Overview of Supervised Learning

김지수 (Jisu KIM)

딥러닝의 통계적 이해 (Deep Learning: Statistical Perspective), 2024년 2학기

The lecture note is a minor modification of the lecture notes from Prof. Joong-Ho Won's "Deep Learning: Statistical Perspective" and Prof. Yongdai Kim's "Statistical Machine Learning".Also, see Section 2.1-2.6, 2.9 and Section 4 from [1].

1 Statistical Machine Learning

There are several fields in machine learning.

- Supervised Learning (지도학습)
  - Predict outcome variable (출력변수) using input variable (입력변수)
  - 회귀(regression), 분류(classification), etc
- Unsupervised Learning (비지도학습)
  - Clarify relations within input variables.
  - 군집분석(clustering), 주성분분석(principal component analysis), 요인분석(factor analysis), etc
- Reinforcement Learning (강화학습)
  - Learn decision making where environment changes according to actions
  - multi-armed bandit problem, markov decision process, etc

This class mostly covers supervised learning.

Data(자료) always comes with random noise(잡음). To see the effect of random noise and whether the algorithm works correctly, we need statistical analysis. In machine learning, statistical theory provides the following:

- understanding overfitting (과적합)
- understanding curse of dimensionality (차원의 저주)
- statistical theory for algorithm: consistency(일치성), optimality(최적합), uncertaintly quantification(불확실성의 정량화)

2 Basic Model for Supervised Learning

- Input(입력) / Covariate(설명 변수) : $x \in \mathbb{R}^p$, so $x = (x_1, \ldots, x_p)$.
- Output(출력) / Response(반응 변수): $y \in \mathcal{Y}$. If y is categorical, then supervised learning is "classification", and if y is continuous, then supervised learning is "regression".
- Model(모형) :
$$y \approx f(x), \qquad f \in \mathcal{M}.$$
  If we include the error ϵ to the model, then it can be also written as
$$y = \phi(f(x), \epsilon).$$
  For many cases, we assume additive noise, so
$$y = f(x) + \epsilon.$$

- Assumption(가정): f belongs to a family of functions $\mathcal{M}$. This is the assumption of a model: a model can be still used when the corresponding assumption is not satisfied in your data.
- Loss function(손실 함수): $\ell(y, a)$. A loss function measures the difference between estimated and true values for an instance of data.
- Training data(학습 자료): $\mathcal{T} = \{(y_i, x_i), i = 1, \ldots, n\}$, where (y_i, x_i) is a sample from a probability distribution P_i. For many cases we assume i.i.d., or x_i's are fixed and y_i's are i.i.d..
- Goal(목적): we want to find f that minimizes the expected prediction error,

  $$f^0 = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(Y,X)\sim P} \left[\ell(Y, f(X))\right].$$

  Here, $\mathcal{F}$ can be different from $\mathcal{M}$; $\mathcal{F}$ can be smaller then $\mathcal{M}$.
- Prediction model(예측 모형): f^0 is unknown, so we estimate f^0 by $\hat{f}$ using data. For many cases we minimizes on the empirical prediction error, that is taking the expectation on the empirical distribution $P_n = \frac{1}{n}\sum_{i=1}^n \delta_{(Y_i, X_i)}$.

  $$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{P_n} \left[\ell(Y, f(X))\right] = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \ell(Y_i, f(X_i)).$$
- Prediction(예측): if $\hat{f}$ is a predicted function, and x is a new input, then we predict unknown y by $\hat{f}(x)$.

3 Linear Regression Models and Nearest Neighbors

For the regression, we assume the additive noise model

$$y = f(x) + \epsilon,\ f \in \mathcal{M}.$$

Linear Regression Model (선형회귀모형) is to find the function f among

$$\mathcal{F} = \left\{ \beta_0 + \sum_{j=1}^p \beta_j x_j : \beta_j \in \mathbb{R} \right\}.$$

For estimating β, we use least squares: suppose the training data is $\{(y_i, x_{ij}) : 1 \le i \le n, 1 \le j \le p\}$. We use square loss

$$\ell(y, a) = (y - a)^2,$$

then the eimpirical loss becomes the residual sum of square (RSS) as

$$\begin{aligned} RSS(\beta) &= \sum_{i=1}^n (y_i - f(x_i))^2 \\ &= \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p x_{ij}\beta_j \right)^2. \end{aligned}$$

Let $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p)$ be the nimimizor of RSS, then the predicted function is

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{j=1}^p \hat{\beta}_j x_j.$$

k-Nearest Neighbor(k-NN, k-최근접 이웃) is to use those observations in the training set $\mathcal{T}$ closest in input space to x to form $\hat{f}(x)$. For an input $x \in \mathbb{R}^p$, let $N_k(x)$ be the neighborhood of x defined by the k closest points among the training sample $x_1, \ldots, x_n$. Then the predicted function for k-nearest neighbor is

$$\hat{f}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i.$$

4 Statistical Decision Theory

Let the training data $\mathcal{T} = \{(y_i, x_i), i = 1, \ldots, n\}$ be i.i.d. from P, or $y|X = x$ is i.i.d. (weaker). We seek a function $f(X)$ for predicting Y given values of the input X.

The Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$ is used for penalizing errors in prediction.

The Risk (위험) or Expected Prediction Error (평균예측오차) is the expected loss as

$$\begin{aligned} R(f) &= \mathbb{E}_{(Y,X)\sim P}\left[\ell(Y, f(X))\right] \\ &= \mathbb{E}_X\left[\mathbb{E}_{Y|X}\left[\ell(Y, f(X))\,|X\right]\right]. \end{aligned}$$

Since we are trying to find f as a function of X, it suffices to minimize conditional risk $\mathbb{E}_{Y|X}\left[\ell(Y, f(X))\,|X = x\right]$ pointwise. Hence the best predictor is

$$f^0(x) = \arg\min_{c\in\mathcal{Y}} \mathbb{E}_{Y|X}\left[\ell(Y, c)\,|X = x\right].$$

For the regression, the most common loss is the square loss

$$\ell(y, a) = (y - a)^2.$$

Then the conditional risk is factorized as

$$\begin{aligned} \mathbb{E}_{Y|X}\left[\ell(Y, f(X))\,|X\right] &= \mathbb{E}_{Y|X}\left[(Y - f(X))^2\,|X\right] \\ &= \mathbb{E}_{Y|X}\left[(Y - \mathbb{E}\left[Y|X\right])^2\,|X\right] + \left(\mathbb{E}\left[Y|X\right] - f(X)\right)^2, \end{aligned}$$

so $R(f)$ is minimized when $f^0(x) = \mathbb{E}\left[Y|X = x\right]$. This conditional expectation is called the regression function (회귀함수). Still, $\mathbb{E}\left[Y|X = x\right]$ is usually unknown.

The nearest neighbor directly implement this using the training data:

$$\hat{f}(x) = Ave\left(y_i|x_i \in N_k(x)\right),$$

where "Ave" denotes average, and $N_k(x)$ is the neighbor containing the k points in the training dataset $\mathcal{T}$ closest to x. Two approximations are happening here:

- expectation is approximated by averaging over sample data;
- conditioning at a point is relaxed to conditioning on some region close to the target point x, i.e., neighbor of x.

In fact, under mild regularity conditions on the joint probability distribution $P(X, Y)$, one can show that as $n, k \to \infty$ such that $k/n \to 0$, then [3]

$$\hat{f}(x) \xrightarrow{p} \mathbb{E}\left[Y|X = x\right].$$

The condition $k/n \to 0$ means that the model complexity k should grow slower than the sample size n.

The linear regression is a model-based approach. We assume that the regression function $f^0(x)$ is approximately linear. For notational convenience, write $x = (1, x_1, \ldots, x_p)^\top$ so that the constant is also included in x. Then we assume

$$\mathbb{E}\left[Y|X = x\right] \approx x^\top\beta = \sum_{j=0}^{p} \beta_j x_j.$$

This is a model-based approach - we specify a model for the regression function. We try to minimize risk among functions of the form $f(x) = x^\top\beta$. Let

$$\beta_* := \left(\mathbb{E}\left[XX^\top\right]\right)^{-1}\mathbb{E}\left[XY\right],$$

then the risk of f becomes

$$\mathbb{E}\left[(Y - X^\top\beta)^2\right] = \mathbb{E}\left[\left(Y - X^\top\beta_*\right)^2\right] + \mathbb{E}\left[\left(X^\top\beta_* - X^\top\beta\right)^2\right],$$

So β_* is minimizing the risk. And the least square $\hat{\beta} = (X^\top X)^{-1}X^\top Y$ can be thought as the average over empirical distribution of training data.

Neural networks are also model based, although far more flexible than the rigid linear model.

For the classification, assume $y \in \mathcal{Y} = \{1, \dots, K\}$. Then the risk becomes

$$\begin{aligned} R(f) &= \mathbb{E}_X \left[\mathbb{E}_{Y|X} \left[\ell(Y, f(X)) \,|X \right] \right] \\ &= \mathbb{E}_X \left[\sum_{k=1}^{K} \ell(k, f(X)) P(Y = k|X) \right]. \end{aligned}$$

Hence the conditional risk is minimized when

$$f^0(x) = \arg\min_{c \in \mathcal{Y}} \sum_{k=1}^{K} \ell(k, c) P(Y = k|X = x).$$

For the classification, the most common loss is the 0-1 loss

$$\ell(y, a) = I(y \neq a) = \begin{cases} 0, & \text{if } y = a, \\ 1, & \text{if } y \neq a. \end{cases}$$

With the 0-1 loss, the optimal prediction function is

$$\begin{aligned} f^0(x) &= \arg\min_{c \in \mathcal{Y}} \sum_{k=1}^{K} I(k \neq c) P(Y = k|X = x). \\ &= \arg\max_{k=1,\dots,K} P(Y = k|X = x). \end{aligned}$$

This optimal classifier is called Bayes rule / Bayes classifier (베이즈분류 / 베이즈모형), and the risk is called the Bayes risk (베이즈위험). Still, $P(Y = k|X = x)$ is usually unknown.

Again, k-NN can directly approximate Bayes rule. The conditional probability $P(Y = k|X = x)$ is approximated as

$$\frac{1}{k} \sum_{x_i \in N_k(x)} I(y_i = k),$$

so k-NN classifier becomes

$$\hat{f}(x) = \arg\max_{k=1,\dots,K} \left\{ \sum_{x_i \in N_k(x)} I(y_i = k) \right\}.$$

5 Curse of Dimensionality

Suppose $X = (X_1, \dots, X_p) \sim Uniform([0, 1]^p)$. Suppose we want to set a hypercubical neighbor of a target point $x \in [0, 1]^p$ to capture a fraction r of the observations. For this, the side length of the hypercube should be

$$e_p(r) = r^{1/p}.$$

$e_{10}(0.01) = 0.63$ and $e_{10}(0.1) = 0.80$, so to capture 1% or 10% of the data to form a local average, we must cover 63% or 80% of the range of each input variable. Such neighbor is no longer "local".

Consider $X = (X_1, \dots, X_p) \sim Uniform(\mathcal{B}_{\mathbb{R}^p}(0, 1))$, where $\mathcal{B}_{\mathbb{R}^p}(0, 1) = \{x \in \mathbb{R}^p : \|x\|_2 = 1\}$ is the unit ball in $\mathbb{R}^2$. For i-th data $X_i = (X_{i1}, \dots, X_{ip})$, $R_i \coloneqq \sqrt{\sum_{j=1}^{p} X_{ij}^2}$ is the distance of X_i to the origin, and $R_{(1)} = \min_{1 \leq i \leq n} R_i$ is the distance from the origin to the nearest neighbor. Since $R_{(1)}$ is also a random variable, then the median of $R_{(1)}$ is

$$Med(R_{(1)}) = \left(1 - (1/2)^{1/n}\right)^{1/p}.$$

If $n = 5000$ and $p = 10$, then $Med(R_{(1)}) \approx 0.52$, more than halfway to the boundary. Hence most data points are closer to the boundary of the sample space than to any other data point. This presents a problem since prediction is much more difficult near the edges of the training sample.

Beyond statistical difficulty for inference, high dimension is very counter-intuitive: reference https://marckhoury.github.io/blog
properties-of-high-dimensional-space/

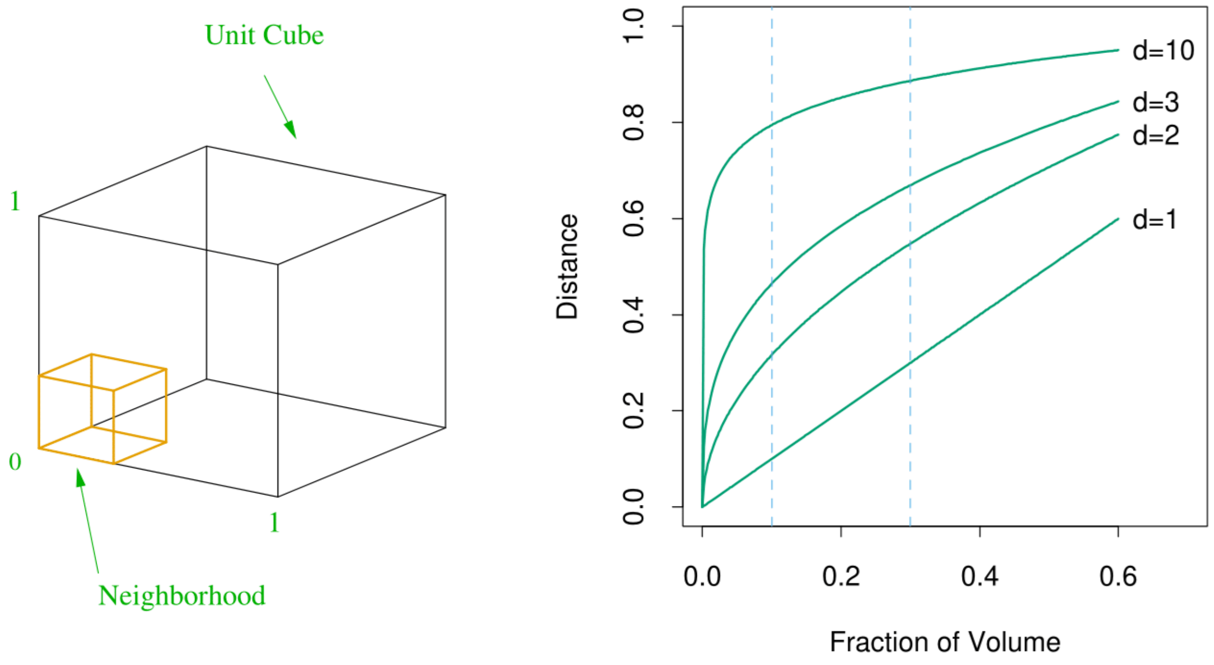


Figure 1: The curse of dimensionality is well illustrated by a subcubical neighborhood for uniform data in a unit cube. The figure on the right shows the side-length of the subcube needed to capture a fraction r of the volume of the data, for different dimensions p. In ten dimensions we need to cover 80% of the range of each coordinate to capture 10% of the data. Figure 2.6 from [1].

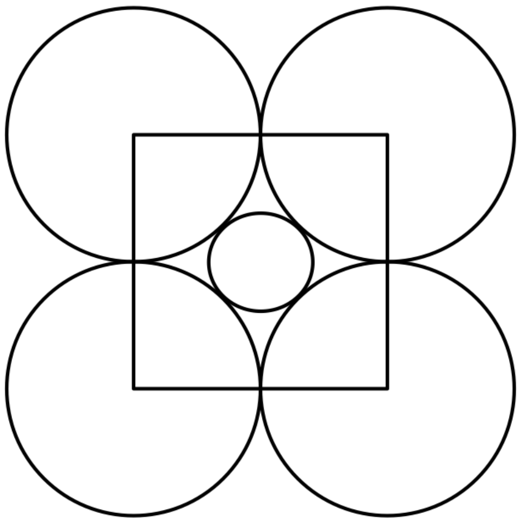


Figure 2: At each corner of the square we place a circle of radius 1/2. The inner circle is just large enough to touch the circles at the corners. Figure 1 from [2].

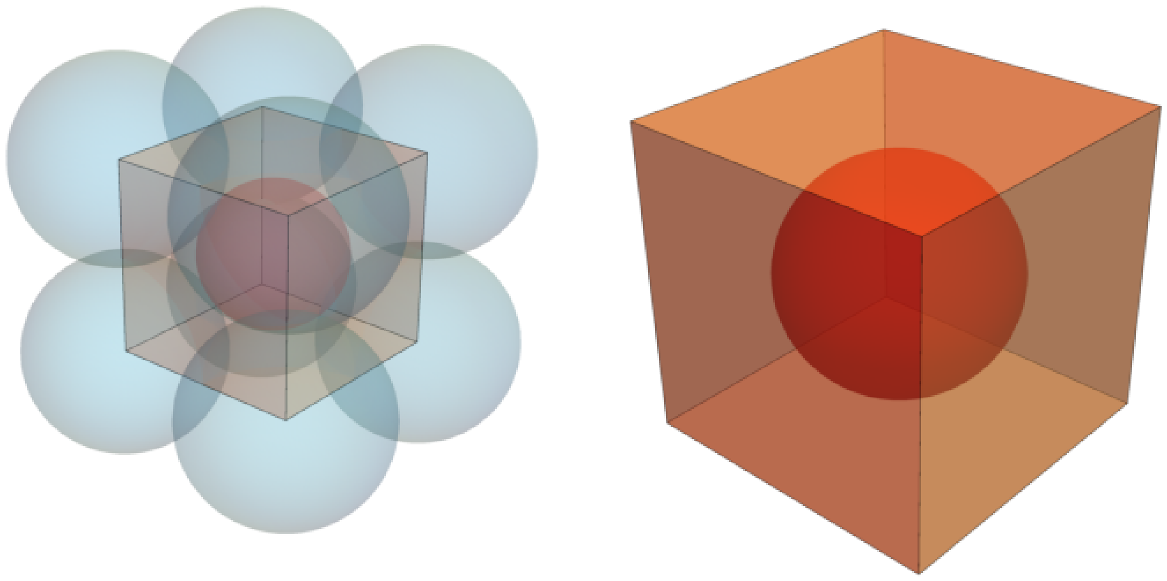


Figure 3: In three dimensions we place a sphere at the each of the eight corners of a cube. Figure 2 from [2].

Consider a square with side length 1. At each corner of the square place a circle of radius 1/2, so that the circles cover the edges of the square. Then consider the circle centered at the center of the square that is just large enough to touch the circles at the corners of the square. In two dimensions it's clear that the inner circle is entirely contained in the square.

We can do the same thing in three dimensions. At each corner of the unit cube place a sphere of radius 1/2, again covering the edges of the cube. The sphere centered at the center of the cube and tangent to spheres at the corners of the cube is shown in red in Figure 3. Again we see that, in three dimensions, the inner sphere is entirely contained in the cube.

To understand what happens in higher dimensions we need to compute the radius of the inner sphere in terms of the dimension.

(the radius of the inner sphere) = (the length of the half diagonal of the cube)−(the radius of the spheres at the corners).

The radius of the inner sphere is equal to the length of the diagonal of the cube minus the radius of the spheres at the corners. See Figure . The half diagonal length is $\sqrt{d}/2$, and the latter value is always 1/2, and hence the radius of the inner sphere is

$$\frac{\sqrt{d}-1}{2}.$$

When $d \leq 3$, $\frac{\sqrt{d}-1}{2} < \frac{1}{2}$ and the sphere is strictly inside the cube. However in four dimensions, the radius of the inner sphere is $\frac{\sqrt{d}-1}{2} = \frac{1}{2}$, so the inner sphere touches the sides of the cube. In five dimensions, the radius of the inner sphere is 0.618034, and the sphere starts poking outside of the cube! By ten dimensions, the radius is 1.08114 and the sphere is poking very far outside of the cube!

6 Overfitting and the Bias-Variance Tradeoff

In k-Nearest Neighbor, the neighbor size k determines the complexity of the prediction model. To choose k, if we know the probability distribution $(Y, X) \sim P$, then we can minimize the risk (위험)

$$\mathbb{E}_{(Y,X)\sim P}\left[\ell(Y, f(X))\right].$$

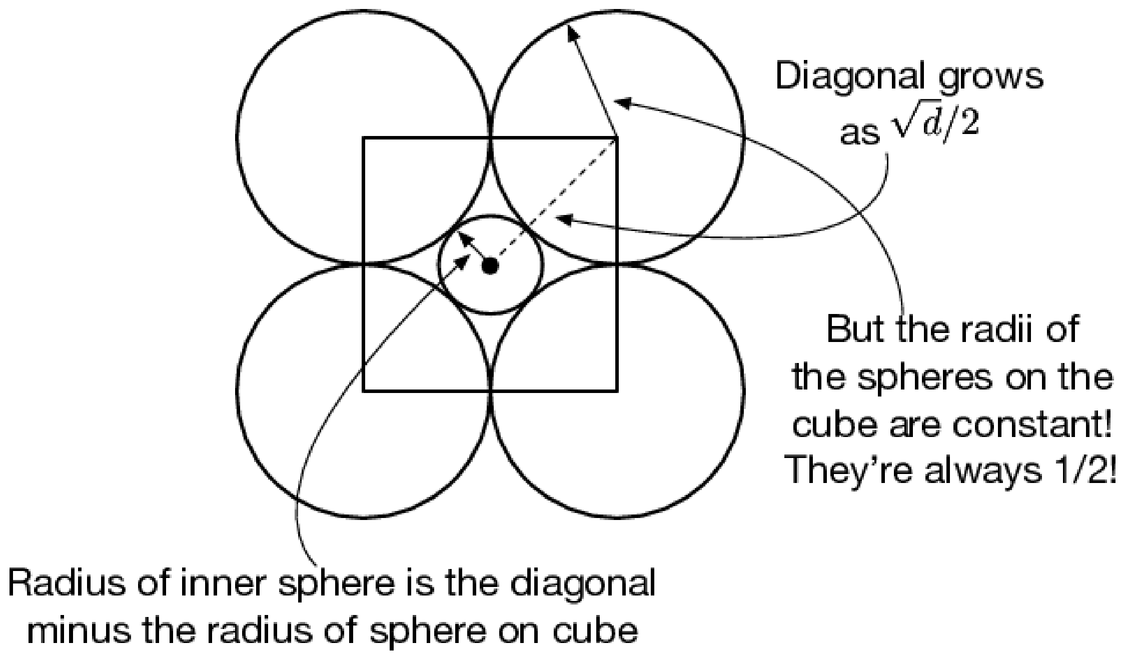

Diagonal grows as $\sqrt{d}/2$
But the radii of the spheres on the cube are constant! They're always 1/2!
Radius of inner sphere is the diagonal minus the radius of sphere on cube

Figure 4: The size of the radius of the inner sphere is growing as the dimension increases because the distance to the corner increases while the radius of the corner sphere remains constant. Figure 3 from [2].

However, P is usually unknown. One way to estimate the risk is by Empirical Risk (경험 위험) or Training Error (학습오차)

$$\frac{1}{n}\sum_{i=1}^{n}\ell(Y_i, f(X_i)),$$

where the training data $\mathcal{T} = \{(Y_i, X_i) : i = 1, \dots, n\}$.

However, training error always underestimates prediction error, since the same dataset $\mathcal{T}$ is used for estimating the prediction model and estimating prediction error. Better estimation for the prediction error is the test error, where the prediction error is computed on the separate test data. The typical behavior of the prediction error on the test set is as in Figure 5. As the model becomes more complex, the prediction error first decreases but then increases, while test error monotonically decreases. Hence, when the model is chosen according to the training error, then the prediction error can be large: this is called overfitting. This phenomenon is explained by the bias-variance tradeoff.

Suppose the regression model

$$Y = f(X) + \epsilon,$$

with $\mathbb{E}[\epsilon] = 0$ and $Var[\epsilon] = \sigma^2$, and we use the square loss $\ell(y, a) = (y - a)^2$. Let $f(X, \mathcal{T})$ be the predicted function trained on the training set $\mathcal{T}$. Then the risk for $f(X, \mathcal{T})$ is

$$\begin{aligned}
R(f) &= \mathbb{E}\left[(Y - f(X, \mathcal{T}))^2\right] \\
&= \mathbb{E}\left[(Y - f(X))^2\right] + \mathbb{E}\left[(f(X) - f(X, \mathcal{T}))^2\right] \\
&= \sigma^2 + \mathbb{E}_X\left[(f(X) - \mathbb{E}_{\mathcal{T}}[f(X, \mathcal{T})|X])^2\right] + \mathbb{E}_X\left[\mathbb{E}_{\mathcal{T}}\left[(f(X, \mathcal{T}) - \mathbb{E}_{\mathcal{T}}[f(X, \mathcal{T})|X])^2 \,|X\right]\right] \\
&= \sigma^2 + \mathbb{E}_X\left[Bias_{\mathcal{T}}(X)^2 + Variance_{\mathcal{T}}(X)\right].
\end{aligned}$$

Then

$$\mathbb{E}_{Y|X}\left[(Y - f(X))^2|X\right] = \mathbb{E}_{Y|X}\left[(Y - \mathbb{E}[Y|X])^2\,|X\right] + (\mathbb{E}[Y|X] - f(X))^2,$$

In general, as the model becomes more complex, then the bias(편의) decreases and the variance(분산) increases.

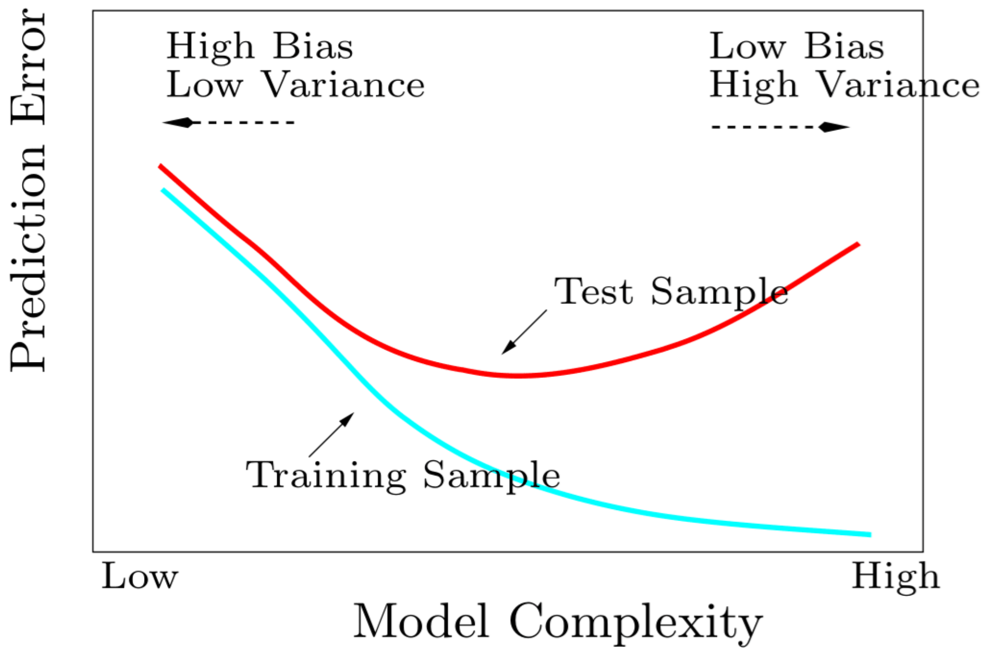


Figure 5: Test and training error as a function of model complexity. Figure 2.11 from [1].

For k-Nearest Neighbor,

$$\hat{f}(x) = f(x, \mathcal{T}) = \frac{1}{k}\sum_{l=1}^{k}(f(x_{(l)}) + \epsilon_{(l)}),$$

where (l) is the index for l-th nearest neighbor. Then the bias and variance becomes

$$Bias_{\mathcal{T}}(x) = f(x) - \frac{1}{k}\sum_{l=1}^{k} f(x_{(l)}),$$

and

$$Variance_{\mathcal{T}}(x) = \frac{\sigma^2}{k}.$$

When $k = 1$, then the bias is minimized and the variance is maximized. On the other hand, when $k = n$, then the bias is maximized and the variance is minimized.

7 Comparison of Linear Regression with Nearest Neighbors

Below is the code for comparing linear regression with nearest neighbors. We measure the performance by MSE (Mean Squared Error)

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - f(X_i))^2,$$

either on training data or test data.

```
require(caret)

## Loading required package: caret
## Warning: package 'caret' was built under R version 4.3.3
## Loading required package: ggplot2
## Loading required package: lattice
```

```r
mse <- function(y_pred, y_true) {

  return (mean((y_pred - y_true)^2))
}


linear_knn_compare <- function(data_train, data_test) {

  par(mfrow = c(2, 2))

  model_lr <- lm(y ~ x, data = data_train)
  plot(data_train[["x"]], data_train[["y"]], main = "Linear Regression",
       xlab = "x", ylab = "y", pch = 16)
  abline(model_lr)
  cat(c("MSE of Linear Regression on Training:",
        mse(data_train[["y"]], predict(model_lr, newdata = data_train)), "\n"))
  cat(c("MSE of Linear Regression on Test:",
        mse(data_test[["y"]], predict(model_lr, newdata = data_test)), "\n"))

  for (k in c(1, 5, 15)) {
    model_knn <- caret::knnreg(y ~ x, data = data_train, k = k)
    plot(data_train[["x"]], data_train[["y"]],
         main = paste("Nearest Neighbor with k =", k), xlab = "x", ylab = "y",
         pch = 16)
    lines(data_train[["x"]], predict(model_knn, newdata = data_train))
    cat(c("MSE of Nearest Neighbor with k =", k, "on Training:",
          mse(data_train[["y"]], predict(model_knn, newdata = data_train)),
          "\n"))
    cat(c("MSE of Nearest Neighbor with k =", k, "on Test:",
          mse(data_test[["y"]], predict(model_knn, newdata = data_test)), "\n"))
  }
}
```

Suppose we have 100 training data and 900 test data, and $\epsilon \sim N(0,1)$.

```r
set.seed(0)
x <- rnorm(1000)
ep <- rnorm(1000)
train <- sort(x[1:100], index.return = TRUE)[["ix"]]
test <- 101:1000
```

First, we consider the simulation

$$y = x + \epsilon.$$

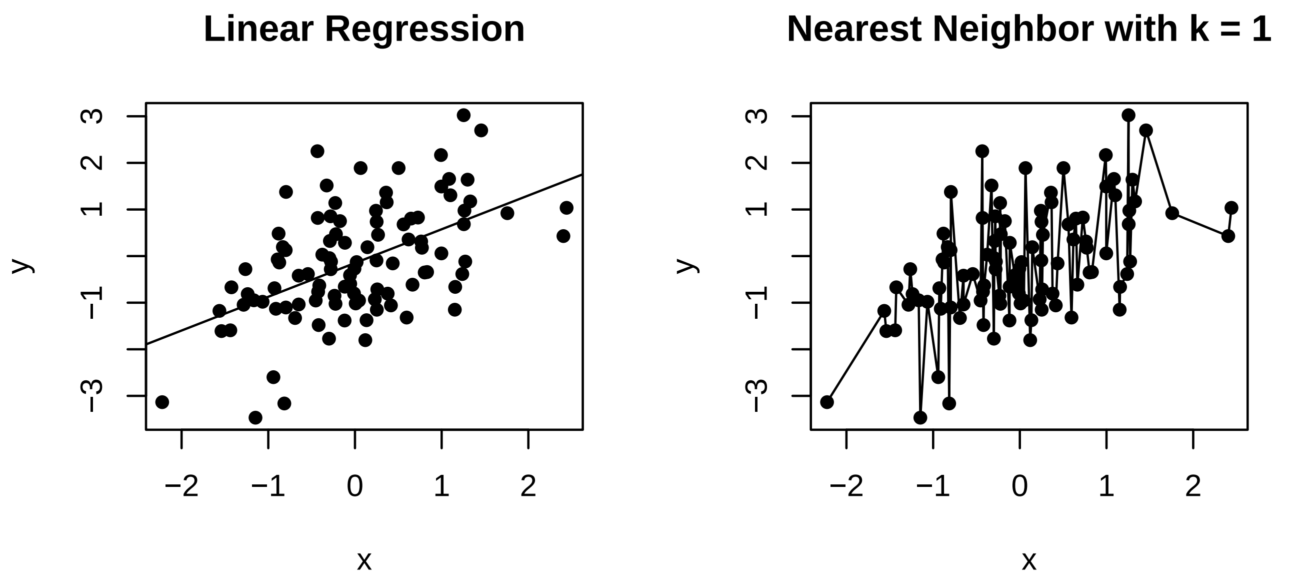
The result is as follows:

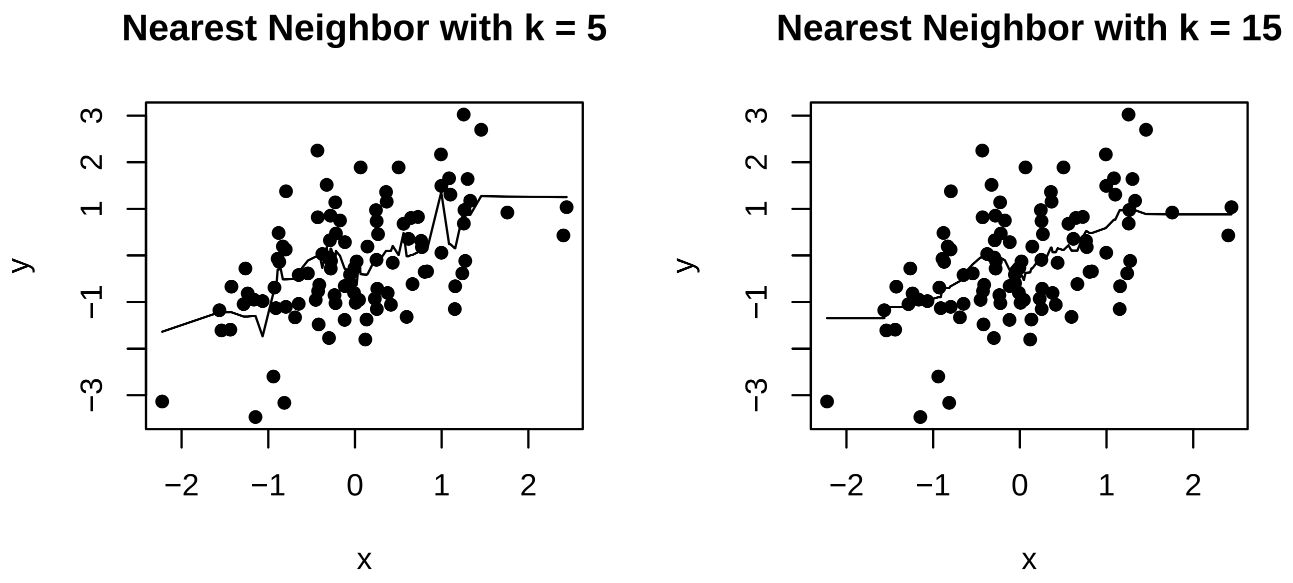
Method	MSE on Training	MSE on Test
Linear	1.0447	1.1621
1-NN	0	2.0972
5-NN	0.9381	1.2779
15-NN	1.0053	1.2757

```r
# y = x + ep
y_l <- x + ep
data_l_train <- list(x = x[train], y = y_l[train])
data_l_test <- list(x = x[test], y = y_l[test])

linear_knn_compare(data_l_train, data_l_test)

## MSE of Linear Regression on Training: 1.04471978605025
## MSE of Linear Regression on Test: 1.16212381385859
## MSE of Nearest Neighbor with k = 1 on Training: 0
## MSE of Nearest Neighbor with k = 1 on Test: 2.09716012610056
## MSE of Nearest Neighbor with k = 5 on Training: 0.938067727758741
## MSE of Nearest Neighbor with k = 5 on Test: 1.27788806945154
## MSE of Nearest Neighbor with k = 15 on Training: 1.00528738504805
## MSE of Nearest Neighbor with k = 15 on Test: 1.27565679650095
```





Second, we consider the simulation

$$y = x(1 - x) + \epsilon.$$

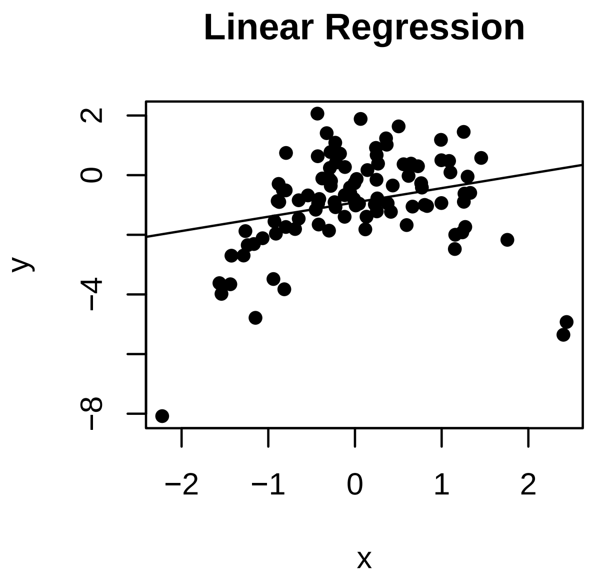
The result is as follows:

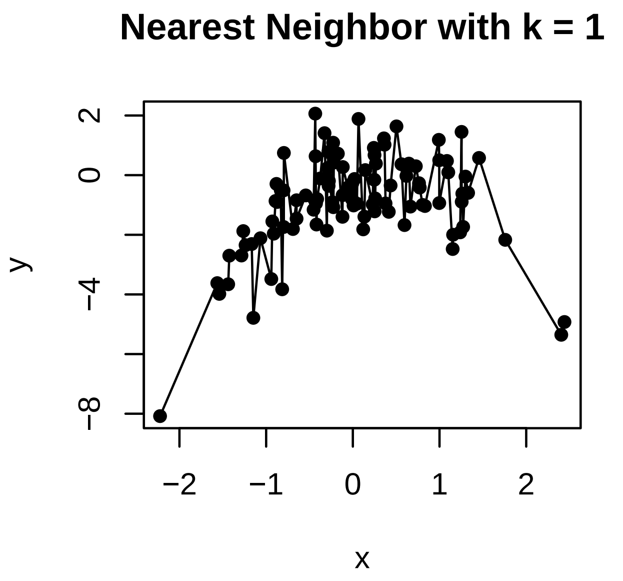
Method	MSE on Training	MSE on Test
Linear	2.4477	3.2330
1-NN	0	2.2437
5-NN	1.1955	1.7266
15-NN	1.5284	2.0868

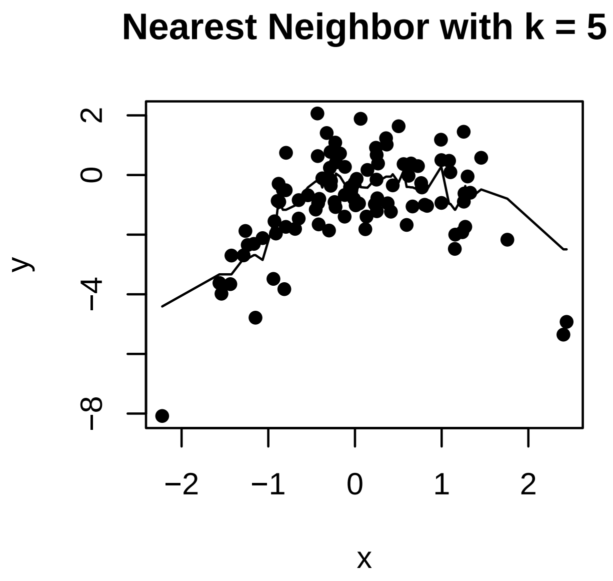
```r
# y = x(1-x) + ep
y_q <- x * (1 - x) + ep
data_q_train <- list(x = x[train], y = y_q[train])
data_q_test <- list(x = x[test], y = y_q[test])
```

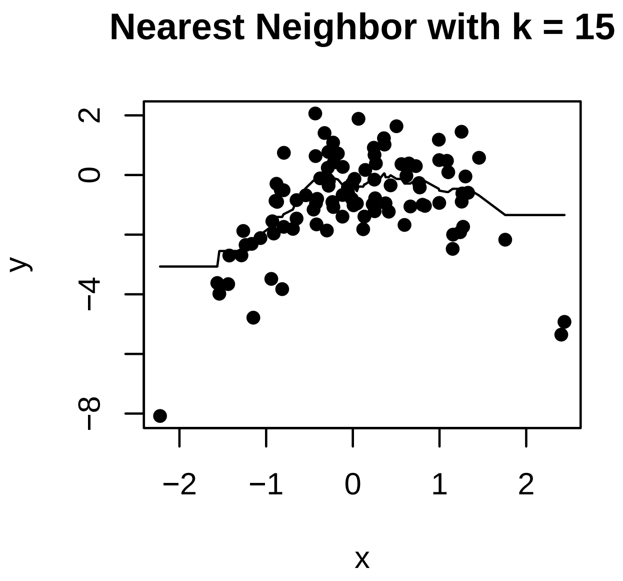
```r
linear_knn_compare(data_q_train, data_q_test)

## MSE of Linear Regression on Training: 2.44774751766011
## MSE of Linear Regression on Test: 3.23304666428842
## MSE of Nearest Neighbor with k = 1 on Training: 0
## MSE of Nearest Neighbor with k = 1 on Test: 2.24369404278609
## MSE of Nearest Neighbor with k = 5 on Training: 1.19552889833504
## MSE of Nearest Neighbor with k = 5 on Test: 1.72664570974827
## MSE of Nearest Neighbor with k = 15 on Training: 1.52842407126519
## MSE of Nearest Neighbor with k = 15 on Test: 2.08680791517323
```









From the simulations, we can compare Linear Regression vs Nearest Neighbor as follows:

Method	Linear Regression	Nearest Neighbor
Assumption	Linear	X
Required Data Size (Relative)	Small	Large
Interpretation	Easy	Impossible
Prediction	Good if Linear is satisfied, Bad otherwise	works regardless of assumptions
Tuning Parameter	X	neighborhood size k

Also, we can see that:

- training error is not a good estimate of prediction error.
- As model complexity grows, we can see bias-variance tradeoff
- Neighborhood size k determines the model complexity of Nearest Neighbor

8 Logistic Regression / Classification (로지스틱 회귀 / 분류)

The logistic model assumes that for $1 \le k \le K-1$,

$$\Pr(y=k|x) = \frac{\exp(\beta_{k0} + x^\top \beta_k)}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + x^\top \beta_l)},$$

and

$$\Pr(y=K|x) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + x^\top \beta_l)}.$$

In other words, $\beta_{K0} = 0$ and $\beta_K = 0$, which is for identifiablity problem. The Bayes classifier is

$$G(x) = \arg \max_{k=1,\ldots,K} P(y=k|x).$$

Now, the decision boundary for i and j is a subset of where $\Pr(y=i|x) = \Pr(y=j|x)$, and this is a hyperplane defined as

$$\left\{ x \in \mathbb{R}^p : (\beta_{i0} - \beta_{j0}) + x^\top (\beta_i - \beta_j) = 0 \right\}.$$

Hence logistic classification is a linear classifier.

There are several ways to motivate logistic regression: for simplicity let $\mathcal{Y} = \{1,2\}$.

1. Consider a linear regression

$$\Pr(y=1|x) = \beta_0 + x^\top \beta.$$

It violates that the constraint

$$\Pr(y=1|x) \in [0,1].$$

A simple remedy for this problem is to set

$$\Pr(y=1|x) = F(\beta_0 + x^\top \beta),$$

where F is a continuous and strictly increasing function with $F(-\infty) = 0$ and $F(\infty) = 1$. Some examples for F are:

- Gaussian: Probit model
- Gompertz: $F(x) = \exp(-\exp(x))$, popularly used in Insurance
- Logistic: $F(x) = \exp(x)/(1+\exp(x))$.

2. Consider the decision boundary

$$\{x : \Pr(Y=1|X=x) = 0.5\}.$$

This is equivalent to

$$\{x : \log(\Pr(Y=1|X=x)/\Pr(Y=2|X=x)) = 0\}.$$

Suppose that the log-odds is linear. That is,

$$\log(\Pr(Y=1|X=x)/\Pr(Y=2|X=x)) = \beta_0 + x^\top \beta,$$

and this implies that

$$\Pr(Y = 1|X = x) = \frac{\exp(\beta_0 + x^\top \beta)}{1 + \exp(\beta_0 + x^\top \beta)}.$$

For the estimation, we use the maximum likelihood approach. The likelihood is simply the probability of the observations given as

$$L(\beta_0, \beta) = \prod_{i=1}^{n} \Pr(y = y_i|x = x_i),$$

and we estimate β by maximizing the log-likelihood. For two-class case where $\mathcal{Y} = \{1, 2\}$, the log-likelihood is simplified as

$$l(\beta_{10}, \beta_1) = \sum_{i=1}^{n} \left(I(y_i = 1)(\beta_{10} + x_i^\top \beta_1) - \log(1 + \exp(\beta_{10} + x_i^\top \beta_1)) \right).$$

One obstacle of using the logistic regression would be computation since maximizing the log-likelihood is not easy. We do it by using the Iteratively Reweighted Least Squares (IRLS) algorithm.

9 Linear Discriminant Analysis (선형판별분석)

Let $f_k(x)$ is the class conditional density of x in class $y = k$ where $y \in \mathcal{Y} = \{1, \ldots, K\}$, i.e.,

$$f_j(x) = p(x|y = k), \qquad k = 1, \ldots, K.$$

Let

$$\pi_k = \Pr(y = k), \qquad k = 1, \ldots, K,$$

be the prior probabilities, with $\sum_{k=1}^{K} \pi_k = 1$. Recall that the Bayes classifier is

$$G(x) = \arg \max_{k=1,\ldots,K} P(Y = k|X = x).$$

Then Bayes theorem gives us

$$P(Y = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}.$$

To see this, assume X is discrete for simplicity, then

$$P(Y = k, X = x) = P(Y = k|X = x)P(X = x) = P(Y = k|X = x)f_k(x)$$

And hence the Bayes classifier is given as

$$G(x) = \arg \max_{k=1,\ldots,K} P(Y = k|X = x) = \arg \max_{k=1,\ldots,K} f_k(x)\pi_k.$$

Suppose that we model each class density as multivariate Gaussian

$$f_k(x) = \frac{1}{(2\pi)^{p/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1}(x - \mu_k) \right),$$

where μ_k is the mean vector and Σ_k is the covariance matrix. Then

$$\log(f_k(x)\pi_k) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1}(x - \mu_k) + \log \pi_k - \frac{p}{2}\log(2\pi).$$

Hence by letting

$$\delta_k(x) := -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1}(x - \mu_k) + \log \pi_k,$$

then the Bayes classifier is given as

$$G(x) = k \text{ such that } \delta_k(x) > \delta_l(x) \text{ for all } l \neq k.$$

We call the functions $\delta_j(x)$ the discriminant functions.

For Linear Discriminant Analysis (LDA), we assume that

$$\Sigma_k = \Sigma \text{ for all } k.$$

In this case, we can see that

$$\begin{aligned}\log \frac{P(Y=k|X=x)}{P(Y=l|X=x)} &= \log \frac{f_k(x)}{f_l(x)} + \log \frac{\pi_k}{\pi_l} \\ &= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k+\mu_l)^\top \Sigma(\mu_k-\mu_l) + x^\top \Sigma(\mu_k-\mu_l).\end{aligned}$$

Hence the Bayes classifier is given as

$$G(x) = k \text{ such that } \delta_k(x) > \delta_l(x) \text{ for all } l \neq k,$$

where

$$\delta_k(x) := x^\top \Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^\top \Sigma \mu_k + \log \pi_k.$$

That is, the Bayes classifier is a linear classifier. The functions δ_k are called the linear discriminant functions.

If we don't assume $\Sigma_k = \Sigma$ for all k, then the decision boundary of the Bayes classifier is a quadratic function. This is called Quadratic Discriminant Analysis (QDA).

We can easily estimate μ_j and Σ_j by

- $\hat{\pi}_j = n_j/n$, where $n_j = \sum_{i=1}^n I(y_i = j)$.
- $\hat{\mu}_j = \sum_{i=1}^n x_i I(y_i = j)/n_j$.
- $\hat{\Sigma}_j = \sum_{i=1}^n (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^\top I(y_i = j)/(n_j - 1)$.

And we estimate Σ by the pooled variance-covariance matrix

$$\hat{\Sigma} = \frac{1}{n-K}\sum_{j=1}^K (n_j - 1)\hat{\Sigma}_j.$$

10 LDA or Logistic Classification

- Logistic classification and LDA both have linear decision boundaries.
- Logistic classification only needs the specification of the conditional distribution $\Pr(Y=k|X=x)$, that is, $\Pr(X=x)$ is completely undetermined. On the other hand, the LDA needs the specification of the joint distribution $\Pr(Y,X)$. In fact, in LDA, the marginal distribution of x is a mixture of Gaussians

$$\Pr(x) = \sum_{k=1}^K \pi_k N(\mu_k, \Sigma).$$

  Hence, LDA needs more assumptions and hence less applicability than the logistic regression.
- Categorical input variables are allowable for the logistic regression (using dummy variables) while LDA has troubles with such inputs.
- However, LDA is a useful tool when some of the output are missing (semi-supervised learning).
- LDA is useful when Gaussian assumptions are reasonable.
- LDA works better for multi-class problems ($K > 2$).
- In practice, for a two-class problem, logistic classification and LDA are often very similar.

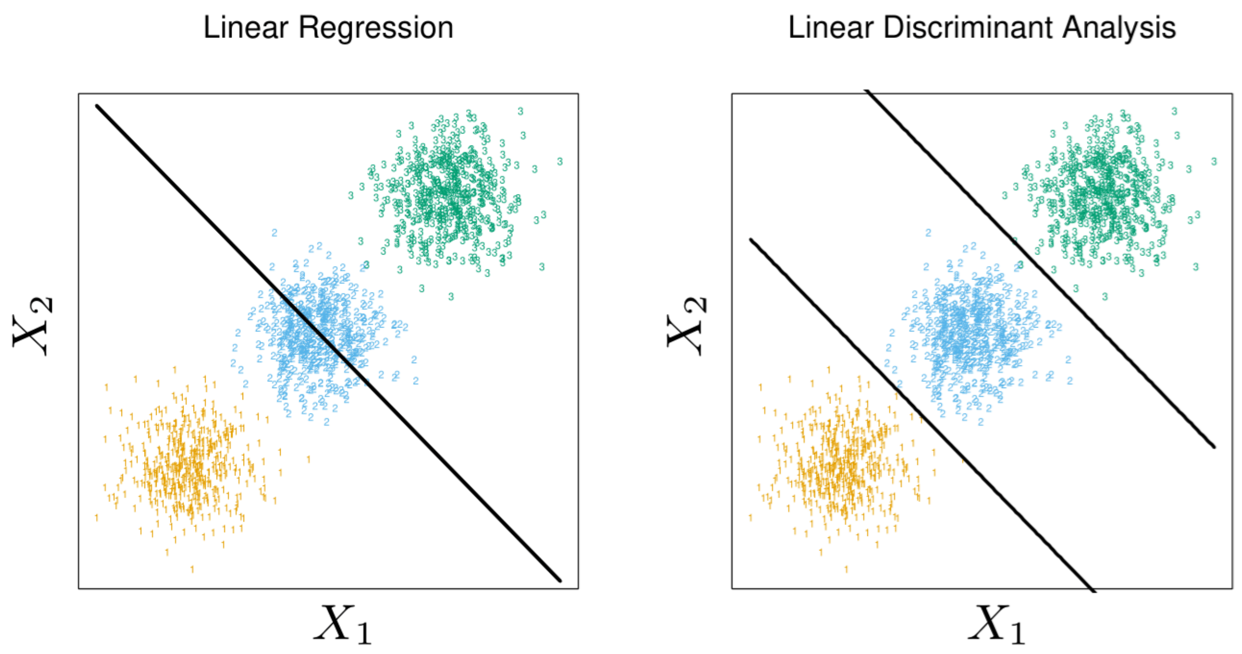


Figure 6: The data come from three classes in $\mathbb{R}^2$ and are easily separated by linear decision boundaries. The right plot shows the boundaries found by linear discriminant analysis. The left plot shows the boundaries found by linear regression of the indicator response variables. The middle class is completely masked (never dominates). Figure 4.2 from [1].

10.1 Multi-class problems with regression approach

There is a serious problem with the regression approach when the number of classes $K \geq 3$, especially prevalent when K is large. Because of the rigid nature of the regression model, classes can be masked by others. Figure 6 illustrates an extreme situation when $K = 3$. The three classes are perfectly separated by linear decision boundaries, yet linear regression misses the middle class completely.

In Figure 7 we have projected the data onto the line joining the three centroids (there is no information in the orthogonal direction in this case), and we have included and coded the three response variables Y_1, Y_2, and Y_3. The three regression lines (left panel) are included, and we see that the line corresponding to the middle class is horizontal and its fitted values are never dominant! Thus, observations from class 2 are classified either as class 1 or class 3. The right panel uses quadratic regression rather than linear regression. For this simple example a quadratic rather than linear fit (for the middle class at least) would solve the problem, but in general, if $K \geq 3$ classes are lined up, polynomial terms up to degree $K - 1$ might be needed to resolve them.

Note: masking problem is severe in ordinary regression (that is, regress Y on X), but it is also present in logistic regression as well. For tackling masking problem, LDA is better than logistic classification.

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning.* Springer Series in Statistics. Springer, New York, second edition, 2009. Data mining, inference, and prediction.

[2] Marc Khoury. Counterintuitive properties of high dimensional space, 2018.

[3] Charles J. Stone. Consistent nonparametric regression. *Ann. Statist.*, 5(4):595–645, 1977. With discussion and a reply by the author.

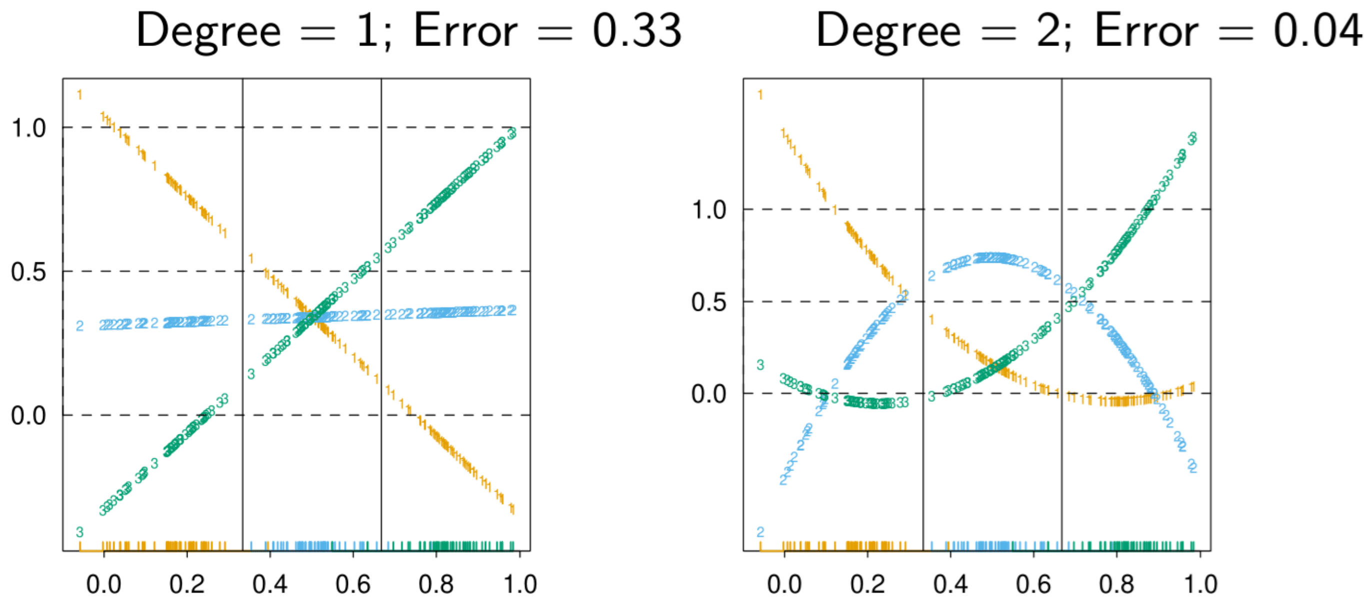


Figure 7: The effects of masking on linear regression in IR for a three-class problem. The rug plot at the base indicates the positions and class membership of each observation. The three curves in each panel are the fitted regressions to the three-class indicator variables; for example, for the blue class, yblue is 1 for the blue observations, and 0 for the green and orange. The fits are linear and quadratic polynomials. Above each plot is the training error rate. The Bayes error rate is 0.025 for this problem, as is the LDA error rate. Figure 4.3 from [1].