Deep Learning

midterm exam (take home): April 30–Due May 7

Name : 정 상 만 Student Number : 20190310290

Problem 1.
Write down a summary for historical trends in deep learning.

Solution)

The summary from the lecture notes...

• 1940-60 (cybernetics)
-development of theories of biological learning (McCulloch and Pitts, 1943; Hebb, 1949)
-implementation of the first models, the perceptron (Rosenblatt, 1958)
-adaptive linear element (ADALINE) (Widrow and Hoff, 1960)
• 1980-90 (connectionism, cognitive science)
-a large number of simple computational units can achieve intelligent behavior when networked together
-distributed representation (Hinton et al., 1986)
-backpropagation to train a neural network with one or two hidden layers (Rumelhart et al., 1986)
-modelling sequences with neural network (LSTM (long short-term memory) network)
• 2006-present (deep learning)
-CIFAR(Canadian Institute of Advanced Research) NCAP(Neural Computation and Adaptive Perception) research initiative
-Hinton et al. (2006); showed that a deep belif network could be efficiently trained using a strategy called greedy layer-wise pretraining
-Bengio et al. (2007); Ranzato et al. (2007) showed deep neural networks outperformed competing AI systems based on other machine learning technologies as well as hand-designed functionality

+

딥 러닝 구조는 인공신경망(Artificial neural networks)에 기반하여 설계된 개념으로, 1980년 Kunihiko Fukushima 에 의해 소개된 Neocognitron (패턴 인식에서의 모델로써, self-organizing neural network의 일종)에서 시작되었으며, 1989년에 오류역전파 알고리즘(backpropagation algorithm)에 기반하여 우편물에 손으로 쓰여진 우편번호를 인식하는 deep neural networks를 소개하였다. 이 때, 알고리즘은 성공적으로 동작했으나 신경망 학습에 소요되는 시간이(우편물의 10개 숫자를 인식하기 위해 학습하는 시간) 거의 3일이 걸렸기 때문에 다른 분야에 일반적으로 적용하기에는 비현실적인 것으로 여겨졌다.

이는 다양한 원인이 있는데, 대표적으로 과적합 문제, 시뮬레이션의 초기 상태를 어떻게 선택하느냐에 따라 발산하는 문제, 생물학적 신경망과는 다르다는 이슈 등이 끊임 없이 제기되면서 인공신경망은 관심에서 멀어졌고, 90년대와 2000년대에 Support Vector Machine (SVM)과 같은 기법들이 각광받게 된다.

본격적으로 딥 러닝이란 용어를 사용한 것은 2000년대 제프리 힌튼과 Ruslan Salakhutdinov에 의해서이며, 기존 신경망의 과적합 문제를 해결하기 위해 다양한 연구를 시도하여 과적합 문제를 해결하고자 하였다.

현재 딥 러닝은 기존 인공신경망 모델의 단점이 차츰 극복하게 되고, 하드웨어의 발전이 뒤따르면서 학습 시간을 단축시키며, 빅 데이터가 학습에 이용되면서 다양한 분야에서 그 성능이 각광받고 있다. 특히, 자동 음성 인식과 컴퓨터비전 분야에서 최고 수준의 성능을 보여주고 있다. 최근에는 Convolution Neural Networks 기반의 딥 러닝 알고리즘이 뛰어난 성능을 발휘하고 있으며, 컴퓨터비전과 음성인식등의 분야에서 특히 탁월한 성능을 보이고 있다.

■

Problem 2.

Write down the definition for eigendecomposition for a real symmetric matrix A. Write down the definition for singular value decomposition for a general matrix A. Using these definition, discuss about Principle Component Analysis for a given collection of m vectors, $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$, $x^{(i)} \in R^n$, $i = 1, \cdots, m$.

Solution)

Definition. **(Eigendecomposition for a square matrix)**

Let A be a square $n \times n$ matrix with n linearly independent eigenvectors q_i of A, and $i = 1, ..., n$. Then A can be factorized as $A = Q \Lambda Q^{-1}$ where Q is the square $n \times n$ matrix whose $i-th$ column is the eigenvector q_i of A, and Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, $\Lambda_{ii} = \lambda_i$.

(In real symmetric matrices)

For $n \times n$ real symmetric matrix A, A can be decomposed as $A = Q \Lambda Q^T$ where Q is an orthogonal matrix whose columns are the eigenvectors of A, and Λ is a diagonal matrix whose entriesare the eigenvalues of A.

Definition. **(Singular Value Decomposition of $m \times n$ matrix)**

For a real or complex $m \times n$ matrix A, A is a factorization of the form $A = U \Sigma V^T$ such that $U : m \times m$, $V : n \times n$ are an orthogonal matrix and $\Sigma : m \times n$ diagonal matrix.

Definition. **Principle component Analysis (PCA)**

For the data matrix $X = \begin{pmatrix} x_{11} \cdots x_{1j} \cdots x_{1q} \\ \vdots \ \vdots \ \vdots \ \vdots \ \vdots \\ x_{n1} \cdots x_{nj} \cdots x_{nq} \end{pmatrix}$, The idea of PCA is to summarize the q variables $X = (X_1, ..., X_q)^T$ using linear combinations (i.e., weighted averages):

$$Y_1 = a_1^T X = a_{11} X_1 + \cdots + a_{j1} X_j + a_{q1} X_q \quad (= X^T a_1) \Leftrightarrow \begin{pmatrix} y_{11} \\ \vdots \\ y_{n1} \end{pmatrix} = a_{11} \begin{pmatrix} x_{11} \\ \vdots \\ x_{n1} \end{pmatrix} + \cdots + a_{1j} \begin{pmatrix} x_{1j} \\ \vdots \\ x_{nj} \end{pmatrix} + \cdots + a_{q1} \begin{pmatrix} x_{1q} \\ \vdots \\ x_{nq} \end{pmatrix}$$

Such a linear combination can be written in matrix notation

$$y_1 = X a_1 \Leftrightarrow \begin{pmatrix} y_{11} \\ \vdots \\ y_{n1} \end{pmatrix} = \begin{pmatrix} x_{11} \cdots x_{1q} \\ \vdots \ \ddots \ \vdots \\ x_{n1} \cdots x_{nq} \end{pmatrix} \begin{pmatrix} a_{11} \\ \vdots \\ a_{q1} \end{pmatrix}.$$

How should we choose the weights $a_1 = (a_{11}, ..., a_{q1})^T$?, we call a linear combination **normalized** if $\sum_{j=1}^{q} a_{1j}^2 = 1$. For PCA, we assume in the following that the data matrix X is **mean centered**. i.e., the mean is zero for all variables / columns that if

$$\text{Given } X = \begin{pmatrix} x_{11} \cdots x_{1j} \cdots x_{1q} \\ \vdots \ \vdots \ \vdots \ \vdots \ \vdots \\ x_{n1} \cdots x_{nj} \cdots x_{nq} \end{pmatrix}, \ \frac{1}{n} \sum_{i=1}^{n} x_{ij} = 0 \text{ for all columns.}$$

If this is not the case, first center the data by subtracting the mean. Now, the first principal component PC_1 is the normalized linear combination $y_1 = Xa_1$ that has **largest sample variance**.
In that case, PC_j is the normalized linear combination $y_j = Xa_j$ that has **largest sample variance** subject to the constraint that its sample correlations with $PC_1, ..., PC_{j-1}$ **equal zero** $(a_j^T a_k = 0, j \neq k)$. i.e., PC_1 is direction of largest variance, PC_2 is perpendicular to PC_1 again largest variance. PC_3 perpendicular to PC_1, PC_2 again largest variance, etc.

PC_1 : straight line with smallest orthogonal distance to all points.
PC_1 & PC_2 : plane with smallest orthogonal distance to all points.
First k PC 's : linear subspace of dimension k with smallest ortbogonal distance to all points.

Let us define **Loadings** and **Scores.**

The vectors a_j are called **Loadings** $a_j = (a_{1j}, a_{2j}, ..., a_{qj})^T$, a_{1j} : loading of first variable on PC_j ..etc.
The values y_{ij} are called **Scores** $y_j = (y_{1j}, ..., y_{ij}, ..., y_{nj})^T$.

Note) a_j means the eigenvectors for the covariance matrix.

**** Computation of PCA**
There are several ways for computing principal components.

1. Eigendecomposition of the covariance matrix.
2. Singular value decomposition of the data matrix.

1. Eigendecomposition

(1) Calculate the eigendecomposition of the sample covariance matrix S : $n \times n$ matrix ($n = m$ in this case) to obtain the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_q$ and the eigenvectors $a_1, a_2, ..., a_q$.

(2) Sort the eigenvalues and accordingly the eigenvectors a_j of A such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_q$.

(3) The j-th principal component PC_j is given by $y_j = Xa_j$ where a_j is the j-th eigenvector (j-th column of A). (i.e., linear transformation for matrix X)

**** Properties of PCA**
- The sample variance of PC_j (y_j) equals λ_j, the j-th largest eigenvalue of S.
- The total (sample) variance of all q variables equals $\sum_{j=1}^{q} S_{jj} = \sum_{j=1}^{q} \lambda_j$.
- The first k principal components $PC_1, ..., PC_k$ explain $\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{q} \lambda_j}$ percent of the total variance.
- The sample correlations between different y_j's are zero.
- All PCs y_j have sample mean zero.

**** PCA by hand : example**

Given the data matrix $X=\begin{pmatrix} 0.2 & 0.45 & 0.33 & 0.54 & 0.77 \\ 5.6 & 5.89 & 6.37 & 7.9 & 7.87 \\ 3.56 & 2.4 & 1.95 & 1.32 & 0.98 \end{pmatrix}$, find the principal components. (In this case, rows are the variables, and columns are the observations).

solution) First, we make X' that the matrix applied the centered mean zero as:

$$X'=\begin{pmatrix} -1.1930 & -0.0370 & -0.5919 & 0.3792 & 1.4427 \\ -1.0300 & -0.7647 & -0.3257 & 1.0739 & 1.0464 \\ 1.5012 & 0.3540 & -0.0910 & -0.7140 & -1.0502 \end{pmatrix}.$$

The covariance matrix Σ if $n=5$ is:

$$\Sigma=cov(X)=\frac{1}{n-1}X'X'^{T}=\begin{pmatrix} 0.0468 & 0.1990 & -0.1993 \\ 0.1990 & 1.1951 & -1.0096 \\ -0.1993 & -1.0096 & 1.0225 \end{pmatrix}$$

Then, after finding the eigenvalues and the eigenvectors, we obtain the result by eigendecomposition as follows.

$$\Sigma A=A\Lambda \text{ where } A=\begin{pmatrix}\overrightarrow{\alpha_1} & \overrightarrow{\alpha_2} & \overrightarrow{\alpha_3}\end{pmatrix}=\begin{pmatrix} 0.5699 & 0.7798 & 0.2590 \\ 0.5765 & -0.6041 & 0.5502 \\ -0.5855 & 0.1643 & 0.7938 \end{pmatrix},\ \Lambda=\begin{pmatrix}\lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3\end{pmatrix}=\begin{pmatrix} 2.7596 & 0 & 0 \\ 0 & 0.1618 & 0 \\ 0 & 0 & 0.0786 \end{pmatrix}.$$

We can choose the eigenvector corresponding to $\max\{\lambda_i\}$, and can make the variable z_1 that

$$\overrightarrow{z_1}=\overrightarrow{\alpha_1}^{T}X=(0.5699 \ \ 0.5765 \ \ -0.5855)\begin{pmatrix}\overrightarrow{x_1} \\ \overrightarrow{x_2} \\ \overrightarrow{x_3}\end{pmatrix}=0.5699\overrightarrow{x_1}+0.5765\overrightarrow{x_2}-0.5855\overrightarrow{x_3}$$
$$=(-2.1527 \ \ -0.6692 \ \ -0.4718 \ \ 1.2533 \ \ 2.0404)$$

In the same way to z_1, we obtain z_2 and z_3 that

$$Z=A^{T}X \Leftrightarrow \begin{pmatrix}\overrightarrow{z_1} \\ \overrightarrow{z_2} \\ \overrightarrow{z_3}\end{pmatrix}=\begin{pmatrix}\overrightarrow{\alpha_1}^{T} \\ \overrightarrow{\alpha_2}^{T} \\ \overrightarrow{\alpha_3}^{T}\end{pmatrix}X=\begin{pmatrix} -2.1527 & -0.6692 & -0.4718 & 1.2533 & 2.0404 \\ -0.0615 & 0.4912 & -0.2798 & -0.4703 & 0.3204 \\ 0.3160 & -0.1493 & -0.4047 & 0.1223 & 0.1157 \end{pmatrix}.$$

The covariance matrix of Z is $cov(Z)=\begin{pmatrix} 2.7596 & 0 & 0 \\ 0 & 0.1618 & 0 \\ 0 & 0 & 0.0786 \end{pmatrix}$. Note that we can see the uncorrelated relation that the covariance of three variables z_1, z_2 and z_3 is zero. (Σ의 고유벡터는 서로 직교하는데, X가 이 고유벡터를 새로운 축인, 즉 주성분으로 하여 선형변환 되었기 때문).

We know that $Var(\overrightarrow{z_i})=\lambda_i$, $\frac{\lambda_1}{\lambda_1+\lambda_2+\lambda_3}=\frac{2.7596}{2.7596+0.1618+0.0786}=0.920$. Thus, if we take z_1 and skip z_2 and z_3, then we can reduce from 3-dimension to 1-dimension with preserving 92% of the variance of X.

** Summary (영작이 서툴러 한글로 다시 작성하겠습니다.)

- 고유값 분해를(공분산 방법) 통한 PCA 수행 절차

1. 기존 데이터 X의 공분산 행렬 계산
2. 공분산행렬의 고유값과 고유벡터 계산
3. 고유값의 크기 순서대로 고유벡터를 나열
4. 정렬된 고유벡터 가운데 가장 큰 일부들 선택
5. 그에 따른 해당 고유벡터와 X의 내적 (고유벡터를 새로운 기저로 하여 원데이터를 선형변환)

- SVD 이용한 방법과 PCA의 관계

SVD의 정의에서, U와 V에 속한 열벡터(특이벡터)는 서로 직교하는 성질을 지니고, Σ의 특이값은 모두 0 또는 양수이며 내림차순으로 정렬될 때, Σ의 k번째 대각원소는 $\Sigma_k = \sqrt{\lambda_k}$ 인 관계를 갖는다. $A = U\Sigma V^T$라 두면,

$$\begin{aligned} A^T A &= (U\Sigma V^T)^T U\Sigma V^T \\ &= V\Sigma U^T U\Sigma V^T \\ &= V\Sigma^2 V^T \\ &= V\Lambda V^T \end{aligned}$$

여기서 대각성분이 행렬 A의 특이값이고, 나머지 성분이 0인 Σ는 대각행렬이므로 대각행렬의 거듭제곱은 대각원소들만 거듭제곱해준 결과와 같다.

따라서 Σ의 제곱은 각 대각원소, 즉 행렬 A의 특이값들을 제곱해준 값과 같고, 이는 A의 고유값에 제곱근을 취한 값과 동일하므로 Σ를 제곱한 행렬 Λ는 $A^T A$의 고유값으로 이뤄진 행렬이 된다. 이는 정확히 주성분 분석의 결과와 같다.

■

Problem 3. (Numerical test)

In the following example, we will learn how to perform the PCA(Principle Compoenet Analysis) for a given set of vectors:

1. Generate 10 vectors $x^{(i)}$ in R^{50} such that $x_j^{(i)} = rand()$, for $j = 1:50$, for $i = 1:10$
2. Using PCA, find their lossy compression vectors in R^3.
3. Calculate decoding vectors $d^{(i)}$ for each $x^{(i)}$ and compute the error, $\sqrt{\frac{1}{10}\sum_{i=1}^{10} \| x^{(i)} - d^{(i)} \|^2}$.

Solution)

```
clear all, clc
% Data Size
n=10; m=50;
% Initial
X=zeros(n,m); musum=0;
% Data matrix X
for i=1:n
    for j=1:m
        X(i,j)=rand();
    end
    musum=musum+X(i,:);
end
mu=1/n*musum;
% Mean centering
Z=X-mu;
% Covariance
  S=1/n*(Z'*Z);
% Eigenvalues
[V D]=eig(S);
lambda1=sort(eig(S),'descend');
lambda2=lambda1(1:9)
% Fraction of total variance
f=@(r) sort(sum(lambda2(1:r))/sum(lambda2),'descend'); r=length(lambda2);
for i=1:r
    lam(i)=f(i);
end
alpha=find(lam>=0.85);
new_r=length(alpha); % reduced dimension. (choose dimensionality)
new_V=V(:,end-new_r+1:end);
% Reduced dimensionality data
for i=1:10
W(i,:)=new_V'*X(i,:)';
end
W
% Scree graph
plot(lambda2,'r-o')
xlabel('component number')
ylabel('Eigenvalues(Variance of X)')
title('Scree plot for rows x1...xm')
```

(1) Generate 10 vectors $x^{(i)}$ in R^{50} such that $x_j^{(i)} = rand()$, for $j = 1:50$, for $i = 1:10$

```
clear all, clc
% Data Size
n=10; m=50;
% Initial
X=zeros(n,m); musum=0;
% Data matrix X
for i=1:n
    for j=1:m
        X(i,j)=rand();
    end
    musum=musum+X(i,:);
end
```

(2) Using PCA, find their lossy compression vectors in R^3.

```
new_V=V(:,end-new_r+1:end);
```

(3) Calculate decoding vectors $d^{(i)}$ for each $x^{(i)}$ and compute the error, $\sqrt{\frac{1}{10}\sum_{i=1}^{10} \| x^{(i)} - d^{(i)} \|^2}$.

```
% Decoding vectors and error
d=W*new_V';
normsum=0;
for i=1:10
    normsum=normsum+norm(X(i,:)-d(i,:))^2;
end
error_d=sqrt(1/10*normsum)
```

■

Problem 4.

Discuss about overflow and underflow issues related to the computation of the following function:

$$x^2 \sin(1/x)$$

Solution)

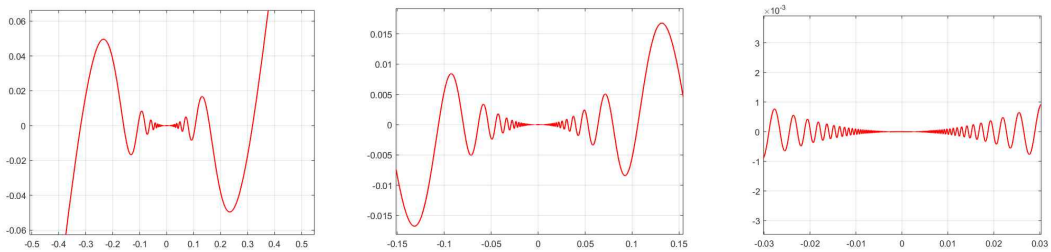


Figure 1. The graph of $f(x) = x^2 \sin(1/x)$

The following function occur the underflow problem. In MATLAB, if x =1.0000e-162, then $f(x) = 0$. If we consider the case of $x \to 0$, but $f(x)$ cannot be equals to 0. On the other way, if we put the value of x is zero, then $f(x)$ is NaN (Not a number).

In the same way, overflow problem occur the similar situation. If x =1.0000e+155, then $f(x) = \infty$ in MATLAB. But we can define the value of $f(x)$ in this case analytically. Also, if we put ∞ in this software, we obtain the value 'inf' (infinity). (This shows the undefined value is well-defined in MATLAB.)

■

Problem 5.

When the gradient descent method is applied to the following problem.

$$x^* = \text{argmin}_x \phi(x), \text{ where } \phi(x) = \frac{1}{2}x^T Ax - x^T b, \; A = \begin{bmatrix} 1 & 0 \\ 0 & 0.1^{15} \end{bmatrix}, \; b = [1 \;\; 1]^T$$

with the initial $x = [0 \;\; 0]^T$, discuss about the convergence behavior and performance of the method.

Solution)

For given the initial $x = [0 \;\; 0]^T$, following the equation is called the gradient descent method:

$$x_{i+1} = x_i - \epsilon \nabla \phi(x_i)$$

where ϵ : learning rate that the controller for modifying the moving distance. The convergence of this algorithm can be different through the properties of f and choosing ϵ properly, and this algorithm converges to local optimal solution. Hence, we cannot ensure the obtained solution is global. Thus the solution of the method depend on the initial x_0.

Now, using MATLAB, we obtained the some numerical results as follows.

If $A = \begin{bmatrix} 1 & 0 \\ 0 & \alpha \end{bmatrix}$, α is parameter, then we can consider the surface of ϕ as follows.

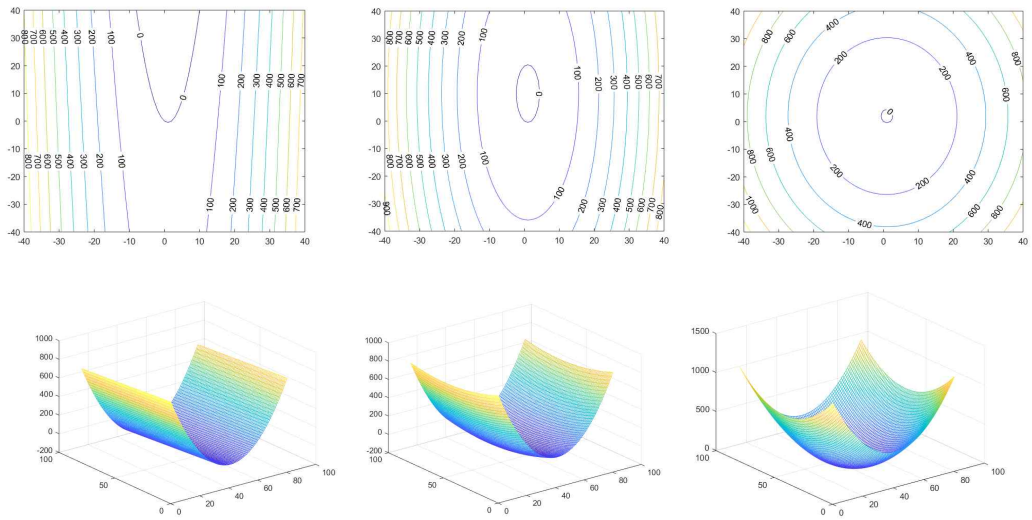


Figure 2. The contour plot and mesh plot of ϕ for $\alpha = 0.1^{15}$, $\alpha = 0.1$, $\alpha = 0.5$, respectively.

Clearly, we can find the minimum easily at $\alpha = 0.5$, even $\alpha = 0.1$. However, our case of $\alpha = 0.1^{15}$ does not.

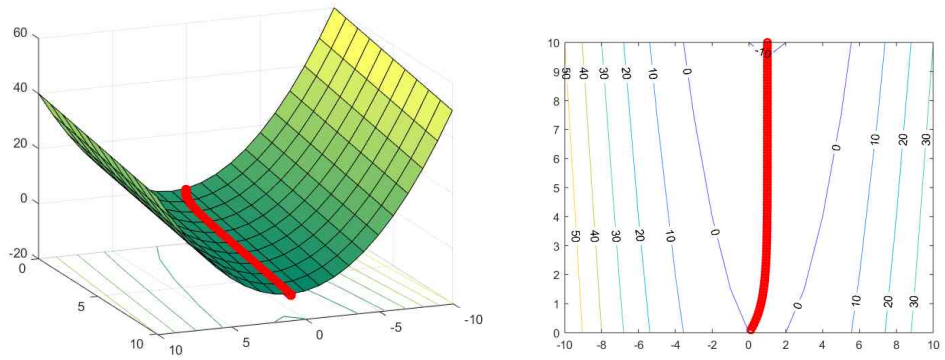


Figure 3. The solution using gradient descent method and mesh plot of ϕ for $\alpha = 0.1^{15}$.

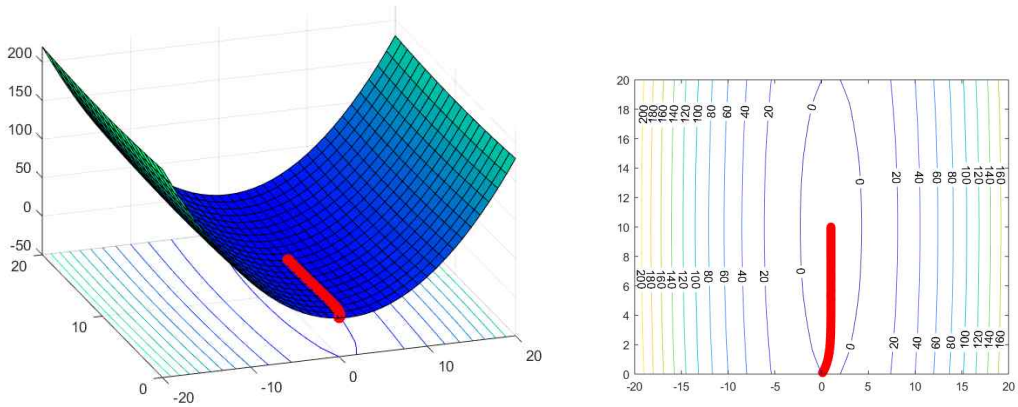


Figure 4. The solution using gradient descent method and mesh plot of ϕ for $\alpha = 0.1$.

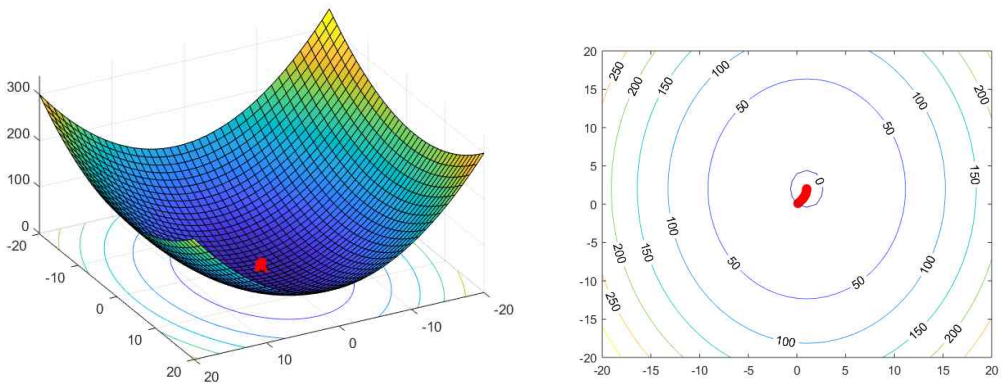


Figure 5. The solution using gradient descent method and mesh plot of ϕ for $\alpha = 0.5$.

Thus, the case of $\alpha = 0.1^{15}$, the performance of the method does not efficient and the convergence behavior goes to infinity if the domain of the surface not defined.

■

Gradient_Descent_test.m

```
clear all, clc

% parameters
x=[0 0]'; % Initial guess
A=[1 0; 0 0.5]; b=[1 1]';
epsilon=0.1; % Learning rate
max_iter=1000;
precision=0.001;
% function phi
phi=@(x) 1/2*x'*A*x-x'*b;
grad_phi=@(x) A*x-b;
fvals=[];
% Gradient descent method
x_new=x;
x_old=x_new;
for i=1:max_iter
    x_new=x_old-epsilon.*grad_phi(x_old);
    iter=i;
    if (norm(grad_phi(x_new))<=precision)
        break
    end
    x_old=x_new;
    xvals(iter,:)=x_new;
    fvals(iter)=phi(x_new);
end
xvals
fvals

% Graph
s=-20:20; t=-20:20;
[X,Y]=meshgrid(s,t);
Z=X.^2/2+0.5*Y.^2/2-X-Y;
figure(1)
contour(X,Y,Z,'ShowText','on')
hold on
plot3(xvals(:,1),xvals(:,2),fvals,'r-o','LineWidth',2)

figure(2)
plot3(xvals(:,1),xvals(:,2),fvals,'r-o','LineWidth',2)
grid on
hold on
surfc(X,Y,Z)
```

Problem 6.

The following data is given. The doctor makes decision on the type of cancers of the patient depending on the given data. In what probability, the patient with sympton2 will have the liver cancer?

	y=sympton1	y=sympton2
x=liver	p(y|x)=1/3	p(y|x)=1/4
x=stomach	p(y|x)=1/5	p(y|x)=1/2

It is also provided that p(x=liver)=1/3 and p(x=stomach)=2/3.

Solution)

The patient with sympton2 will have the liver cancer $\Rightarrow P(x = liver \mid y = sympton2)$.
By Bayes' theorem, we can write this as

$$P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)},\ P(y) = P(y \mid x = liver) + P(y \mid x = stomach) = 3/4,\ P(x = liver) = 1/3.$$

Since the probability for the patient with the liver cancer will have the sympton2 is $P(y \mid x) = 1/4$.
Thus, we obtain the probability as

$$P(x = liver \mid y = sympton2) = \frac{1}{9}.$$

■

Problem 7.

Show the following identity

$$E[\hat{\sigma}_m^2] = E\left[\frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \hat{\mu}_m\right)^2\right] = \frac{m-1}{m}\sigma^2$$

assuming that $x^{(i)}$ are independent and identically distributed with a Gaussian distribution, mean μ and variance σ, where

$$\hat{\mu}_m = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} \ .$$

Solution)

Consider $\sum_{i=1}^{m}\left(x^{(i)} - \hat{\mu}_m\right)^2 = \sum_{i=1}^{m}\left(x^{(i)}\right)^2 - m\,\hat{\mu}_m^2$. Then, the expected value of this expression is

$$\begin{aligned}
E\left(\sum_i \left(x^{(i)} - \hat{\mu}_m\right)^2\right) &= \sum_i E\left(\left(x^{(i)}\right)^2\right) - mE\left(\hat{\mu}_m^2\right) \\
&= \sum_i (\mu^2 + \sigma^2) - m\frac{1}{m^2}\left(\sum_i E\left(\left(x^{(i)}\right)^2\right) + \sum_{i<j} x^{(i)}x^{(j)}\right) \\
&= m\,\mu^2 + m\,\sigma^2 - \frac{1}{m}\left(m\,\mu^2 + m\,\sigma^2 + m(m-1)\mu^2\right) \\
&= m\,\mu^2 + m\,\sigma^2 - \mu^2 - \sigma^2 - (m-1)\mu^2 \\
&= m\mu^2 + m\,\sigma^2 - m\mu^2 - \sigma^2 \\
&= (m-1)\sigma^2
\end{aligned}$$

Now, since $\hat{\sigma}_m^2 = \frac{1}{m}\sum_i \left(x^{(i)} - \hat{\mu}_m\right)^2$, we obtain $E\left(\hat{\sigma}_m^2\right) = \frac{1}{m}(m-1)\sigma^2 = \frac{m-1}{m}\sigma^2$.

■

Problem 8.

For a point estimator $\hat{\theta}$ of the true value θ, discuss about the MSE (Mean Squared Error) related to the bias and variance of the point estimator. In addition, propose a good strategy for reducing the MSE for the point estimator.

Solution)

Preliminary

For the data set $\{x^{(1)}, ..., x^{(m)}\}$ with $i.i.d$, the point estimator is defined by $\hat{\theta}_m = g(x^{(1)}, ..., x^{(m)})$.

The bias for the point estimator is defined by $E(\hat{\theta}_m) - \theta$, θ : true value, E is the expected value of the estimator. (the mean for all of the data).

-과소적합(underfitting) : 모형이 훈련 집합의 오차 값을 충분히 작게 만들지 못할 때 발생
-과대적합(overfitting) : 훈련 오차와 시험 오차의 차이가 너무 클 때 발생
-수용력(capacity) : 모형이 다양한 함수들에 적합하게 하는 능력으로, 주어진 학습 모형의 과대적합 또는 과소적합 가능성을 제어할 수 있음

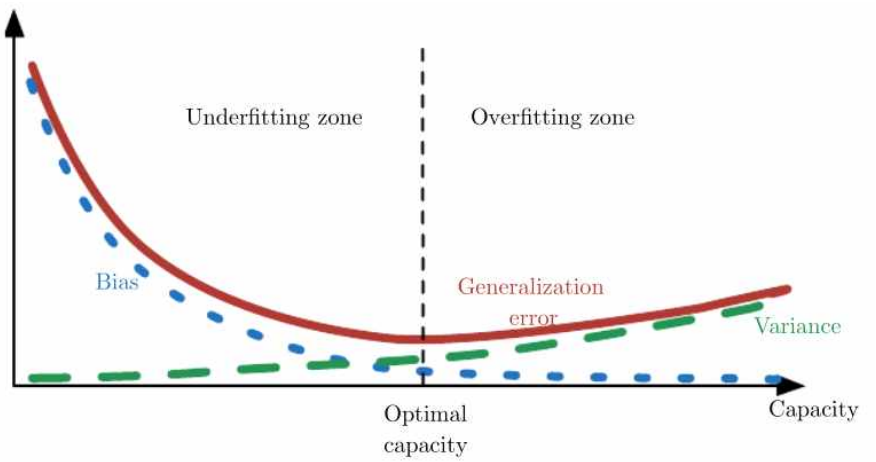
Mean Squared Error

-참값과 추정을 위한 모형의 오차는 크게 편향과 분산 두 축의 오차로써 분류된다.
-어떤 통계적 모형 두 개가 있다고 하면, 추정량이 편향이 큰 모형과 분산이 큰 모형 중 어느 것을 선택하는 것이 더 나은 모형을 선택하는지 결정하기 위해 쓰이는 교차검증 방법 중 하나로, 다음의 식으로 정의되는 추정량들의 평균제곱오차(MSE)를 이용한다.

$$MSE = E\left[(\hat{\theta}_m - \theta)^2\right] = \text{Bias}(\hat{\theta}_m)^2 + \text{Var}(\hat{\theta}_m)$$

-위 식을 살펴보면, MSE는 편향과 분산 모두에 관여하므로 MSE를 줄이는 것은 편향과 분산 모두를 줄이는 것이 된다.

-아래는 편향과 분산의 관계에 대한 그래프이다.



Reducing the MSE for the point estimator

⇒ Maximizing Log Likelihood = Minimizing MSE

Example)

Assume the data is described by the linear model $y = wX + \epsilon$, where $\epsilon_i \sim N(\epsilon_i; 0, \sigma_\epsilon^2)$ and σ_ϵ^2 is known and the data points are $i.i.d.$.

Recall the likelihood is the probability of the data given the parameters of the model, in this case the weights on the features, w.

The log likelihood of our model is

$$\log p(y \mid X, w) = \sum_{i=1}^{N} \log p(y_i \mid x_i, \theta)$$

But since the noise ϵ is Gaussian (i.e. normally distributed), the likelihood is just

$$\begin{aligned}
\log p(y \mid X, w) &= \sum_{i=1}^{N} \log N(y_i \mid x_i w, \sigma^2) \\
&= \sum_{i=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma_\epsilon^2}} \exp\left(-\frac{(y_i - x_i w)^2}{2\sigma_\epsilon^2}\right) \\
&= -\frac{N}{2} \log 2\pi\sigma_\epsilon^2 - \sum_{i=1}^{N} \frac{(y_i - x_i w)^2}{2\sigma_\epsilon^2}
\end{aligned}$$

where N is the number of data points. Thus,

$$\begin{aligned}
w_{MLE} &= \operatorname{argmax}_w - \sum_{i=1}^{N} (y_i - x_i w)^2 \\
&= \operatorname{argmin}_w \frac{1}{N} \sum_{i=1}^{N} (y_i - x_i w)^2 \\
&= \operatorname{argmin}_w MSE_{train}
\end{aligned}$$

That is, the parameters w chosen to maximise the likelihood are exactly those chosen to minimise the mean-squared error.

■

Problem 9.

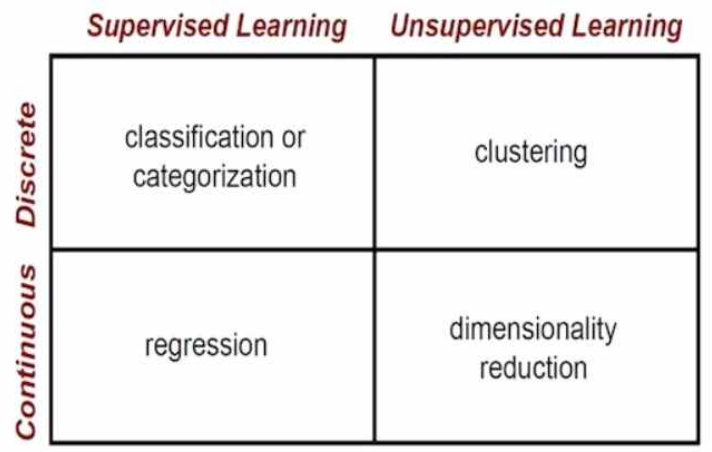
Compare supervised and unsupervised learning and give an example of each learning problems.

Solution)

Learning → Supervised vs Unsupervised and Reinforcement

Supervised : Labeled data / Direct feedback / Predict outcome or future.
Unsupervised : No labels / No feedback / "Find hidden structure"
Reinforcement : Decision process / Reward system / Learn series of actions

Example : Supervised vs Unsupervised



■

Problem 10.

Compare gradient descent and stochastic gradient descent. Discuss the usage/advantage of the stochastic gradient descent for the learning algorithms.

Solution)

-확률적 경사하강법(stochastic gradient descent)과 경사하강법(gradient descent) 두 알고리즘은 데이터에 대한 매개변수를 평가를 하고 값을 조정하면서 손실함수(Loss Function)을 최소화하는 값을 구하는 접근 방법 (미분 값(기울기)이 최소가 되는 점을 찾아 알맞은 weight(가중치 매개변수)를 찾아 냄)

-특히 함수가 어렵고 복잡하여 수학적 접근 방법으로 풀기 어려운 문제에도 잘 동작하지만, 경사 하강법은 모든 훈련 데이터에서 대해서 값을 평가하고 매개변수 업데이트를 진행하기 때문에 속도가 느림

-확률적 경사하강법은 확률적으로 선택한 데이터에 대해서 값을 평가하고 매개변수를 업데이트를 하기 때문에 경사 하강법에 비해서 빠른 속도를 보장함.

-경사하강법은 이론적으로 확률적 경사하강법보다 오류 기능을 최소화, 그러나 확률적 경사하강법은 데이터 집합이 커지면 훨씬 빠르게 값에 수렴함. 즉 작은 데이터 세트의 경우 경사 하강법이 바람직하지만 큰 데이터 세트의 경우 확률적 경사하강법이 더 좋음.

-실제로 확률적 경사하강법은 대부분의 응용 프로그램에 사용하는데, 왜냐하면 확률적 경사하강법은 오류 기능을 충분히 최소화하면서 대용량 데이터 세트에 더 빠르고 효율적으로 사용할 수 있기 때문임.

■