# Influencing improvement

- Agent's component
- Agent's prior knowledge, which influence that mode lit builds
- the feedback available to learn from.

## Components

- Given an intelligent agent that performs some intelligent tasks, any components of agent program can be improved by learning.

## Prior knowledge

- Inductive learning (귀납적 학습): learning a general function or rule (possibly incorrect) from specific input-output pairs
  - Bottom to top
  - Specific to general
- Deductive learning (연역적 학습): going from a general rule to a new specific rule that is logically entailed, but is useful because it allows more efficient processing
  - Top to bottom
  - general to specific

## Feedback

- **feedback** on its percept sequence
- **no feedback** on its percept sequence
- **rewards** for taking a sequence of actions based on its percept sequence

| - | Supervised Learning | Unsupervised Learning |
|---|---|---|
| Training Data | labeled | unlabeled |
| Computational complexity | simpler | Computationally complex |
| Accuracy | high | less accurate |

### Supervised learning

- the agent observes some examples of input-output pairs first and then learns a function or a relationship that maps from inputs to output.
- Attributes/Features: the inputs are independent variables in the problem domain
- Target attribute: the output is the dependent variable which is dependent on the inputs.
- Model: the learned function or relationship
- The agent learns a model using examples and uses this model to predict the outcomes for new inputs.

### Unsupervised learning

- The agent collects adequate examples in the problem domain but it does not get any explicit feedback to the examples.
- The agent can make sense of the examples through identifying clusters or frequent patterns in the data.
- When shown a large number of examples, the agent can learn to identify clusters of similar examples.

### Reinforcement learning

- the agent learns from a series of actions which can be rewards or punishments to improve its performance in completing the task under consideration.
- the feedback helps the agent to enforce positive actions and reduce the negative actions through adjusting the policy.

# Supervised Learning Technique

- Decisinon tree
- Random forests
- Linear regrasssion
- Logistic regression
- K nearest neighbours
- Support vector machines
- Neural networks

## Regression problem

- to predict a continuous value as the output for a given input
- weather temperature: solar radiation, wind direction and speed, geographic location..

- how to predict the output value of a new data instance on the basis of observed features from the existing data (historical examples) in the problem domain.
- **Elements**
  - Collection of existing or historical data samples which are represented by a set of attributes or independent variables
  - The output values of the existing data samples
    - the output variable or attribute must be continuous
- **Regressor**: a function describes the relationship between the attributes of a data sample and the output.
  - takes the values of attributes of a data sample and predicts the output value of this given data sample.

### Evaluate a regressor

- R Square/Adjust R Square
- MSE Mean Square Error/RMSE Root Mean Square Error
- MAD Mean Absolute Error

### Examples of regression problems

- Predict the fuel price using the Brent crude oil price, financial performance of the oil related companies (cash flow, projects lined up, etc.) and/or geopolitical risks (OPEC announcements, government sanctions, etc.)
- Predict the house price of a suburb from the suburb's profile
- Predict the blood pressure of a patient based on the patient's health profile
- Predict the electricity price using temperature, demand and time.

## Classification problem

- to predict discrete or categorical value as the output for a given input
- Pass or Failed given learning outcomes, student ID, prior learning, attitude, commitment and attendance.
- how to put a new data instance into one of predefined categories or classes on the basis of observed features from the existing data in the problem domain.
- **Elements**
  - Collection of existing or historical data samples with class labels
  - Predefined categories or classes
  - Adequate samples in each category or class in the existing or historical data.
- **Logic-based techniques**

    - Decision tree
    - Learning set of rules
- **Perceptron-based techniques**
    - Single-layer perceptron
    - Multi-layer perceptron
    - RBF network
- **SVM**
- **Statistical learning techniques**
    - Naive Bayes classifier
    - Bayesian networks
- **Instance-based learning**
    - K-nearest neighbor (KNN)

### Evaluate a classifier

- Confusion matrix
- Precision
- Recall/Sensitivity
- Specificity
- F1-Score
- Area Under Curve & Receiver Operating Characteristics Curve (AUC-ROC)

### Examples of classification problems

- banking, healthcare, medical diagnosis, marketing (sentiment aalysis), telecommunication, agriculture, security (fraud detection).
- e-mails into spam or non-spam class
- loan applications into an approved or a rejected class.
- patients into having a certain disease or not having that disease groups.
- text into positive or negative sentiment.
- customers into churn or non-churn classes.

## Overfitting

- general phenomenon with all types of learning models.
- a modeling error that occurs when a function is too closely or exactly fit to a limited set of data points.
- more likely as the complexity of models and the number of input attributes increase
- less likely as the number of training examples is large.

# Decision Tree

- if-then statements to define patterns in data
- A if-then statement splits the training data into two or more branches based on some values
- **Best Split**: The results of each branch should be as homogeneous as possible, or has the lowest *impurity* possible.
    - Information gain
    - Gini index

## Implement Decision Tree

- the split (a feature and a condition) that leads to the lowest *impurity* in the resulting child nodes, in a greedy manner
- For categorical features: each unique value can be a split condition.
- For continuous features: midpoints between consecutive sorted unique values are used as split conditions.
- For each potential split condition, the algorithm calculates the impurity of the resulting child nodes.
- The lowest impurity node becomes the split point for that branch.
- The process is then repeated recursively for each child node until all leaf nodes are pure, or the stopping criteria are met.

## The selectino of best split attributes

- **ID3**: employs a top-down, greedy search through the space of possible branches with no backtracking *using information gain*
- C4.5: *using information gain ratio*
- CART: *using Gini Index*
- Gini Index
- Chi-Square
- Reduction in Variance

## Entropy

> the fundamental quantity in information theory. It is a measure of the uncertainty of a random variable.

- the fundamental quantity in information theory. It is a measure of the uncertainty of a random variable

- A more homogeneous node with a clear majority class has low impurity and low entropy, while a more mixed distribution of classes has high impurity and high entropy.

## Information gain

> the decrease in entropy.

- The information gain from the attribute test on (split on A) is the expected reduction in entropy.
- Information gain computes the difference between entropy before the split and average entropy after the split of the dataset based on given attribute values
- $Entropy(S) = -\sum p_i \log_2 p_i$
- $Gain(S, A) = Entropy(S) - Entropy_{remain}(S, A)$

## Gini index

- **For classification**, another impurity measure commonly used for classification tasks in decision trees.
- a lower Gini index indicates lower impurity, meaning that the samples in the node predominantly belong to a single class
- a bit more computational efficient than entropy as it does not involve logarithm calculations. but results are quite similar.
- $Gini(S) = 1 - \sum_{i=1}^{K} p_i^2$

## Variance

- **For regression tree**, the target variable is continuous rather than categorical.
- use variance as a measure of impurity in regression trees.
- lower variance indicates that the data points are closely clustered around the mean
- $Var(S) = \frac{1}{N} \sum (y_i - \mu)^2$

## Prediction

- For classification tasks: the predicted class label is **the majority class** among the training samples in the leaf node.
- For regression tasks: the predicted value is **the mean of the target values** of the training samples in the leaf node.

# Dealing with Overfitting

- **Overfitting**: a common issue in decision trees, where the model captures noise or outliers in the training data rather than the underlying pattern.
- the model performs poorly when applied to new, unseen data.

## Setting Stopping Criteria

- prevent the tree from becoming overly complex, which may lead to overfitting
- applied during the tree construction process
- limiting the maximum depth of the tree
- setting a minimum number of samples per leaf node
- requiring a minimum impurity decrease for a split

## Pruning Strategies

- applied after the tree has been fully grown
- removing branches from the fully grown tree to simplify its structure
- ensure that it captures the underlying patterns in the data rather than noise or outliers
- Pruned trees perform significantly better than unpruned trees when the data contain a large amount of noise.

## Ensemble Methods

- combine multiple decision trees to form a more robust and accurate model.
- address overfitting by averaging the predictions of the individual trees, reducing variance and improving generalization.
- Random Forests
- Gradient Boosted Trees

# Random Forest

> combines multiple weak decision tree models to create a stronger learning model.

- two types of randomness are introduced to ensure that the individual decision trees are diverse and less prone to overfitting.
- **Random sampling of the input data**
- **Bootstraping**:
    - involves sampling with replacement 복원추출 (meaning that some instances appearing multiple times and others not appearing) from the original dataset, creating a new dataset.

    - each decision tree is trained on a slightly different set of data points, reducing the likelihood of overfitting.
- **Random selection of features at each split**
    - At each split in each decision tree, a random subset of features is considered when determining the best split.
    - each tree in the ensemble does not rely on the same set of features for making decisions, resulting in a more diverse set of trees.
    - By considering only a subset of features at each split, the model is less likely to be influenced by a small number of dominant features, leading to a more balanced and accurate prediction.

| 구분 | 데이터 무작위성 (Bootstrapping) | 속성 무작위성 (Feature Subset Selection) |
|---|---|---|
| 적용 위치 | 트리 **훈련 데이터** 선택 단계 | 트리의 **각 분할(split)** 단계 |
| 방법 | 원본 데이터셋에서 **복원 추출**(with replacement)로 샘플링하여 새로운 학습용 부분집합 생성 | 전체 속성 중 무작위로 **일부 속성만** 선택 후, 그 속성들로만 분할 기준 탐색 |
| 특징 | - 각 나무가 다른 데이터 포인트로 학습됨<br>- 일부 샘플은 여러 번 등장, 일부는 제외될 수 있음 | - 각 분할이 다른 속성을 사용 가능<br>- 동일한 속성에 과도하게 의존하지 않음 |
| 효과 | - 트리 간의 **데이터 다양성** 확보<br>- 과적합 감소 | - 트리 간의 **속성 다양성** 확보<br>- 소수 지배적 속성의 영향 축소 |
| 결과 | 더 다양한 데이터 시나리오를 반영한 트리들 생성 | 더 다양한 의사결정 규칙을 반영한 트리들 생성 |

## Predict with Random Forest

- aggregating the predictions of all individual decision trees in the forest.
- **Majority voting**: For classification, Count the number of times each class is predicted by the individual decision trees. The class with the highest count is considered as the final prediction.
- **Averaging**: For regression, Calculate the mean of the predictions made by the individual decision trees. The mean value is considered as the final prediction.

# Linear regression

> **a learning technique that finds a linear relationship between input variables and the target variable** based on a fundamental assumption that there is a linear relationship between input variables and the target variable

- e.g. the input variables (engine size, weight and car age) ➡️ target variable (car fuel efficiency)
  - assumption that there is a linear relationship
- A linear regression technique learns a set of coefficients to estimate the linear relationship between $x$ and $y$, denoted as $h_w$, which can be represented by the following equation.
  - $$h\_w(x) = w_0 + w_1 x_1 + ... + w_n x_n = \sum_{i=0}^{n} w_i x_i$$
  - $w$ is a **weight vector**
  - $$\hat{y} = \sum_{i=0}^{n} w_i x_i$$
- linear regression model is an approximate function between the input variables and the target variable, **there will be an error between the output of the model and the actual output value for a data sample**
  - This error can be represented by a loss function, which calculates the mean square error
  - $$Loss(h_w) = \frac{1}{2m} \sum_{j=1}^{m} (h_w(x_j) - y_j)^2 = \frac{1}{2m} \sum_{j=1}^{m} (y_j - \sum_{i=0}^{n} w_i x_{j,i})^2$$
- **for solving regression problems**

# Solving a linear regression problem

- to find the best linear relationship $h_w$that best fits the training data of $m$ data samples.
  - makes the loss to be minimised.
- to find the best weight vector $w^*$, such that for a given training dataset of $m$ data samples.
  - $w^* = \arg\min_w Loss(h_w)$
- **gradient descent**: continuously resamples the gradient of the weight coefficients in the opposite direction depending on the weight $w$.
  - Until the loss function $Loss(h_w)$ reaches the global minimum
  - to change the individual components of $w$ a little bit in the direction that minimises $Loss(h_w)$, and to do this many times.

- $$w_i \leftarrow w_i + \alpha \sum_{j=1}^{m} x_{j,i} \Big( y_j - h_w(x_j) \Big)$$
  - $\alpha$: the step size, **the learning rate**
- **Training model**: the process of iteratively updating weights with a learning rate to minimise loss, where the final weight vector defines the model used for predicting new data.
- use regularisation on a multivariate linear function to avoid overfitting.
- **Batch gradient descent**: consider the entire training dataset $(X, y)$ at once.
  - $w_0 \leftarrow w_0 + \alpha \sum_{j=1}^{m} \Big( y_j - (w_0 + w_1 x_j) \Big)$
  - $w_1 \leftarrow w_1 + \alpha \Big( \sum_{j=1}^{m} (y_j - (w_0 + w_1 x_j)) \cdot x_j \Big)$
- **Stochastic gradient descent (SGD)**: consider only a single training data sample $(x_j, y_j)$ at a time.
  - $w_0 \leftarrow w_0 + \alpha \big( y_j - (w_0 + w_1 x_j) \big)$
  - $w_1 \leftarrow w_1 + \alpha \big( (y_j - (w_0 + w_1 x_j)) \cdot x_j \big)$
  - can be used in an online setting, where new data is coming one at a time, or offline, where we cycle through the same data as many times as is necessary, taking a step after considering each single example.
  - With a fixed learning rate $\alpha$, the stochastic version does not guarantee convergence.
  - often faster than batch gradient descent.
  - With a schedule of decreasing learning rates (SA), the stochastic version does guarantee convergence.
- These update rules are derived as the next weight update equations by taking the partial derivatives of the loss function with respect to $w_0$ and $w_1$.

# Logistic Regression

> an extension of linear regression in such a way that the output of a linear regression model goes through a logistic function

- $$y(x) = \frac{1}{1 + e^{-x}}$$
- The output value of this logistic function is between 0 and 1.
- 0 is for certainly being labeled "0" and 1 is for certainly being labeled "1", and **a value between 0 and 1 represents the probability of being labeled "1"**
- a logistic regression model: a linear regression model + a logistic function
- **mainly for solving classification problems**

# Nearest Neighbor

> a technique to predict the output of a given new sample based on a collection of existing samples.

- is to find the k-nearest neighbours of given sample in the collection and determine the output based on these k neighbours.
- k is always **chosen to be an odd number.**
- can be used for **both classification and regression problems.**
  - For classification: majority vote of the neighbours.
  - For regression: mean/median (or regression) of the neighbours.
- **Instance-based learning**
  - KNN does not learn a separate model.
  - Instead, it stores all training data and uses them directly at prediction time.
- **Non-parametric model**
  - KNN has no parameters (like weights in linear regression) to train.
  - The model is essentially the full dataset plus a distance measure.

# Distance measures

- **Minkowski distance** or $L^p$ **norm**
  - $L^p(x_j, x_q) = \left(\sum_i |x_{j,i} - x_{q,i}|^p\right)^{1/p}$
  - **Euclidean distance**: $p = 2$, for the dimensions are measuring similar properties, such as the width, height and depth of 3D objects.
  - **Manhattan distance**: $p = 1$, for the dimensions are measuring dissimilar properties, such as age, weight, and gender of a patient.
  - **Hamming distance**: the number of attributes on which the two points differ, for Boolean attribute values

# Nomarlization

- use the raw data from each dimension then the total distance will be affected by a change in scale in any dimension
- To avoid this, apply normalization to the measurements in each dimension.
- to compute the mean $\mu_i$ and standard deviation $\sigma_i$ of the values in each dimension, and rescale them
- The rescaling is done using the formula:

- $x'_{j,i} = \frac{x_{j,i} - \mu_i}{\sigma_i}$
  where $x'_{j,i}$ is the normalized value, $x_{j,i}$ is the original value, $\mu_i$ is the mean, and $\sigma_i$ is the standard deviation.

## Time complexity

- Conceptually trivial: Given a set of N examples and a query $x_q$, iterate through the examples, measure the distance to $x_q$ from each one, and keep the best k.
- $NN(k, x_q)$'s time complexity is $O(N)$, N is the number of examples in the training dataset.
- Use a **k-dimensional tree**: a balanced binary tree with an arbitrary number of dimensions.
  - Time complexity can be improved to $O(\log N)$
  - appropriate **only when there are many more examples than dimensions**
  - It works well with up to 10 dimensions with thousands of examples.
- Use a **Hash table with a locality-sensitive hash (LSH)**
  - Time complexity can be improved to $O(1)$

# SVM

> a framework for finding a boundary that distinctly classifies the data points in an optimal way.

- supervised learning, binary classification
- SVM chooses the boundary with the maximum possible geometric margin, which has the largest distance to the nearest training data points of any class
- initially designed for binary classification problems but can also be applied for solving multi-class classification problems

## Linear discriminant

- $X_i$ is multiplied by its matching weight $w_i$
- all these products are added together and passed to a threshold function
- **Decision surface**: if $g(x) = w \cdot x > 0$ then $f(x) = +1(class1)$ else $f(x) = -1(class2)$
- **Decision function**: $f(x) = \text{sign}(g(x)) = \text{sign}(w_0 + w_1 x)$
  - To make a decision, the continuous value $g(x)$ is passed through the sign function so that it outputs either +1 or -1.
- If the data from the two classes can be separated with a hyperplane, **linearly separable**.

## Hyperplane

- separates the data in 2D by a line or in 3D by a plane

- The orientation of the hyperplane is given by the vector $w$
- the location of the hyperplane is given by $w_0$
- The distance from the origin to the hyperplane is $\frac{|w_0|}{\|w\|}$
- If a given data sample $x^*$ and $g(x^*) = 0$, then this data sample is on the separation boundary. It can normally be assigned to any class.
- **geometric margin**: the minimum distance between the samples and the hyperplane by constructing and solving a constrained optimization problem
  - $\gamma_i = y_i \left( \frac{w}{\|w\|} \cdot x_i + \frac{w_0}{\|w\|} \right)$
- **primary optimization problem**: to maximize the minimal geometric distance across the training dataset of m samples.
  - $$\max_{w,w_0} \left( \min_{i=1,\ldots,N} \gamma_i \right) = \max_{w,w_0} \left( \min_{i=1,\ldots,N} \left( y_i \left( \frac{w}{\|w\|} \cdot x_i + \frac{w_0}{\|w\|} \right) \right) \right)$$
  - $\min_{w,w_0} \ \frac{1}{2}\|w\|^2$
  - $\text{s.t.} \ \ y_i(w \cdot x_i + w_0) \geq \min_{i=1,\ldots,N} \left( y_i(w \cdot x_i + w_0) \right)$
- **dual optimization problem**: easier to solve. More importantly the dual optimisation problem enables the so-called kernel trick in SVM
  - $\max_\alpha \ \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$
  - $\min_\alpha \ \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \left( x_i \cdot x_j \right) \ - \ \sum_{i=1}^N \alpha_i$
  - $\text{s.t.} \quad \sum_{i=1}^m \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, 2, \ldots, N$

# Attractive Properties

- SVMs construct a maximum margin separator
  - the largest possible distance to example points, helping to improve generalization
- SVMs create a linear separating hyperplane
  - **kernel trick**: to embed the data into a higher-dimensional space
  - Often data that are not linearly separable in the original input space are easily separable in a higher-dimensional space
  - In general (excepted some special cases) if we have $N$ data points then they will always be separable in spaces of $N$ dimensions or more
- SVMs are a nonparametric method
  - retain training examples and potentially need to store them all
  - In practice, they often end up retaining only a small fraction of examples
  - have the flexibility to represent complex functions, but they are resistant to overfitting
- not usually expect to find a linear separator in the input space $x$, but we can find linear separators in the high-dimensional feature space $F(x)$ simply by replacing $(x_j x_k)$ in
  - $argmax_\alpha \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k (x_j \cdot x_k)$

    - with $K(x_j, x_k) = F(x_j) \cdot F(x_k)$
    - $F(x_j) \cdot F(x_k)$ can often be computed without first computing $F$ for each point.
- In a higher dimensional feature space, which is created by transformation $F(x)$, if we can express $K(x_j \cdot x_k) = F(x_j) \cdot F(x_k)$, the kernel function $K(x_j \cdot x_k)$ can be applied to pairs of input data to evaluate dot product in some corresponding feature space.
    - kernel trick is to plug a kernel function $K(x_j \cdot x_k)$ into the dual optimisation problem to replace $(x_j \cdot x_k)$
    - Optimal linear separators can be found efficiently in feature spaces with billions of (or, in some cases, infinitely many) dimensions.
    - we can learn in the higher-dimensional space, but we compute only kernel functions rather than the full list of features for each data point.

# Classification evaluation metrics

| 용어 | 설명 |
|---|---|
| True Positive (TP) | 실제 1, 예측 1 |
| True Negative (TN) | 실제 0, 예측 0 |
| False Positive (FP) | 실제 0, 예측 1 (0을 잘못 1로 예측) |
| False Negative (FN) | 실제 1, 예측 0 (1을 놓쳐서 0으로 예측) |

## Accuracy

> the proportion of correctly classified instances (data points or samples) among the total instances.

$$Accuracy = \frac{TP + TN}{\text{Total number of predictions}}$$

- a ratio of the number of correct predictions
- it may not be suitable for imbalanced datasets where the class distribution is skewed
    - a model that always predicts the majority class will have high accuracy but may not be useful in practice.

## Precision

> the proportion of true positives among all positive predictions.

$$Precision = \frac{TP}{\text{number of positive predictions}} = \frac{TP}{TP + FP}$$

- the model's ability to not mistakenly view negatives as positives
- A high precision value indicates that the model has made fewer false positive predictions.
- it's useful to minimize the number of false positives.

## Recall

> Sensitivity, True Positive Rate, the proportion of true positive instances among the actual positive instances

$$Recall = \frac{TP}{\text{number of positive instances}} = \frac{TP}{TP + FN}$$

- the model's ability to not mistakenly view actual positives as negatives
- A high recall value indicates that the model has successfully identified a large portion of the actual positive instances
- it's useful when the cost of false negatives is high
  - e.g. in medical diagnosis, where failing to identify a disease can have severe consequences

## F1 Score

> the harmonic mean of precision and recall

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

- a balanced evaluation of the model's performance
- It is particularly useful when dealing with imbalanced datasets, where one class is significantly more prevalent than the other.
- It's useful for imbalanced datasets to balance precision and recall.

# Regression evaluation metrics

## Mean Absolute Error (MAE)

> the average of the absolute differences between the predicted values and the actual values

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- It measures the average magnitude of errors made by the model, without considering their direction
- A lower MAE value indicates that the model has made smaller prediction errors

## Mean Squared Error (MSE)

> the average squared difference between the predicted and actual values

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- It's useful when you want to penalise larger errors more heavily, making it more sensitive to outliers at the same time
- often used as a loss function when training regression models
- A lower MSE value indicates that the model has smaller prediction errors, with a strong preference for avoiding large errors.

## Root Mean Squared Error (RMSE)

> the square root of MSE

$$RMSE = \sqrt{MSE}$$

- it more interpretable as it is in the same units as the dependent variable.

## R-Squared (Coefficient of Determination)

> how well the regression model approximates the actual data

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- the proportion of "sum squared regression (SSR)" and "total sum of squares (SST)"
  - SSR obviously captures the model's prediction errors
  - SST is the variance of the target variable.
    - can be viewed as a naive model ($\hat{y} = \bar{y}$) using the average value of the target variable as the prediction
- $R^2 = 1$: The model predicts perfectly (the error is 0).
- $R^2 = 0$: The model does no better than a naive model that always predicts the mean of the target variable.

- $R^2 < 0$: The model performs worse than simply predicting the mean, meaning its predictions increase the error compared to the naive baseline.