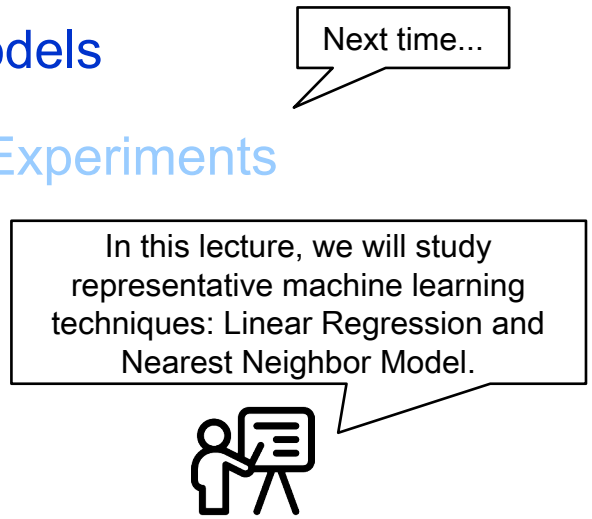
Linear Regression and *k*-NN (Part A)

Theory

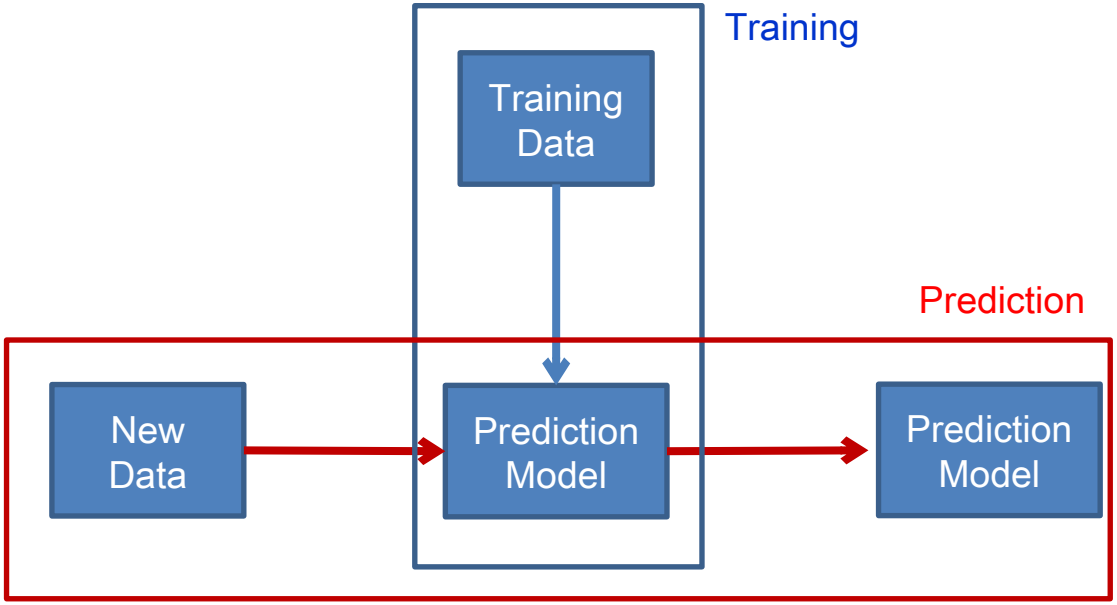
Contents

- Understadning and Implementing Representative Machine Learning Techniques
  - Univariate Linear Regression
  - Multivariate Linear Regression
  - Nearest Neighbor Models
  - Implementation and Experiments



INTRO

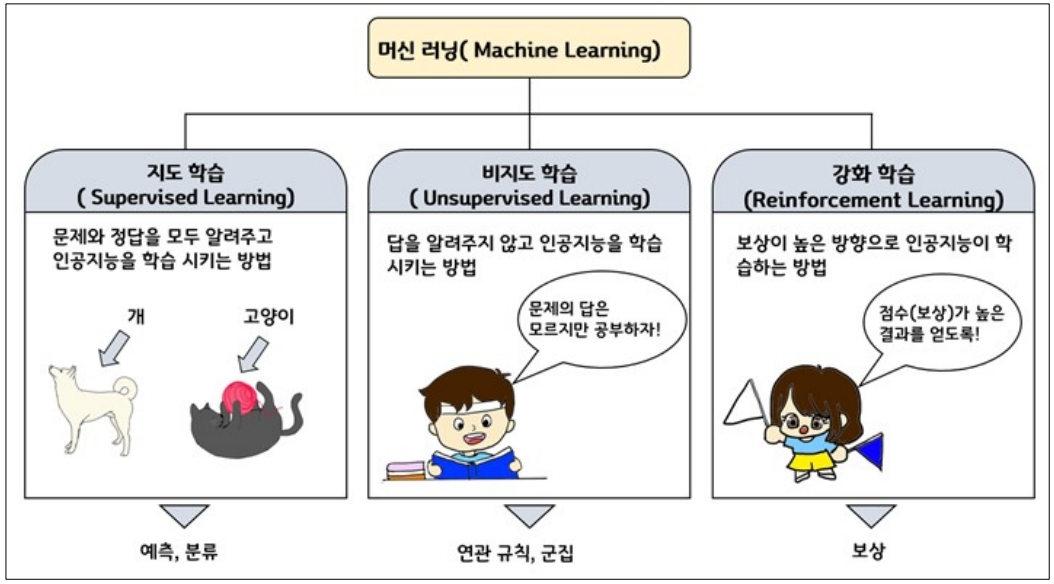
- Machine Learning (기계 학습 , 머신 러닝)
  - A field of technology that develops algorithms and statistical models used by computer systems to perform pattern recognition and inference, and to perform tasks on their own without explicit instructions
  - Computer systems use machine learning algorithms to process large amounts of data and identify patterns.
    => Through this, accurate results can be predicted.
  - In other words, Machine Learning is a technology that recognizes patterns autonomously through **learning** and performs accurate **prediction** through **inference**.



https://aws.amazon.com/ko/what-is/machine-learning/

INTRO

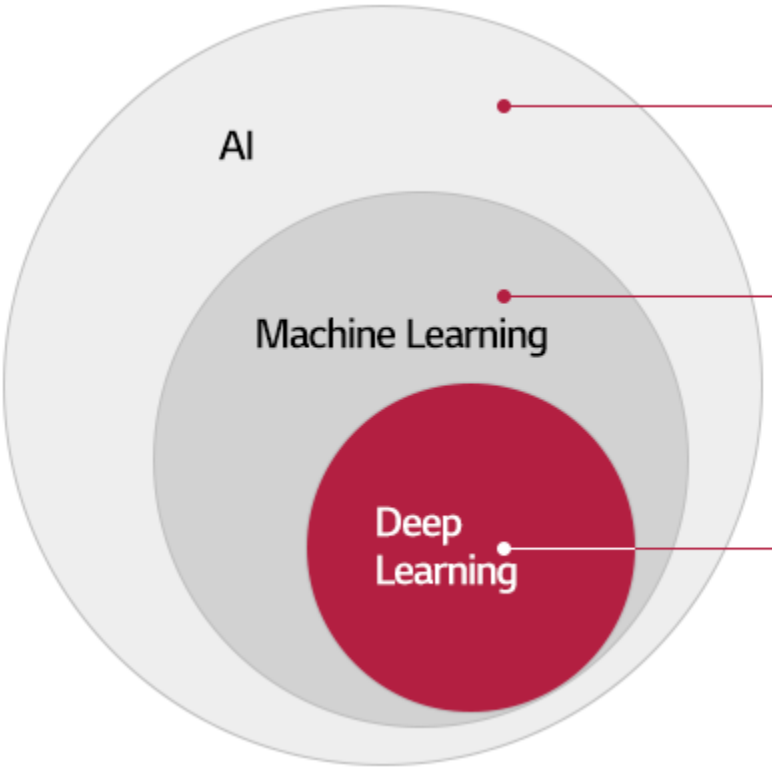
- Classification of Machine Learning
  - Machine learning is classified into **Supervised Learning**, **Unsupervised Learning**, and **Reinforcement Learning**.
    - **Supervised Learning**: A method of training a machine to find patterns between data and answers using multiple (**data**, **answers**) as input, so that it can correctly predict the answer when new data is given.
    - **Unsupervised Learning**: A method of training a machine to find patterns and correlations between data using multiple (**data**) as input.
    - **Reinforcement Learning**: A method of training a machine using continuous (**actions**, **rewards**) as input, so that it autonomously finds the actions that lead to high rewards.



http://ai4school.org/?page_id=2617

INTRO

- AI(인공지능), ML(기계학습), DL(딥러닝) Relationships



인공지능(Artificial Intelligence)

Pertains to all automation where a machine can perform tasks that a human should do.

머신러닝(Machine Learning)

The machine autonomously learns patterns for decision-making from data without explicit programming of rules.

딥러닝(Deep Learning)

A model based on artificial neural networks, where the machine performs the entire process—from feature extraction to judgment—from unstructured data in a single step.

UNIVARIATE LINEAR RE-GRESSION

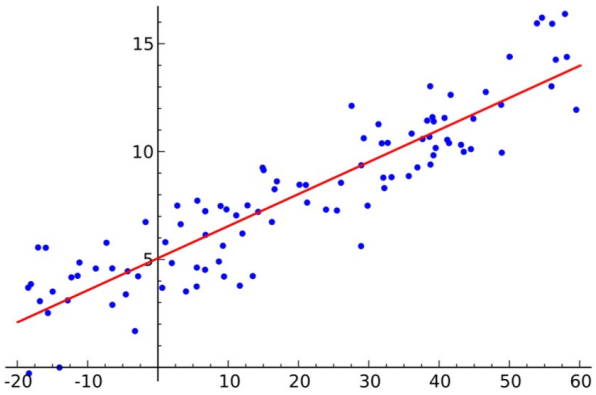
Regression

- The problem to solve
  - Input: Multiple pairs of (x_i, y_i), $i = 1, 2, 3, 4, \ldots, N$ (N = number of sample data)
    - x_i : data (features)
    - y_i : Answer/Label for the given data(x_i)
    - ex: Room/Bathroom area (= feature) and monthly rent (= label)
      x_1 = 30 m^2, y_1 = 30 만원,
      x_2 = 45 m^2, y_2 = 40 만원,
      x_3 = 60 m^2, y_3 = 60 만원, ... In this situation with N=3 sample data, given
      x_new = 52 m^2 what is the monthly rent y_new = ?
  - **Goal:** To develop a machine learning model that understands the relationship between data and answers from multiple (**data**, **answer**) samples, and predicts the answer when new data is given!

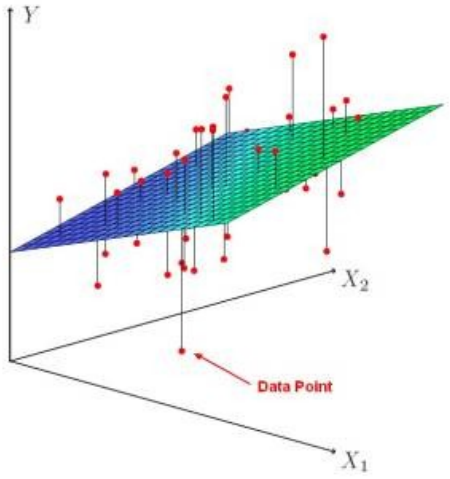
- Approach: Use the Regression (회귀) technique
  - Regression (회귀)
    - A technique for modeling the correlation $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{y}$ between one or more **independent variables** $\boldsymbol{x}$ and a single **dependent variable** $\boldsymbol{y}$ (i.e., a method for calculating what the function $\boldsymbol{h}$ should be).
    - A method to understand the influence of the **independent variable(s)** on the **dependent variable** and, based on this, to generate a model that predicts the value of the **dependent variable** corresponding to a given value of the **independent variable(s)**.

Linear Regression

- Linear Regression (선형 회귀)
  - Regression
    - A technique for modeling the correlation $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{y}$ between one or more **independent variables** $\boldsymbol{x}$ and a single **dependent variable** $\boldsymbol{y}$
  - Linear Regression
    - A method for finding the function $\boldsymbol{h}$ by assuming it is a **linear function**.
    - If the **independent variable** is a scalar, $\boldsymbol{h}$ is a **straight line graph**, and if the **independent variables** are multiple (a vector), $\boldsymbol{h}$ becomes a **plane in a multidimensional space**.
    - The case where the **independent variable** $(\boldsymbol{x_i})$ is a scalar is called the **Univariate Linear Regression** model,
      and the case where the **independent variable** $(\boldsymbol{x_i})$ is a vector composed of multiple values is called the **Multivariate Linear Regression** model.



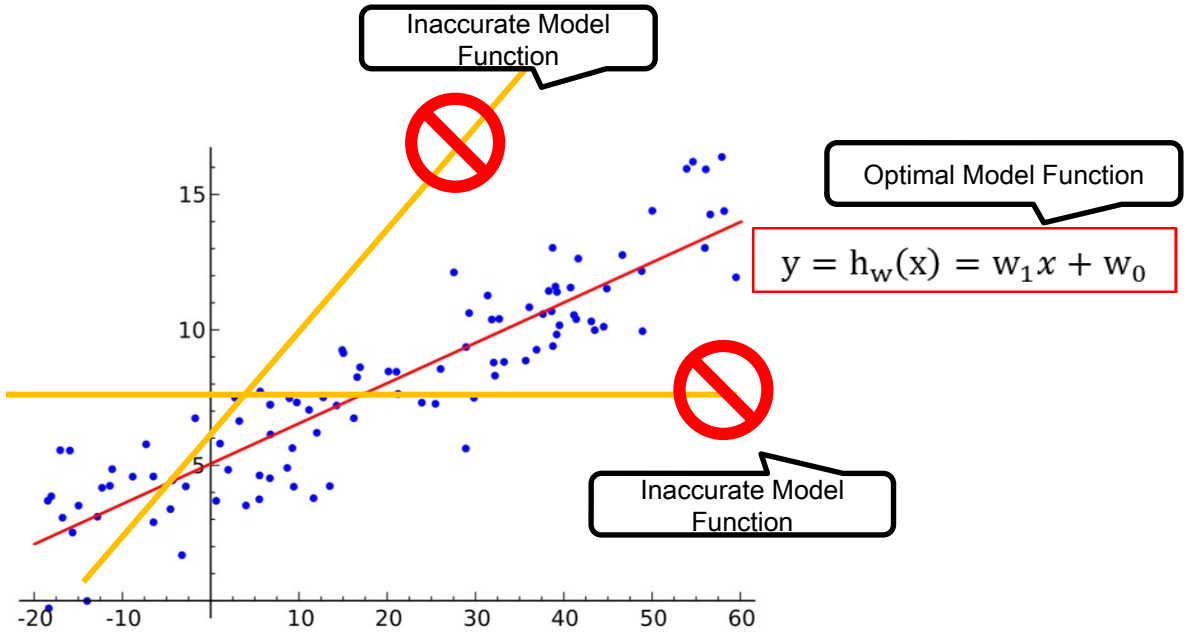
$y = ax + b$



https://datacadamia.com/data_mining/multiple_regression

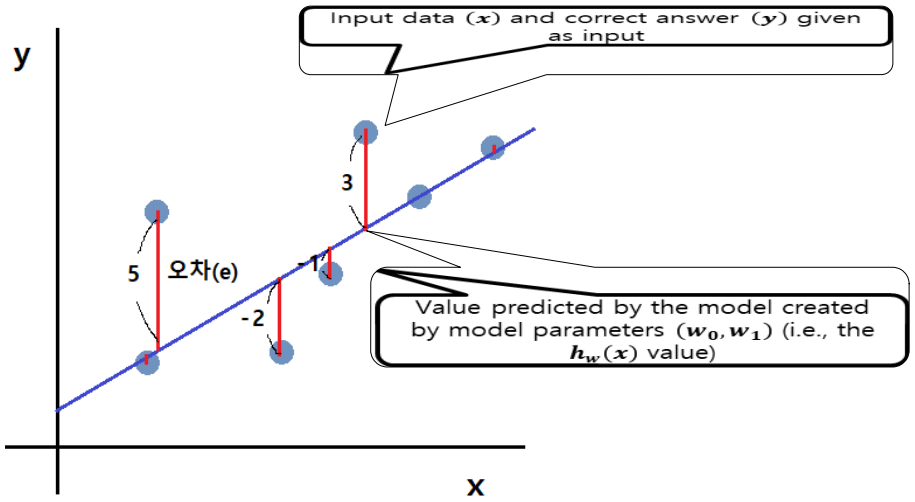
Univariate Linear Regression

- ULR
  - A technique for linearly (=straight-line graph) modeling the correlation between a scalar **independent variable** $\boldsymbol{x}$ and a **dependent variable** $\boldsymbol{y = h(x)}$.
  - Approach
    - Assume an arbitrary straight-line graph function $\boldsymbol{h_w(x) = w_1 x + w_0}$ (w_1: slope/gradient, w_0: y-intercept).
    - Adjust the values of w_1 and w_0 to make the function $\boldsymbol{h_w(x)}$ represent/describe the entire dataset as accurately as possible.
    - But, what exactly does it mean to "most accurately" represent...?



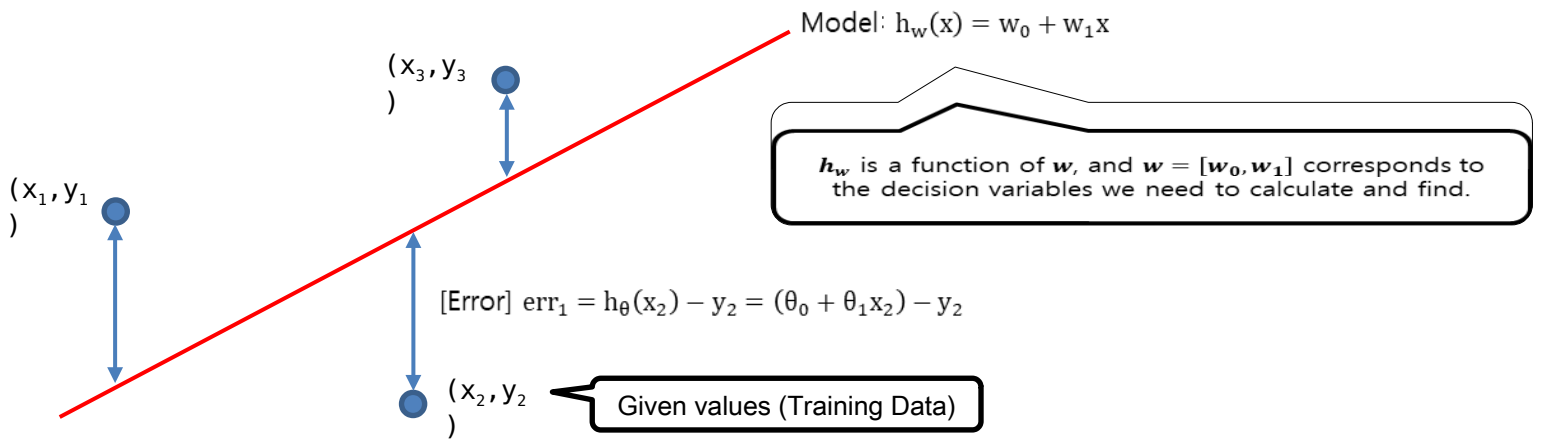
Univariate Linear Regression

- ULR
  - Loss Function (오차함수)
    - The **core** of the ULR problem is to define a **Loss Function** to measure how accurately the function $h_w(x)$ **represents** the entire dataset, and then to find the w_1, w_0 values that **mini mize the Loss Function**.
      - ✓ **Note:** For a given data pair (x_i, y_i) used for training, the returned value of $h(x_i)$ is the **predicted value**, and y_i corresponds to the **correct answer** (ground truth).
      - ✓ (x_i, y_i) are the values given for training, and to find the optimal function $h_w(x)$, we must find the optimal w_1, w_0.
    - Defining the Loss Function



Univariate Linear Regression

- ULR
  - Defining the Loss Function
    - Error = Predicted Value – Actual Value
      - ✓ When $(\boldsymbol{x}, \boldsymbol{y})$ is given, the **Actual Value** (ground truth) is $\boldsymbol{y}$, and the **Predicted Value** is $\boldsymbol{h_w(x)}$.
      - ✓ The difference between the two is defined as the **Error**, and the **Optimal Model Function** can be found by searching for $\boldsymbol{w_1, w_0}$ that minimizes the sum of the errors over the entire training dataset.



    - **(Modified) Error** = $\sum(\boldsymbol{y_i} - \boldsymbol{h_w(x_i)})^2, \forall i = 1,2,\dots,N$ // **For all given sample data...**
      - ✓ If the errors for the first and second training data are -5 and +5, respectively, the sum becomes 0 (=no error?), which is a misleading assessment. To prevent this erroneous judgment, the goal is to find $\boldsymbol{w_0, w_1}$ that minimizes the **sum of the squared errors** for all data, by calculating the **square of each error**.

Univariate Linear Regression

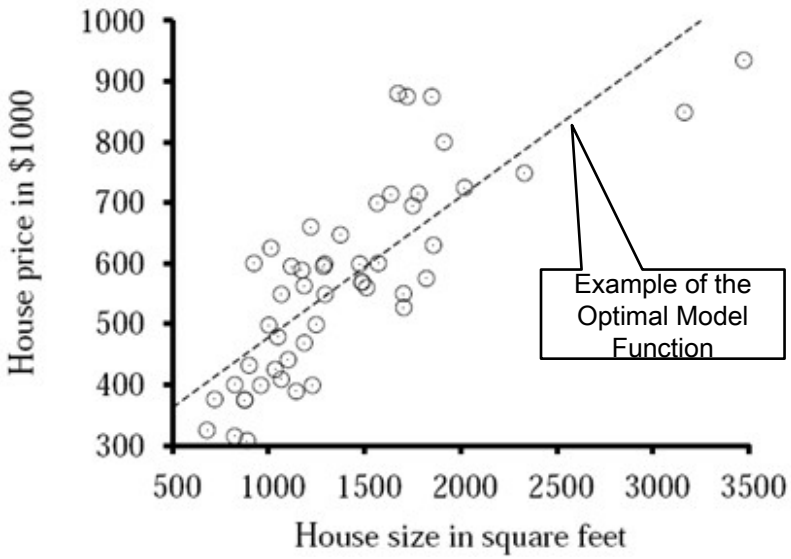
- The task of finding $h_{\mathbf{w}}$ that best fits the data

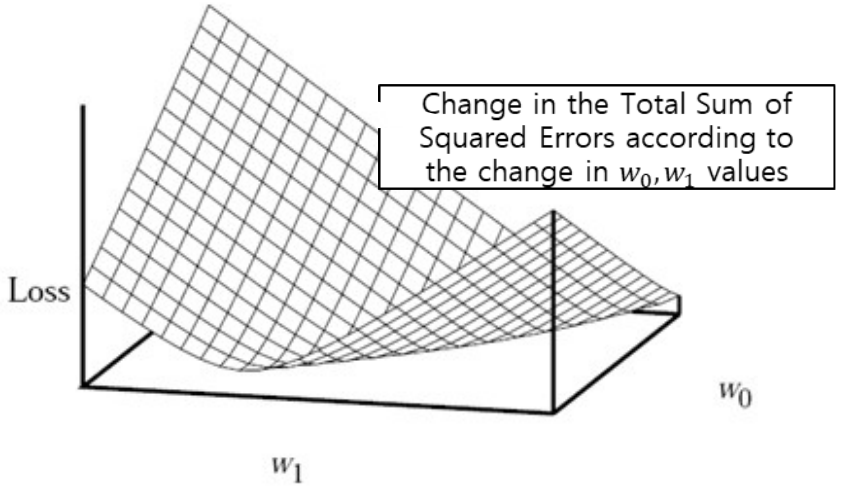
$$h_{\mathbf{w}}(x) = w_1 x + w_0$$

- Need to find the values of the weights $[w_0, w_1]$ $(= \mathbf{w})$ that minimize the sum of squared error over all the training examples:

$$\sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$$

N : The number of samples (x_i, y_i) for training





Univariate Linear Regression

- To minimize the sum $\sum_{j=1}^{N}(y_j - (w_1 x_j + w_0))^2$

$$\frac{\partial}{\partial w_0}\sum_{j=1}^{N}(y_j - (w_1 x_j + w_0))^2 = -2\sum_{j=1}^{N}(y_j - (w_1 x_j + w_0)) = 0$$

$$\frac{\partial}{\partial w_1}\sum_{j=1}^{N}(y_j - (w_1 x_j + w_0))^2 = -2\sum_{j=1}^{N}(y_j - (w_1 x_j + w_0))x_j = 0$$

These equations have a unique, closed-form solution:

$$w_0 = \left(\sum y_j - w_1\left(\sum x_j\right)\right)/N; \qquad w_1 = \frac{N\left(\sum x_j y_j\right) - \left(\sum x_j\right)\left(\sum y_j\right)}{N\left(\sum x_j^2\right) - \left(\sum x_j\right)^2}$$

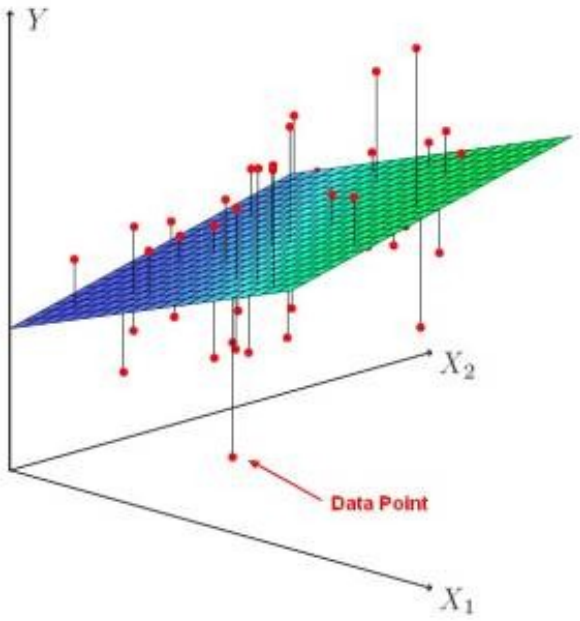
Although we could use the Local Search algorithm learned previously, we can calculate the optimal $\boldsymbol{w_0, w_1}$ using the property that the error is minimized at the point where the derivative equals 0.

The result of rearranging the second equation above for $\boldsymbol{w_1}$ and then plugging the resulting $\boldsymbol{w_1}$ into the equation for $\boldsymbol{w_0}$ and simplifying.

MULTIVARIATE LINEAR RE-GRESSION

Multivariate Linear Regression

- MLR
  - Until now, we have looked at **Univariate LR** where the independent variable is only **one (a scalar)**.
  - What if there are multiple independent variables?
    - ex: {room: 3 개 , Bathroom: 2 개 } = 80 만원 ...
  - **We extend Univariate LR to use Multivariate LR!** (**The operating principle is the same**)



Multivariate Linear Regression

- Each example $\mathbf{x}_j$ is an d-element vector

$$h_{\mathbf{w}}(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^T \mathbf{x}_j = \sum_i w_i x_{j,i}$$

where $x_{j,0} = 1$ is a dummy input attribute (used for the purpose of finding the optimal **intercept** value)

- $h_{\mathrm{w}}(\underline{\mathrm{x}}_{\mathrm{j}})$

$= \mathrm{w}_0 \mathrm{x}_{\mathrm{j},0} + \mathrm{w}_1 \mathrm{x}_{\mathrm{j},1} + \mathrm{w}_2 \mathrm{x}_{\mathrm{j},2} + \cdots + \mathrm{w}_{\mathrm{d}} \mathrm{x}_{\mathrm{j,d}}$

$= \sum_{\mathrm{i}} \mathrm{w}_{\mathrm{i}} \mathrm{x}_{\mathrm{j,i}}$

$= \underline{\mathrm{w}}^{\mathrm{T}} \underline{\mathrm{x}}_{\mathrm{j}}$

$$= \begin{bmatrix} \mathrm{w}_0 & \cdots & \mathrm{w}_{\mathrm{d}} \end{bmatrix} \begin{bmatrix} \mathrm{x}_{\mathrm{j},0} \\ \cdots \\ \mathrm{x}_{\mathrm{j,d}} \end{bmatrix}$$

Multivariate Linear Regression

- The best weight vector **w** that minimizes loss over the examples:
- $$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j (y_j - \mathbf{w} \cdot \mathbf{x}_j)^2$$
- That is, the goal is to minimize total_error = $\text{minimize } \|\mathrm{Xw} - \mathrm{y}\|_2^2$
  - Find w* that satisfies $\mathrm{Xw} - \mathrm{y} = 0$
- The analytical solution is

$$\mathbf{w}^* = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}$$

To prepare for the case where the $\boldsymbol{X}$ matrix is non-invertible, we calculate $\boldsymbol{w}^*$ through the **pseudo-inverse** by multiplying both sides (on the left) by $\boldsymbol{X^T}$, and then multiplying both sides (on the left again) by $\boldsymbol{(X^T X)^{-1}}$.

where $\mathbf{X}$ is the data matrix of inputs with one d-dimensional example per row

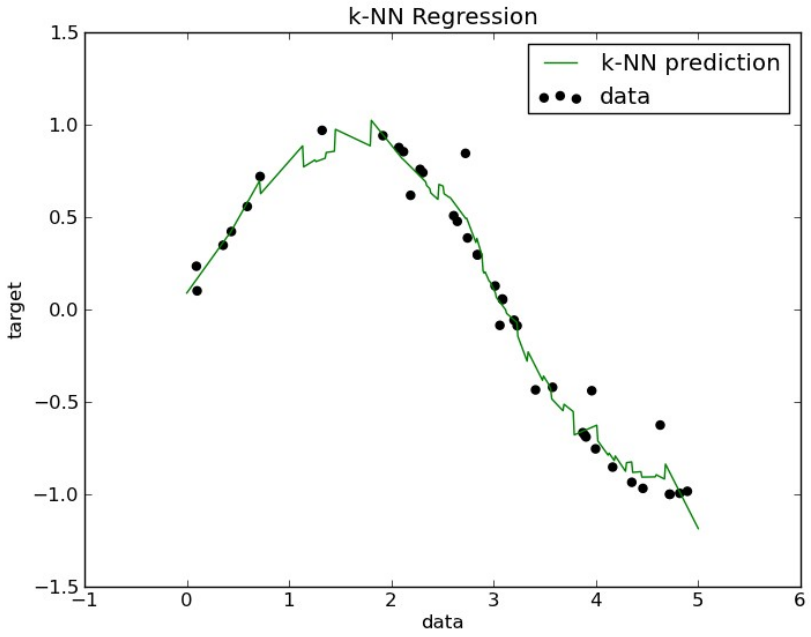
- j-th row contains $d + 1$ feature values including $x_{j,0} = 1$
- Note, Univariate LR corresponds to the special case where $\boldsymbol{d} = 1$.

NEAREST NEIGHBOR MODELS

Nearest Neighbor Models (k-NN)

- If the **data is distributed linearly**, it can be predicted using **Linear Regression**.
- However, what if the **data is non-linear**, as shown below?

Let's use k-NN!



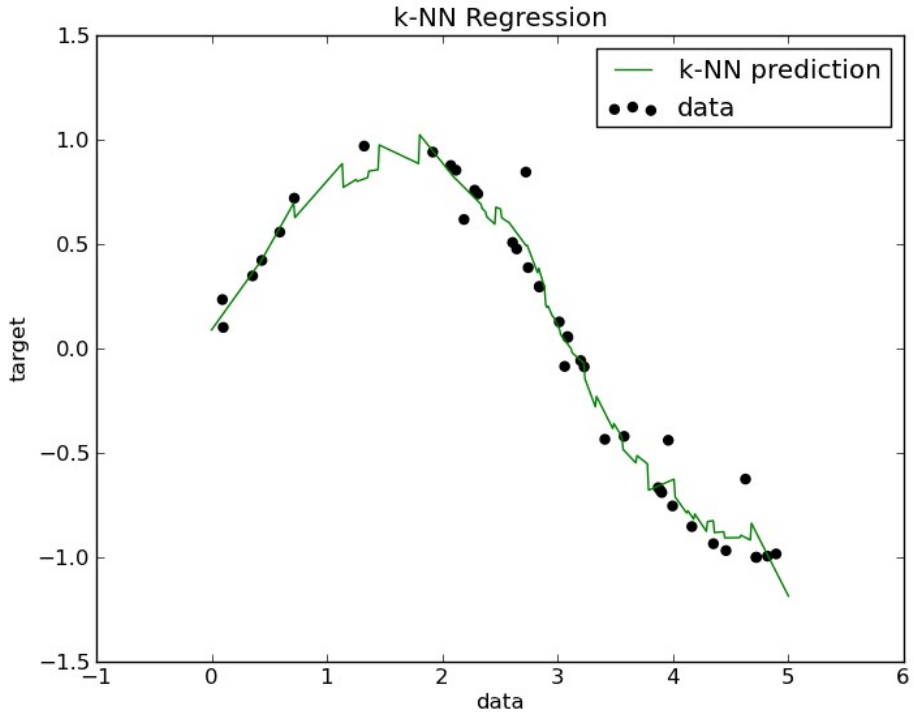
Nearest Neighbor Models (k-NN)

- NN model
  - **Input:** Training data `(x_i, y_i)`
  - **Usage:** When unknown data `x_new` is given, predict what the answer `y_new` is.

- Method of Operation
  - The NN model does **not** have a separate **'training' process**.
  - When a new input $\boldsymbol{x_{new}}$ is given, it extracts k data points from the training data that have $\boldsymbol{x}$ values **neighboring/adjacent** to $\boldsymbol{x_{new}}$, and then calculates $\boldsymbol{y_{new}}$ by finding the **average** of the $\boldsymbol{y_i}$ values of those data points.
    - For this, a **method to evaluate** how close/adjacent two $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ are is required.

Nearest Neighbor Models (k-NN)

- Given a query $\mathbf{x}_q$ (= Unknown input data), find the k nearest neighbors $NN(k, \mathbf{x}_q)$
  - $\boldsymbol{NN(x_q, k)}$: Used by extracting a set of $\boldsymbol{k}$ data points that are adjacent/neighboring to $\boldsymbol{x_q}$ along t he $\boldsymbol{x}$-axis.
  - **Regression** (Two methods for predicting $\boldsymbol{y_{new}}$ using $\boldsymbol{k}$ neighboring values):
    - mean or median of $NN(k, \mathbf{x}_q)$ or // NN 집합 내의 y_i 값의 평균/중간값을 사용하거나
      solve a linear regression problem on $NN(k, \mathbf{x}_q)$ // NN 집합 내의 값에 LR 적용

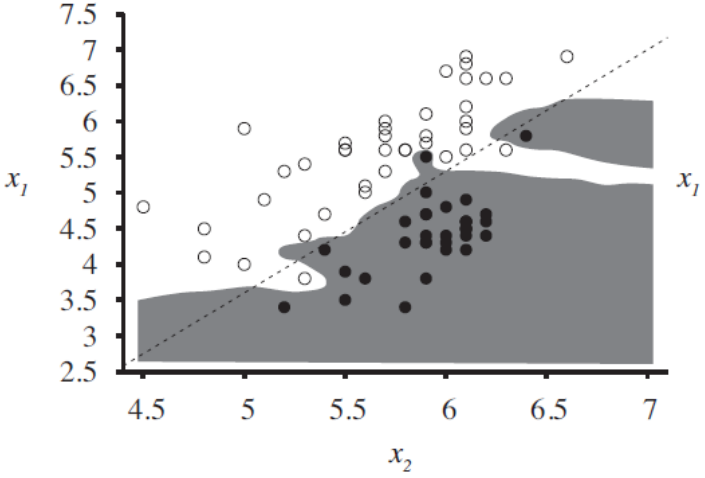


- **Green solid line**: The result of applying k-NN to all $\boldsymbol{x}$ values between $[\boldsymbol{0}, \boldsymbol{5}]$.
- **Black dots**: The $(\boldsymbol{x_i}, \boldsymbol{y_i})$ values given beforehand for training.

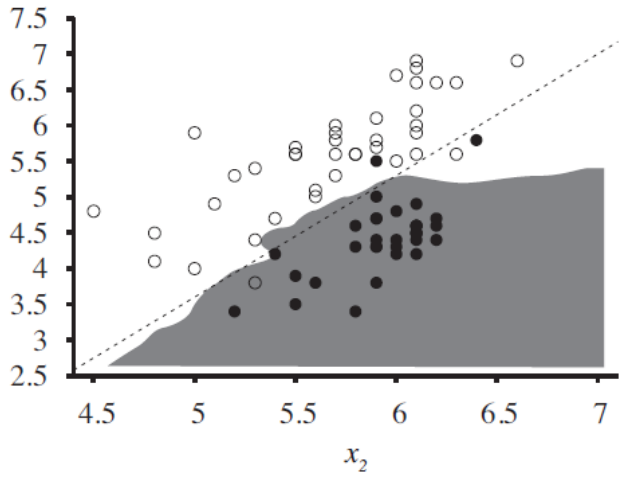
- The k-NN technique does not create a separate model or go through training, but it shows effective performance in predicting values.
- Furthermore, it can handle **non-linear data** more effectively compared to linear regression.

Nearest Neighbor Models (k-NN)

- Given a query $\mathbf{x}_q$, find the k nearest neighbors $NN(k, \mathbf{x}_q)$
  - Furthermore, the k-NN technique can also be used for **classification** problems.
  - Classification: plurality vote of $NN(k, \mathbf{x}_q)$ // Majority rule



Overfitting with $k = 1$



O.K. with $k = 5$

- In the left graph, which visualizes the (data, answer) pairs, the **data** = (x_1, x_2), and the **answer** = **(white 0 or black 1)**.
- The goal of a general classification problem is to establish the optimal criteria (e.g., the solid line in the graph) that separates different types of data and to accurately classify unknown data.

Classification using k-NN: Method of Operation

- For all (x_1, x_2), after selecting the k closest neighbors, the query is classified as white if the majority of selected data is white and classified as black if the majority is black.
- The problem of **overfitting** occurs when $k = 1$, while $k = 5$ shows considerably smooth classification performance.
- Therefore, determining the value of k is very important → The optimal k value can be practically determined by trying various

Nearest Neighbor Models (k-NN)

- Distance Measurement Method
  - The ***NN*** technique uses 'neighboring or closest data' in its calculation process, and 'neighboring' means a short '**distance**'.
  - Distance can be measured in various ways, and a generalized method called **Minkowski distance** is used to measure distance.

- Distance metric: Minkowski distance (also known as L^p norm)

$$L^p(\mathbf{x}_j, \mathbf{x}_q) = \left(\sum_i \left| x_{j,i} - x_{q,i} \right|^p \right)^{1/p}$$

  - $p = 2$: Euclidean distance // The most commonly used measurement standard
  - $p = 1$: Manhattan distance
    - $p = 1$ with Boolean attributes(when each $\boldsymbol{x}$ value consists only of 0s and 1s): Hamming distance

Nearest Neighbor Models (k-NN)

- Note:
  - Depending on the problem, it may be necessary to **normalize** the data ($\boldsymbol{x}$) values (**Normalization**: transforming variables with different ranges so they have similar ranges).
  - Example: Monthly Rent Prediction Problem
    - In a training dataset composed of $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y}$ is the answer (label)
    - **Data = $(\boldsymbol{x_1}, \boldsymbol{x_2})$**, where $\boldsymbol{x_1}$ = house size (square meters), and $\boldsymbol{x_2}$ = number of bathrooms,
    - If $\boldsymbol{x_1}$ has values between 0 and 100, and $\boldsymbol{x_2}$ has values between 0 and 5
    - When calculating the distance between two different data points, the difference in the $\boldsymbol{x_1}$ values is large, and the difference in the $\boldsymbol{x_2}$ values is usually small. This means the distance value is primarily determined by $\boldsymbol{x_1}$, and $\boldsymbol{x_2}$ has almost no influence on the prediction.
    - When calculating the distance, it is necessary to **limit the range of each variable's values to a similar level** so that $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ variables contribute equally to the distance calculation.

- Normalization: $x_{j,i} \rightarrow (x_{j,i} - \mu_i)/\sigma_i$
  - μ_i: mean of the values in the i-th dimension
  - σ_i: standard deviation of the values in the i-th dimension

Nearest Neighbor Models (k-NN)

- Various "distance" calculation methods

Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i_j})^2]}$$

Manhattan Distance

Manhattan Distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

$$d(x, y) = (\sum_{i=1}^{n} (x_i - y_i)^p)^{\frac{1}{p}}$$

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

https://www.geeksforgeeks.org/k-nearest-neighbours/

Nearest Neighbor Models (k-NN)

- Example of KNN code for classifying the input value (p) into Group 0 or Group 1

```
distance=[]
for group in points:
    for feature in points[group]:

        #calculate the euclidean distance of p from training points
        euclidean_distance = math.sqrt((feature[0]-p[0])**2 +(feature[1]-p[1])**2)

        # Add a tuple of form (distance,group) in the distance list
        distance.append((euclidean_distance,group))

# sort the distance list in ascending order
# and select first k distances
distance = sorted(distance)[:k]

freq1 = 0 #frequency of group 0
freq2 = 0 #frequency og group 1

for d in distance:
    if d[1] == 0:
        freq1 += 1
    elif d[1] == 1:
        freq2 += 1

return 0 if freq1>freq2 else 1
```

Calculates the Euclidean distance between the input value (p) and all points in the dataset.

Extracts k data points that have the minimum distance.

Counts the number of data points included in Group 0 and the number of data points included in Group 1 among the k data points.

https://www.geeksforgeeks.org/k-nearest-neighbours/

End