsymbol{x}^{(t)}$. [Right] Iterates of the gradient descent algorithm.

Let us look at an example. Suppose we like to minimize the function $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2$. Then, the gradient descent written in MATLAB or Python is as follows.

```
% MATLAB code: Gradient Descent for min f(x) = ||Ax-y||^2
% Assume we have x0, A, y and alpha.
x = x0;
alpha = 0.01;
for i=1:max_itr
    df = A'*(A*x - y);
    x  = x - alpha * df;
end
```

Essentially, as long as we have a way to compute the gradient $\nabla f(\boldsymbol{x}^{(t)})$, the gradient descent algorithm can be executed. In case where $\nabla f(\boldsymbol{x}^{(t)})$ is difficult to derive, e.g., it is a deep neural network, numerical differentiation can be used to approximate $\nabla f(\boldsymbol{x}^{(t)})$. Such

 

approximation is typically doable, because we are only evaluating the gradient at $\boldsymbol{x}^{(t)}$, not trying to obtain the entire function $\nabla f(\boldsymbol{x})$ for any $\boldsymbol{x}$.

How about the step size $\alpha^{(t)}$? As a rule of thumb, we can always choose a constant $\alpha$ and see how well it converges. This type of **constant step size** tends to work well in practice although it requires tuning. Alternatively, we can adopt a **line search** method in the optimization literature, e.g., the Amijo and Wolfe line search (See Nocedal and Wright Chapter 3). For analysis purpose, we may pose the optimal step size problem as an **exact line search** problem:

$$\alpha^{(t)} = \underset{\alpha}{\text{argmin}} \quad f\left(\boldsymbol{x}^{(t)} + \alpha \boldsymbol{d}^{(t)}\right), \tag{1.66}$$

where $\boldsymbol{d}^{(t)} = -\nabla f(\boldsymbol{x}^{(t)})$. By solving this scalar minimization, we are guaranteed that the $\alpha$ will make the maximum reduction of the objective along the direction $\boldsymbol{d}^{(t)}$.

**Exercise 3.4.** Consider three step size rules:

(1) $\alpha^{(t)} = \alpha$;

(2) $\alpha^{(t)} = \alpha / \|\nabla f(\boldsymbol{x}^{(t)})\|_2$;

(3) $\alpha^{(t)} = \alpha / t$.

Give an interpretation of each rule, and comment on their pros and/or cons.

**Exercise 3.5**. Assume that the objective function is $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$. At $\boldsymbol{x}^{(t)}$, show that the optimal step size according to the exact line search method is given by

$$\alpha^{(t)} = -\frac{\nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d}^{(t)}}{\boldsymbol{d}^{(t)T}\boldsymbol{H}\boldsymbol{d}^{(t)}}.$$

With the exact line search, gradient descent is guaranteed to converge to the global minimum if $f$ is convex. The following result is due to Luenberger and Ye (Chapter 8).

**Theorem 13.** *Assume that $f$ is twice differentiable so that $\nabla^2 f$ exists. Let $\boldsymbol{x}^*$ be the global minimizer. Assume that $\lambda_{\min}\boldsymbol{I} \preceq \nabla^2 f(\boldsymbol{x}) \preceq \lambda_{\max}\boldsymbol{I}$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Run gradient descent with exact line search. Then,*

$$f(\boldsymbol{x}^{(t+1)}) - f(\boldsymbol{x}^{(t)}) \leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right)^2 \left(f(\boldsymbol{x}^{(t)}) - f(\boldsymbol{x}^*)\right), \tag{1.67}$$

*Thus, $f(\boldsymbol{x}^{(t)}) \to f(\boldsymbol{x}^*)$ as $t \to \infty$.*

 

The implication of the theorem can be seen from an inductive argument:

$$\begin{aligned} f(\boldsymbol{x}^{(t+1)}) - f(\boldsymbol{x}^{(t)}) &\leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right)^2 \left(f(\boldsymbol{x}^{(t)}) - f(\boldsymbol{x}^*)\right) \\ &\leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right)^4 \left(f(\boldsymbol{x}^{(t-1)}) - f(\boldsymbol{x}^*)\right) \\ &\leq \vdots \\ &\leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right)^{2t} \left(f(\boldsymbol{x}^{(1)}) - f(\boldsymbol{x}^*)\right). \end{aligned}$$

Since $f(\boldsymbol{x}^{(1)}) - f(\boldsymbol{x}^*)$ is bounded (or otherwise there will not exist a solution), $f(\boldsymbol{x}^{(t+1)}) - f(\boldsymbol{x}^{(t)}) \to 0$ as $t \to \infty$ because $1 - \frac{\lambda_{\min}}{\lambda_{\max}} < 1$.

## Lagrange Multiplier

The general form of a constrained optimization is

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \geq 0, \qquad i = 1, \ldots, m \\ & h_j(\boldsymbol{x}) = 0, \qquad j = 1, \ldots, k. \end{aligned} \tag{1.68}$$

The optimality condition of a constrained optimization requires more work than an unconstrained optimization. We need to first define a function called the Lagrangian function

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\nu}) \stackrel{\text{def}}{=} f(\boldsymbol{x}) - \sum_{i=1}^{m} \mu_i g_i(\boldsymbol{x}) - \sum_{j=1}^{k} \nu_j h_j(\boldsymbol{x}). \tag{1.69}$$

The variables $\boldsymbol{\mu} \in \mathbb{R}^m$ and $\boldsymbol{\nu} \in \mathbb{R}^k$ are called the **Lagrange multipliers** or the **dual variables**. The first order necessary optimality is stated in the following **Karush-Kahn-Tucker** (KKT) conditions:

**Theorem 14** (KKT Conditions)**.** *If $(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*)$ is the solution to the constrained optimization in Equation (1.68), then all the following conditions should hold:*

*(i)* $\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*) = \mathbf{0}$. *(Stationarity)*

*(ii)* $g_i(\boldsymbol{x}^*) \geq 0$ *and* $h_j(\boldsymbol{x}^*) = 0$ *for all* $i$ *and* $j$. *(Primal Feasbility)*

*(iii)* $\mu_i^* \geq 0$ *for all* $i$ *and* $j$. *(Dual Feasibility)*

*(iv)* $\mu_i^* g_i(\boldsymbol{x}^*) = 0$ *for all* $i$ *and* $j$. *(Complementary Slackness)*

 

Here is an interpretation of the KKT Condition. The **stationarity condition** states that the gradient $\nabla_{\boldsymbol{x}}\mathcal{L}$ evaluated at the optimal point $(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*)$ should be zero. This implies that $(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*)$ is a **saddle point** of $\mathcal{L}$. If we do not have any constraint, then the stationarity is reduced to the first order necessary optimality of an unconstrained problem where $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*) = \boldsymbol{0}$. The **primal feasibility** states that at the optimal point $(\boldsymbol{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\nu}^*)$ the constraints $g_i(\boldsymbol{x}^*) \geq 0$ and $h_j(\boldsymbol{x}^*) = 0$ must be satisfied, for otherwise the solution is invalid. It is called "primal" because the condition only involves the primal variable $\boldsymbol{x}^*$. The **dual feasibility** requires that the Lagrange multiplier $\boldsymbol{\mu} \succeq 0$. Note that the dual feasibility only applies to the inequality constraint $g_i(\boldsymbol{x}) \geq 0$, not the equality constraint $h_j(\boldsymbol{x}) = 0$. The Lagrange multiplier of the equality constraint $\boldsymbol{\nu}$ can take any real number. The **complementary slackness** states that $\mu_i^*$ and $g_i(\boldsymbol{x}^*)$ has to be complementary to each other. If $g_i(\boldsymbol{x}^*) > 0$ then $\mu_i^* = 0$; If $\mu_i^* > 0$ then $g_i(\boldsymbol{x}^*) = 0$. We can also have both: $g_i(\boldsymbol{x}^*) = 0$ and $\mu_i^* = 0$. Geometrically, complementary slackness says that the Lagrange multiplier is zero if $\boldsymbol{x}^*$ is in the interior of the constraint.

Without going into the details of constrained optimization (which will take a few dedicated lectures just to setup the mathematical background), let us study a few examples and gain some insights of how to proceed a constrained optimization. Readers interested in the theoretical aspects of constrained optimization should consult Boyd and Vandenberghe's *Convex Optimization* Ch.5 or Nocedal and Wright's *Numerical Optimization* Ch.12.

**Example 1**. Consider the optimization

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \;\; & \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|^2, \\ \text{subject to} \;\; & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \end{aligned} \tag{1.70}$$

The Lagrangian function of the problem is

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|^2 - \boldsymbol{\nu}^T(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}).$$

The stationarity condition implies that $\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x}, \boldsymbol{\nu}) = \boldsymbol{0}$, which means that $2(\boldsymbol{x}-\boldsymbol{x}_0) - \boldsymbol{A}\boldsymbol{\nu} = \boldsymbol{0}$. Thus, we have $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{A}^T\boldsymbol{\nu}$. Multiplying both sides by $\boldsymbol{A}$ yields $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{A}\boldsymbol{A}^T\boldsymbol{\nu}$, and since the primal feasibility implies that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$, we can show that

$$\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{\nu} = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_0 \quad \Rightarrow \quad \boldsymbol{\nu} = (\boldsymbol{A}\boldsymbol{A}^T)^{-1}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_0).$$

Therefore, the solution is $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}_0)$

In this example, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$ implies that the linear constraints are consistent. Therefore, it is not possible to have an overdetermined system where $\boldsymbol{A}$ is a tall matrix. If $\boldsymbol{A}$ is a square matrix and if $\boldsymbol{A}^{-1}$ exist, then the optimal solution is $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$. (Why?) Therefore, to

 

make Example 1 valid, $\boldsymbol{A}$ must be a short and fat matrix.

Here is a MATLAB / Python program using `CVX` to verify the correctness of our derivation.

```
% MATLAB code: Use CVX to solve min ||x - x0||, s.t. Ax = y
m = 3; n = 2*m;
A     = randn(m,n); xstar = randn(n,1);
y     = A*xstar;
x0    = randn(n,1);
cvx_begin
   variable x(n)
   minimize( norm(x-x0) )
   subject to
        A*x == y;
cvx_end
% you may compare with the solution x0 + A'*inv(A*A')*(y-A*x0).
```

**Exercise 3.6**. Solve the optimization problem

$$\begin{aligned} &\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \;\; \|\boldsymbol{x}-\boldsymbol{x}_0\|_2, \\ &\text{subject to} \;\; \boldsymbol{w}^T\boldsymbol{x} = w_0, \end{aligned} \tag{1.71}$$

and comment on its geometric interpretation. Compare your solution with the one returned by `CVX`.

With the numerical solver `CVX` we can also tackle problems that are not differentiable, for example, when the objective function is $\|\boldsymbol{x}\|_1$.

**Example 2**. Solve the LASSO problem using `CVX`:

$$\begin{aligned} &\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \;\; \|\boldsymbol{x}\|_1, \\ &\text{subject to} \;\; \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \end{aligned} \tag{1.72}$$

```
% MATLAB code: Use CVX to solve min ||x||_1, s.t. Ax <= y
m = 100; n = 50;
A  = randn(m,n);
x0 = randn(n,1);
y  = A*x0 + rand(m,1);
cvx_begin
  variable x_l1(n)
  minimize( norm( x_l1, 1 ) )
```

 

```
  subject to
    A*x_l1 == y;
cvx_end
```

The last example below involves both equality and inequality constraints.

**Example 3**. Solve the following least squares over positive quadrant problem, both analytically and numerically via `CVX`.

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \;\; & \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{b}\|^2, \\ \text{subject to} \;\; & \boldsymbol{x}^T\mathbf{1}=1, \quad \text{and} \quad \boldsymbol{x}\geq\mathbf{0}. \end{aligned} \tag{1.73}$$

The geometric interpretation of this problem is that we are trying to project $\boldsymbol{b}$ onto a surface defined by $\boldsymbol{x}^T\mathbf{1}=1$ and $\boldsymbol{x}\geq\mathbf{0}$. The surface is essentially a hyperplane lying in the first quadrant. The first constraint $\boldsymbol{x}^T\mathbf{1}=1$ says that the sum of all elements of $\boldsymbol{x}$ is 1, which can be interpreted as the probabilities of a random variable at $n$ states. The non-negativity suggests that the values of $\boldsymbol{x}$ cannot be negative, which is necessary for probabilities.

To solve the problem, we first write down the Lagrangian function:

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda},\gamma)=\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2-\boldsymbol{\lambda}^T\boldsymbol{x}-\gamma(1-\boldsymbol{x}^T\mathbf{1}).$$

The stationarity condition implies that $\nabla_{\boldsymbol{x}}\mathcal{L}=\boldsymbol{x}-\boldsymbol{b}-\boldsymbol{\lambda}+\gamma\mathbf{1}=\mathbf{0}$, and so we have $x_i=b_i+\lambda_i-\gamma$. The complementary slackness implies that $\lambda_i x_i=0$. Let us consider two cases. Case 1: If $\lambda_i=0$, then $x_i=b_i-\gamma$. So by complementary slackness we have $x_i\geq 0$ and so $b_i-\gamma\geq 0$. Case 2: If $\lambda_i>0$, then by complementary slackness we have $x_i=0$. So $x_i=b_i+\lambda_i-\gamma=0$, and hence $b_i+\lambda_i=\gamma$. Since $\lambda_i>0$, this further implies that $b_i<\gamma$. Combining these observations, we conclude that

- If $b_i>\gamma$, then $x_i=b_i-\gamma\geq 0$;
- If $b_i<\gamma$, then $x_i=0$;
- If $b_i=\gamma$, then $x_i=\lambda_i$, which implies that $x_i=0$.

This can be compactly written as

$$\boldsymbol{x}=\max(\boldsymbol{b}-\gamma\mathbf{1},0).$$

It remains to determine $\gamma$. By primal feasibility, we have $\boldsymbol{x}^T\mathbf{1}=1$. This implies that $\max(\boldsymbol{b}-\gamma\mathbf{1},0)^T\mathbf{1}=1$. Therefore, we can determine $\gamma$ by finding the root of the function $g(\gamma)=\max(\boldsymbol{b}-\gamma\mathbf{1},0)^T\mathbf{1}-1$.

Here is the MATLAB / Python code to solve this problem

 

```
%MATLAB code: solve min ||x-b|| s.t. sum(x) = 1, x >= 0.
n = 10;
b = randn(n,1);
fun = @(gamma) myfun(gamma,b);
gamma = fzero(fun,0);
x = max(b-gamma,0);
```

where the function `myfun` is defined as

```
function y = myfun(gamma,b)
y = sum(max(b-gamma,0))-1;
```

The same problem can be solved using `CVX`.

```
%MATLAB code: Use CVX to solve min ||x-b|| s.t. sum(x) = 1, x >= 0.
cvx_begin
   variable x(n)
   minimize( norm(x-b) )
   subject to
        sum(x) == 1;
        x      >= 0;
cvx_end
```

 

# Reading Suggestions

Much of the concepts we discussed in this chapter can be found in standard undergraduate / graduate textbooks. The references suggested below are a few of the most relevant ones.

### Linear Algebra

For linear algebra, any undergraduate textbooks would suffice. Readers looking to more in-depth discussion can consult an intermediate level textbook. For quick review, a very well-written tutorial can be found from Stanford's CS 229.

Textbook / Beginner:
Gilbert Strang, Linear Algebra and Its Applications, 5th Edition.

Textbooks / Intermediate.
Carl Meyer, Matrix Analysis and Applied Linear Algebra, SIAM, 2000.

Tutorials / Notes.
`http://cs229.stanford.edu/section/cs229-linalg.pdf`

Good to know list: The keywords below are what we called "good to know", meaning that it would help you understand the mathematical "jargons" we use in ECE 595. For linear algebra, the list consists of: Matrix-vector multiplication $\boldsymbol{A}\boldsymbol{x}$, transpose $\boldsymbol{A}^T$, symmetric matrices $\boldsymbol{A} = \boldsymbol{A}^T$, norm $\|\boldsymbol{x}\|$, trace $\text{Tr}(\boldsymbol{A})$, inverse $\boldsymbol{A}^{-1}$, determinant $|\boldsymbol{A}|$, eigenvalue and eigenvector $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$, range space $\mathcal{R}(\boldsymbol{A})$, null space $\mathcal{N}(\boldsymbol{A})$, projection $\boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$, pseudo-inverse $\boldsymbol{A}^+$, singular value decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$, unitary matrix $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$, positive semi definite $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \geq 0$, matrix derivative $\nabla_{\boldsymbol{x}}(\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x})$.

### Probability

In terms of probability, an excellent reference book is Dimitri Bertseka's Intro to Probability, used in MIT's 6.012. At Purdue, an equivalent course is ECE 302. A thorough exposure to these topics is necessary for ECE 595. Advanced probability such as Purdue's ECE 600 is good to have but optional.

Textbook / Beginner:
Dimitri Bertsekas, Introduction to Probability, Athena Scientific, 2008, 2nd Edition.

Tutorials / Notes:
`https://engineering.purdue.edu/ChanGroup/ECE302/lecture.html`

Good to know list: The good to know list for probability consists of most of the key concepts in ECE 302. They are: Random variable $X$, probability density function $p_X(x)$, cumulative

 

distribution function $F_X(x)$, expectation $\mathbb{E}[X]$, variance $\text{Var}[X]$, function of random variable $\mathbb{E}[g(X)]$, joint distribution $p_{\boldsymbol{X}}(\boldsymbol{x})$, covariance $\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^T] - \mathbb{E}[\boldsymbol{X}]\mathbb{E}[\boldsymbol{X}]^T$, joint Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, conditional distribution $p_{\boldsymbol{X}|Y}(\boldsymbol{x}|y)$, prior $p_Y(y)$, Law of Large Number, Central Limit Theorem.

**Optimization**

Unconstrained optimization for a single variable should be covered in most of the freshmen calculus courses. For constrained optimization, advanced calculus in sophomore year should have some basic coverage. The optimization we use in this course is slightly more advanced. However, we only require some basic concepts and not the bulk of the technical details. The following references books are more than sufficient for ECE 595 (perhaps an overkill).

Textbook / Intermediate:

Jorge Nocedal and Stephen Wright, Numerical Optimization, Springer 2nd Edition.
Stephen Boyd and Lieven Vandenberghe, Convex Optimization, Cambridge 2004.

Textbook / Advanced:

Dimitri Bertsekas, Angelia Nedic and Asuman Ozdaglar, Convex Analysis and Optimization, Athena Scientific 2003.

Tutorials / Notes:

`http://cs229.stanford.edu/section/cs229-cvxopt.pdf`
`http://cs229.stanford.edu/section/cs229-cvxopt2.pdf`
`http://eceweb.ucsd.edu/~gert/ECE273/CvxOptTutPaper.pdf`

Good to know list: The good to know list for optimization is basically the first few chapter of Boyd and Vandenberghe's Convex Optimization. However, we do not require readers to master all concepts. It would suffice to just know the terminology and understand it's basic meaning. The topics are: Convex function, convex set, operations which preserve convexity, Lagrange multiplier, KKT conditions, primal optimal, dual optimal, complementary slackness, constrained optimization, duality theorem.

 