Week1 1

Useful Resources: Datasets

- UCI Repository:
  - http://www.ics.uci.edu/~mlearn/MLRepository.html
- Kaggle:
  - http://www.kaggle.com/

- Distributed/parallel computing + machine learning → Distributed/parallel machine learning
- Machine learning + hardware → AI chips

A Motivating Example

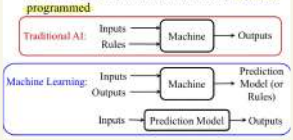
- Given a face image, to classify the face gender: ♀ or ♂

- Once upon a time, to develop an AI system to solve such a task, developers or domain experts need to provide rules and implement them in the system

DL ⊆ ML ⊆ AI
↓ one of methodology
↓ one of tool

Machine Learning Definition

- Machine learning is a type of artificial intelligence that provides computers with the ability to learn from examples/experience without being explicitly programmed



Course Objective

- To provide students with essential concepts and principles of machine learning algorithms
- To enable students to understand how to revise or design (beyond how to use) various machine learning algorithms to solve supervised learning and unsupervised learning problems

A Motivating Example (cont.)

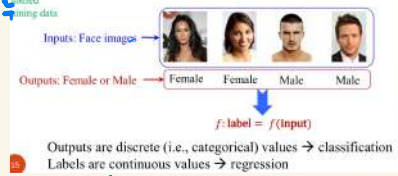
- Limitations:
  - Time consuming
  - The defined rules may not be complete
  - Not able to handle uncertainty

How to Represent an Example?

- Feature engineering (not machine learning focus)
- Representation learning (one of the crucial research topics in machine learning)
  - Deep learning is the current most effective approach to representation learning

Supervised Learning

- The examples presented to computers are pairs of inputs and the corresponding outputs, the goal is to "learn" a mapping or model from inputs to labels

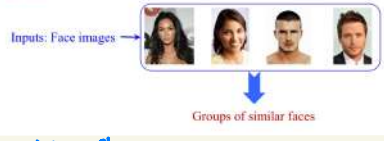


Outputs are discrete (i.e., categorical) values → classification
Labels are continuous values → regression

example:
- Regression
- Classification
- Ensemble Learning

Unsupervised Learning

- The examples presented to computers are a set of inputs without any outputs, the goal is to "learn" an intrinsic structure of the examples, e.g., clusters of examples, density of the examples

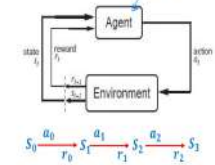


example:
- Clustering
- Density estimation
- Dimensionality Reduction

Reinforcement Learning

- Learning by interacting with an environment to achieve a goal
- Objective: to learn an optimal policy mapping states to actions

(policy) (function)



example:
- Deep Q-Network (DQN) (Mnih)

Limitation of Supervised Learning

- Require sufficient labeled data to train a precise model (i.e., a model with good prediction performance)
  - Sufficiency of labeled data is context-aware, depending on different kinds of applications and specific datasets
- When there is insufficient labeled data, can we still train a precise model?
  - Advanced machine learning paradigms

Advanced paradigms:

- Semi-supervised learning
- Active learning
- Transfer learning

Various machine learning applications:
Face recognition, object recognition, text mining, activity recognition, stock price prediction, etc.

Various learning paradigms:
Supervised learning, unsupervised learning, reinforcement learning, other advanced learning.

Various types of methodologies:
Graphical models, deep learning, empirical risk minimization, entropy-based models, kernel methods, etc.

Various mathematical techniques:
Probability theory, linear algebra, calculus, optimization, information theory, functional analysis, etc.

Typical Learning Procedure

Two phases:

- Training phase
  - Given labeled training data $\{\boldsymbol{x}_i, y_i\}$, for $i = 1, \ldots, N$
  - Apply supervised learning algorithms on $\{\boldsymbol{x}_i, y_i\}$ to learn a model f such that $f(\boldsymbol{x}_i) = y_i$
- Testing or prediction phase
  - Given unseen test data $\boldsymbol{x}_i^*$, for $i = 1, \ldots, T$
  - Use the trained model f to make predictions $f(\boldsymbol{x}_i^*)$'s

→ associate with curiosity, exploration
intrinsic motivation (improve itself)
desire of a learning system to acquire knowledge without any external reward or incentives
useful for continuous improve performance / adapt to new env
extrinsic motivation (
arise from external rewards / incentives.
designed to maximize a specific objective such as accuracy / performance on particular task.
useful in cases where ultimate goal is to optimize a specific outcome / achieve a particular result
dist: could lead to overfitting / suboptimal behavior if rewards not properly designed / calibrated

Supervised Learning (cont.)

In mathematics,

- Given: a set of $\{x_i, y_i\}$ for $i = 1, \ldots, N$, where $x_i = [x_{i1}, x_{i2}, \ldots, x_{im}]$ and y_i is a scalar,
- Goal: to learn a mapping $f: x \to y$ by requiring $f(x_i) = y_i$.
- The learned mapping f is expected to be able to make precise predictions on any unseen x^* as $f(x^*)$

Labeled training data

Output

Prediction model

A vector of features

Example I

x_1	x_2	x_3	x_4
Female	Female	Male	Male
-1	-1	$+1$	$+1$

Image processing & Computer vision tech

$y_1 = -1$ $x_1 = [x_{11}, x_{12}, \ldots, x_{1m}]$
$y_2 = -1$ $x_2 = [x_{21}, x_{22}, \ldots, x_{2m}]$
$y_3 = 1$ $x_3 = [x_{31}, x_{32}, \ldots, x_{3m}]$
$y_4 = 1$ $x_4 = [x_{41}, x_{42}, \ldots, x_{4m}]$

Example II

Compact: easy to operate; very good picture quality; looks sharp!

It is also quite blurry in very dark settings. I will never_buy HP again.

Bag-of-words representation

	F1	F2	F3	F4	F5	F6	…	
y_1 (+1)	1	1	0	0	1	0	…	x_1
y_2 (−1)	0	0	1	1	0	1	…	x_2

Dictionary

F1	F2	F3	F4	F5	F6	…
compact	easy	quite	blurry	good	never_buy	…

Examples III

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Divorced	95K	Yes

binary, categorical, continuous, class

One-hot encoding

	HO	MS_S	MS_M	MS_D	TaxInc	Cheat	
x_1	1	1	0	0	125K	−1	y_1
x_2	0	0	1	0	100K	−1	y_2
x_3	0	0	0	1	95K	+1	y_3

Classification v.s. Regression

- For classification, y is discrete
  - If y is binary, then binary classification
  - If y is nominal not binary, then multi-class classification
- For regression, y is continuous
- Can an example be assigned to more than one categories?

multi-label classification — can be for multiple class

Inductive Learning

- How to learn a predictive model $f(\cdot)$
  - Bayesian Classifiers — Probability
  - Decision Trees — mimic behavior by asking question
  - Artificial Neural Networks
  - Support Vector Machines — Geometry
  - Regularized Regression Model

Training data

Some classification algorithm

$f: x \to y$

Predicted label $y^* = f(x^*)$

Test data x^*

Inductive Learning

Training data

Predicted label y^*

Test data x^*

Lazy Learning — E.g., K nearest neighbor classifier

Typical Learning Procedure

Two phases:

- Training phase
  - Given labeled training data $\{x_i, y_i\}$, for $i = 1, \ldots, N$
  - Apply supervised learning algorithms on $\{x_i, y_i\}$ to learn a model f such that $f(x_i) = y_i$
- Testing or prediction phase
  - Given unseen test data x_i^*, for $i = 1, \ldots, T$
  - Use the trained model f to make predictions $f(x_i^*)$'s

Common Evaluation Metrics

- Classification: Accuracy, error rate, F-score etc.

	Actual	Prediction	
x_1	+1	−1	✗
x_2	−1	−1	✓
x_3	+1	+1	✓
x_4	−1	+1	✗
x_5	−1	−1	✓
x_6	+1	−1	✗
x_7	−1	−1	✓
x_8	+1	+1	✓

Accuracy = 5/8
Error rate = 3/8

- Regression: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE)

Cross Validation

- k-fold cross-validation: partition data into k subsets of the same size
- Hold aside one group for testing and use the rest to build model

Test — Training

Iterations

Training — Test

5-fold Cross-Validation

Partition into 5 subsets

1	2	3	4	5

Hold aside 1 group for testing and use the rest 4 for training

For a specific setting of hyper-parameters H_1, e.g., 10 convolutional layers followed by 2 fully connected layers, etc.

$f(x; \Theta'_1, H_1)$	1. Test	2. Train	3. Train	4. Train	5. Train
$f(x; \Theta'_2, H_1)$	1. Train	2. Test	3. Train	4. Train	5. Train
$f(x; \Theta'_3, H_1)$	1. Train	2. Train	3. Test	4. Train	5. Train
$f(x; \Theta'_4, H_1)$	1. Train	2. Train	3. Train	4. Test	5. Train
$f(x; \Theta'_5, H_1)$	1. Train	2. Train	3. Train	4. Train	5. Test

Evaluation score s_1 for the hyper-parameters setting H_1

$f(X_1; \Theta'_5, H_1)$	$f(X_2; \Theta'_5, H_1)$	$f(X_3; \Theta'_5, H_1)$	$f(X_4; \Theta'_5, H_1)$	$f(X_5; \Theta'_5, H_1)$
1. Test	2. Test	3. Test	4. Test	5. Test

Evaluate performance

Based on evaluation scores of other hyper-parameters settings, $H_2, H_3, \ldots$, to choose the best one

Once the best hyper-parameters setting H_* is determined, all training data is used to train the neural network $f(x; \Theta_*, H_*)$ to deploy for use

Evaluation of Supervised Learning

- Recall: the learned predictive model should be able to make predictions on previously unseen data as accurately as possible
- Solution: the whole training data is divided into "training" and "test" sets, with "training" set used to learn the predictive model and "test" set used to validate the performance of the learned model

Original Training Data — Training Data — Validation Data

All Training data

	F1	F2	F3	F4	F5	F6	…	
y_1 (+1)	1	1	0	0	1	0	…	x_1
y_2 (−1)	0	0	1	1	0	1	…	x_2
…								
y_N (−1)	0	1	0	0	1	1	…	x_N

Used for training — Used for evaluation

Learn $f: x \to y$

$\hat{y}_i = f(x_i)$ for $i = L+1, \ldots, N$ — Evaluate

Some evaluation metric: $\hat{y}_{L+1}, \ldots, \hat{y}_N$ v.s $y_{L+1}, \ldots, y_N$

Notes on Cross-Validation

- Using Neural Networks as an example

Parameters of a neural network, which are to be learned from training data

$f(x; \Theta)$

Output Layer Y

Hidden Layer(s)

Input Layer X

Hyper-parameters of a specific neural network: how many hidden layers, how many hidden nodes in a specific hidden layer, etc.

$f(x; \Theta, H)$

Note: hyper-parameters need to be manually determined, not learned from training data!

Week 2a

Inductive Learning

- How to learn a predictive model $f(\cdot)$
  - Bayesian Classifiers
  - Decision Trees
  - Artificial Neural Networks
  - Support Vector Machines
  - Regularized Regression Model

Uncertainty in Prediction

- Recall: in supervised learning, given a set of $\{x_i, y_i\}$ for $i = 1, \ldots, N$, the goal is to learn a mapping $f: x \to y$ by requiring $f(x_i) = y_i$
- In many applications, the mapping or relationship f between the input features and the output labels is non-deterministic (uncertain)
- For example, suppose you are asked to predict a result of the final of a football cup between Team 1 and Team 2: which team will win?

Notes on Bayesian Classifiers

- Why not computing $P(y = c|\boldsymbol{x})$ for each class directly from the training data, but using Bayes rule to estimate $P(\boldsymbol{x}|y = c)$ and $P(y = c)$ instead?
  - To estimate $P(y = c|\boldsymbol{x})$, one needs to consider each combination of values of $\boldsymbol{x}$
  - E.g., suppose $\boldsymbol{x}$ is m-dimensional and each feature has binary values, then the total number of possible value combinations of $\boldsymbol{x}$ is 2^m --- require a huge size of training dataset and time consuming, not practical!
  - The form of $P(\boldsymbol{x}|y = c)$ can be decomposed based on some assumptions and probability properties --- no need to consider all possible combinations

Summary on Bayesian Classifiers

- Estimate $P(y|\boldsymbol{x})$ via Bayes rule

$$P(y = c|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y = c)P(y = c)}{P(\boldsymbol{x})}$$

- Make predictions based on maximum posterior

$$y^* = c^* \text{ if } c^* = \arg\max_c \frac{P(\boldsymbol{x}|y = c)P(y = c)}{P(\boldsymbol{x})}$$

Advanced decision making: Bayesian Decision Theory (Lecture 2b)

$$y^* = c^* \text{ if } c^* = \arg\max_c P(\boldsymbol{x}|y = c)P(y = c)$$

How to estimate from training data?

- Two implementations will be introduced
  - Naïve Bayes Classifier (Lecture 3)
  - Bayesian Brief Networks or Bayesian Networks (Lecture 4)

An Example


v.s.


- Suppose the next match between the two teams will be hosted by Manchester United
- Head to Head Statistics:

Team	Played	Win	Draw	Loss
Manchester United	151	59	47	45
Manchester City	151	45	47	59

- Among the 59 victories for Manchester United, 32 of them come from playing at home
- Among the games won by Manchester City, 20 of them are obtained while playing on Manchester United home ground
- Among the draw games, 23 of them were played on Manchester United home ground

What result will likely be for the match?

Apply Bayes Rule

- Probability that Manchester United wins: $P(Y = 0) = 0.39$
- Probability that Manchester City wins: $P(Y = 1) = 0.3$
- Probability of a draw game: $P(Y = 2) = 0.31$
- Probability that Manchester United hosted the match it won: $P(X = 0|Y = 0) = 0.54 = 71/59$
- Probability that Manchester United hosted the match won by Manchester City: $P(X = 0|Y = 1) = 0.44$
- Probability that Manchester United hosted the match that is a draw game: $P(X = 0|Y = 2) = 0.49$
- To use Bayes rule to compute $P(Y = 1|X = 0)$, $P(Y = 0|X = 0)$ and $P(Y = 2|X = 0)$

Bayesian decision tree 2b

An Example

- To diagnose whether a patient A is with covid-19: $y = 1$ (Yes) or $y = 0$ (No). Suppose based on a trained Bayesian classifier, we know that $P(y = 1|\boldsymbol{x}_A) = 0.1$. Should the doctor diagnose that A is with covid-19 or not?
- Cost of misclassifying a healthy patient with covid-19:

Stay in hospital or home for quarantine

- Cost of misclassifying a patient with covid-19 as healthy:

Community outbreak!

Decision based on Expected Risk

- Choose the action with minimum risk:

Choose a^* if $a^* = \arg\min_{a_c} R(a_c|\boldsymbol{x})$

- In the covid-19 example,
  - Expected risk of taking action a_0 (predict as healthy)

$$R(a_0|\boldsymbol{x}_A) = 1$$

  - Expected risk of taking action a_1 (predict with covid-19

$$R(a_1|\boldsymbol{x}_A) = 0.9$$

  - Thus, we choose action a_1: predict patient A is more likely with covid-19

Bayesian Classifiers

- From a probability point of view, the mapping $f: \boldsymbol{x} \to y$ can be modeled as a conditional probability $P(y|\boldsymbol{x})$
- Bayesian classifiers aim to learn the mapping $f: \boldsymbol{x} \to y$ for supervised learning in the form of conditional probability $P(y|\boldsymbol{x})$, such that for any input $\boldsymbol{x}^*$, one can use $P(y = c|\boldsymbol{x}^*)$ to predict the probability of $\boldsymbol{x}^*$ belonging to class c, where $c \in \{0, \ldots, C-1\}$
  - How to estimate $P(y = c|\boldsymbol{x}^*)$ for different classes?
  - How to make use of $P(y = c|\boldsymbol{x}^*)$'s to make a prediction?
  - We first review some important probability concepts

Naïve Bayes Classifiers (L3)

- How to estimate $P(\boldsymbol{x}|y = c)$ from the training data
- Assume that the features are conditionally independent given the class label:

$$P(\boldsymbol{x}|y = c) = \prod_{i=1}^{d} P(x_i|y = c)$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]$

$$P(x_1, x_2, \ldots, x_d|y = c) = \prod_{i=1}^{d} P(x_i|y = c)$$

Bayesian Belief Networks (L4)

- A more general approach to modeling the independence and conditional independence among $\boldsymbol{x}$ and y, s.t. the computation of $P(\boldsymbol{x}, y) = P(\boldsymbol{x}|y)P(y)$ is tractable
- Use a graphical representation of the probabilistic relationships among features ($\boldsymbol{x}$) and output class (y)

Bayes Rule or Bayes Theorem

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

- This is induced from product rule

$$P(A,B) = P(A|B) \times P(B) = P(B|A) \times P(A)$$

- Generalized to the case when **A** and **B** are a set of variables

$$P(A_1 \ldots A_k|B_1 \ldots B_p) = \frac{P(B_1 \ldots B_p, A_1 \ldots A_k)}{P(B_1 \ldots B_p)} = \frac{P(B_1 \ldots B_p|A_1 \ldots A_k)P(A_1 \ldots A_k)}{P(B_1 \ldots B_p)}$$

Define Variables

- Let Y be the random variable that represents the result of the match (0, 1, 2)
  - $Y = 0$: Manchester United wins the match
  - $Y = 1$: Manchester City wins the match
  - $Y = 2$: Draw
- Let X be the random variable that represents the team hosting the match (0 or 1)
  - $X = 0$: Manchester United hosts the match
  - $X = 1$: Manchester City hosts the match
- To estimate $P(Y = 0|X = 0)$, $P(Y = 1|X = 0)$, and $P(Y = 2|X = 0)$

$P(Y = 1|X = 0)$

Bayes rule

$$= \frac{P(X = 0|Y = 1) \times P(Y = 1)}{P(X = 0)}$$

Sum rule: $P(X) = \sum_Y P(X, Y)$

$$= \frac{P(X = 0|Y = 1) \times P(Y = 1)}{P(X = 0, Y = 0) + P(X = 0, Y = 1) + P(X = 0, Y = 2)}$$

$$= \frac{P(X = 0|Y = 1) \times P(Y = 1)}{P(X = 0|Y = 0) \times P(Y = 0) + P(X = 0|Y = 1) \times P(Y = 1) + P(X = 0|Y = 2) \times P(Y = 2)}$$

Product rule: $P(X, Y) = P(X|Y)P(Y)$

$$= \frac{0.44 \times 0.3}{0.44 \times 0.3 + 0.54 \times 0.39 + 0.49 \times 0.31} = \frac{0.132}{0.4945} = 0.267$$

Similarly

$P(Y = 0|X = 0) = 0.426$ $P(Y = 2|X = 0) = 0.307$

Marginal Probability

- Let A be a random variable (an input feature / class label in machine learning)
- Marginal probability

$$P(A = a) \quad 0 \le P(A = a) \le 1$$

refers to the probability that variable $A = a$

$$\sum_{a_i} P(A = a_i) = 1$$

Conditional Probability

- Conditional probability:

$$P(B = b|A = a)$$

refers to the probability that the variable B will take on the value b, given that the variable A is observed to have the value a

$$\sum_{b_i} P(B = b_i|A = a) = 1$$

$P(Y = a) = P(Y = a, X = 1) + P(Y = a, X = 2) + \ldots$ an possibilities of X

Estimate Probabilities

Team	Played	Win	Draw	Loss
Manchester United	151	59	47	45
Manchester City	151	45	47	59

- To calculate $P(Y = 0)$, $P(Y = 1)$, and $P(Y = 2)$
  - $P(Y = 0) = \frac{59}{151} \approx 39\%$
  - $P(Y = 1) = \frac{45}{151} \approx 30\%$
  - $P(Y = 2) = 1 - P(Y = 0) - P(Y = 1) = 31\%$

$$\sum_{y_i} P(Y = y_i) = 1$$

Bayesian Classifiers (cont.)

- Bayesian classifiers aim to learn the mapping $f: \boldsymbol{x} \to y$ for supervised learning in the form of conditional probability $P(y|\boldsymbol{x})$ via Bayes rule

posterior $P(y|\boldsymbol{x}) = \frac{P(y,\boldsymbol{x})}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$ prior

- For a classification problem with C classes, given a test data instance $\boldsymbol{x}^*$, a Bayesian classifier computes $P(y = c|\boldsymbol{x}^*), c \in \{0, \ldots, C-1\}$, and $\sum_c P(y = c|\boldsymbol{x}^*) = 1$
- Make a prediction based on the maximum posterior

$$y^* = c^* \text{ if } c^* = \arg\max_c P(y = c|\boldsymbol{x}^*)$$

Return the value of c that maximizes $P(y = c|\boldsymbol{x}^*)$ where $c \in \{0, \ldots, C-1\}$

Loss or Cost

- Actions: a_c, i.e., predict $y = c$, where $c = 0, \ldots, C-1$
- Define λ_{ij} as the loss/cost of a_i when the optimal action is a_j (i.e., predict $y = i$ while true class label is j)
- E.g., in the previous example, $y = 0$: healthy, and $y = 1$: with covid-19 (binary classification)
- We define two corresponding actions, a_0: predict $y = 0$ and a_1: predict $y = 1$, and the losses as

$\lambda_{00} = 0$ predict correctly
$\lambda_{11} = 0$ predict correctly
$\lambda_{01} = 10$ misclassify 1 as 0 (misclassify with covid-19 as healthy)
$\lambda_{10} = 1$ misclassify 0 as 1 (misclassify healthy as with covid-19)

A Special Case

- Making predictions based on maximum posterior is a special case of making decisions based on minimum expected risk
- Define the losses as

$$\lambda_{ij} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases}$$

All correct decisions have no loss and all errors are equally costly

Known as the 0/1 loss

Marginal Probability

- Let A and B be a pair of random variables (features/labels in machine learning).
- Their joint probability

$P(B=b, A=a) \equiv P(A=a, B=b) \equiv P(A=a) \wedge P(B=b)$

refers to the probability that variable $A = a$ and variable $B = b$

Sum Rule

- The connection between joint probability of A and B and marginal probability of A:

$P(A = a) = \sum_{b_i} P(A = a, B = b_i)$ OR $P(A) = \sum_B P(A, B)$

$P(A = a) = \sum_{b_i} \sum_{c_j} P(A = a, B = b_i, C = c_j)$ OR $P(A) = \sum_B \sum_C P(A, B, C)$

Product Rule

- The connections between joint, conditional and marginal probabilities for A and B:

$$P(A = a, B = b) = P(B = b|A = a) \times P(A = a)$$
$$= P(A = a|B = b) \times P(B = b)$$

OR

$P(A, B) = P(B|A) \times P(A) = P(A|B) \times P(B)$

Bayesian Decision Theory: Summary

- If cost of misclassification on different classes is available, rather than only using posterior probabilities (usually estimated by a Bayesian classifier), Bayesian decision theory provides a way to encode the cost information into decision making.

unknown quantity

Estimate Probabilities (cont.)

- Among the 59 victories for Manchester United, 32 of them come from playing at home

$$P(X = 0|Y = 0) = \frac{32}{59} = 54\%$$

- Among the 45 games won by Manchester City, 20 of them are obtained while playing on Manchester United home ground

$$P(X = 0|Y = 1) = \frac{20}{45} = 44\%$$

- Among the 47 draw games, 23 of them were played on Manchester United home ground

$$P(X = 0|Y = 2) = \frac{23}{47} = 49\%$$

- However, the goal is to estimate

$P(Y = 0|X = 0)$ v.s. $P(Y = 1|X = 0)$ v.s. $P(Y = 2|X = 0)$?

Bayesian Classifiers (cont.)

- Based on Bayes rule

$$y^* = c^* \text{ if } c^* = \arg\max_c P(y = c|\boldsymbol{x}^*), c \in \{0, \ldots, C-1\}$$
$$= \arg\max_c \frac{P(\boldsymbol{x}^*|y = c)P(y = c)}{P(\boldsymbol{x}^*)}$$

Constant w.r.t. diff. values of c

$$= \arg\max_c P(\boldsymbol{x}^*|y = c)P(y = c)$$

- Therefore, we make a prediction based on

$$y^* = c^* \text{ if } c^* = \arg\max_c P(\boldsymbol{x}^*|y = c)P(y = c), c \in \{0, \ldots, C-1\}$$

- Take binary classification as an example: 0 vs 1

$P(y = 0|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y=0)P(y=0)}{P(\boldsymbol{x})}$ vs $P(y = 1|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y=1)P(y=1)}{P(\boldsymbol{x})}$

Expected Risk

- Expected risk for taking action a_i:

$$R(a_i|\boldsymbol{x}) = \sum_{c=0}^{C-1} \lambda_{ic} P(y = c|\boldsymbol{x})$$

λ_{10} 0→1
λ_{01} 1→0

- Explanation: to estimate risk of taking an action, one needs to consider all the possible losses
  - Specifically, taking action a_i (predict $\boldsymbol{x}$ belonging to class i), as the ground-truth label of $\boldsymbol{x}$ can be any class in the C classes, we need consider all the possible losses: $\lambda_{i0}, \lambda_{i1}, \ldots, \lambda_{i(C-1)}$
  - We can simply use the average $\frac{\lambda_{i0} + \cdots + \lambda_{i(C-1)}}{C}$ to estimate its risk
  - The possibilities of each loss occurring are different because the probabilities that $\boldsymbol{x}$ belongs to each class are different
    - Use the $P(y = c|\boldsymbol{x})$ as a weight for each loss λ_{ic}, and compute the weighted sum of all possible losses → expected risk

An Example

- Expected risk for taking action a_i: $R(a_i|\boldsymbol{x}) = \sum_{c=0}^{C-1} \lambda_{ic} P(y = c|\boldsymbol{x})$
- Consider the covid-19 example:
  - $P(y = 1|\boldsymbol{x}_A) = 0.1$ and $P(y = 0|\boldsymbol{x}_A) = 0.9$
  - a_0: predict $y = 0$ (healthy), and a_1: predict $y = 1$ (with covid-19)

$\lambda_{00} = 0$ predict correctly
$\lambda_{11} = 0$ predict correctly
$\lambda_{01} = 10$ misclassify 1 as 0 (misclassify with covid-19 as healthy)
$\lambda_{10} = 1$ misclassify 0 as 1 (misclassify healthy as with covid-19)

- Expected risk of taking action a_0 (predict patient A as healthy)

$$R(a_0|\boldsymbol{x}_A) = \lambda_{00} P(y = 0|\boldsymbol{x}_A) + \lambda_{01} P(y = 1|\boldsymbol{x}_A) = 1$$

- Expected risk of taking action a_1

$$R(a_1|\boldsymbol{x}_A) = \lambda_{10} P(y = 0|\boldsymbol{x}_A) + \lambda_{11} P(y = 1|\boldsymbol{x}_A) = 0.9$$

- With the 0/1 loss: $\lambda_{ij} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } i \neq j \end{cases}$
- The expected risk of taking action a_i:

$$R(a_i|\boldsymbol{x}) = \sum_{c=0}^{C-1} \lambda_{ic} P(y = c|\boldsymbol{x})$$
$$= \lambda_{i0} P(y = 0|\boldsymbol{x}) + \cdots + \lambda_{i(i-1)} P(y = i-1|\boldsymbol{x}) + \lambda_{ii} P(y = i|\boldsymbol{x}) + \lambda_{i(i+1)} P(y = i+1|\boldsymbol{x}) + \cdots + \lambda_{i(C-1)} P(y = C-1|\boldsymbol{x})$$
$$= \sum_{c=0}^{C-1} P(y = c|\boldsymbol{x}) - P(y = i|\boldsymbol{x}) = 1 - P(y = i|\boldsymbol{x})$$

$\sum_c P(y = c|\boldsymbol{x}) = 1$

- The expected risk of taking action a_i:

$$R(a_i|\boldsymbol{x}) = 1 - P(y = i|\boldsymbol{x})$$

- Choose an action with minimum expected risk,

Choose a_i if $R(a_i|\boldsymbol{x}) = \min_{a_c} R(a_c|\boldsymbol{x})$

Equivalent to

Predict $y = c^*$ if $P(y = c^*|\boldsymbol{x}) = \max_c P(y = c|\boldsymbol{x})$

Week 3 3ab

f(x)=y

f: x→y f: P(y|x) training: P(y|x) testing: P(y=c|x*)

Bayesian Classifiers: Recall

- To learn a prediction function via $P(y|\mathbf{x})$ using Bayes rule

$$P(y=c|\mathbf{x}) = \frac{P(\mathbf{x}, y=c)}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y=c)P(y=c)}{P(\mathbf{x})}$$

- Make predictions based on maximum posterior

$$y^* = c^* \text{ if } c^* = \arg\max_c \frac{P(\mathbf{x}|y=c)P(y=c)}{P(\mathbf{x})} \quad \text{the 0/1 loss}$$

$$y^* = c^* \text{ if } c^* = \arg\max_c P(\mathbf{x}|y=c)P(y=c)$$

Easy to estimate

Still difficult to estimate as $\mathbf{x}$ contains many input variables, some are discrete, and others are continuous

Naïve Bayes Classifiers

- How to estimate $P(\mathbf{x}|y)$ from the training data?
- Assume that the features are conditionally independent given the class label:

$$P(\mathbf{x}|y=c) = \prod_{i=1}^{d} P(x_i|y=c)$$

where $\mathbf{x} = [x_1, x_2, \dots, x_d]$

$$P(x_1, x_2, \dots, x_d|y=c) = \prod_{i=1}^{d} P(x_i|y=c)$$

Independence

$P(A|B) = P(A)$, $P(B|A) = P(B)$

- Let A and B be two random variables.
- A is said to be independent of B, if the following condition holds:

$$P(A,B) = P(A|B) \times P(B) = P(A) \times P(B) \qquad P(A|B) = P(A)$$

$$P(A,B) = P(B|A) \times P(A) = P(A) \times P(B) \qquad P(B|A) = P(B)$$

- E.g., let A and B denote the results of two matches in different leagues, knowing the result of one match (e.g., the value of A) does not affect the possibility of result for the other match (i.e., the value of B).

Naïve Bayes – Induction

- The conditional independence between **A** and **B** given **C** can also be written as follows:

$$P(\mathbf{A}, \mathbf{B}|\mathbf{C}) = P(\mathbf{A}|\mathbf{C}) \times P(\mathbf{B}|\mathbf{C})$$

Naïve Bayes Classifier: $P(\mathbf{x}|y=c) = \prod_{i=1}^{d} P(x_i|y=c)$

Assume that the features are conditionally independent given the class label

$$P(\mathbf{x}|y=c) = P(x_1, x_2, \dots, x_d|y=c)$$

Define $\mathbf{X}^{(d-1)} = [x_1, \dots, x_{d-1}]$

$$= P(\mathbf{X}^{(d-1)}, x_d|y=c) \quad \text{Features are conditionally independent given class label}$$

$$= P(\mathbf{X}^{(d-1)}|y=c)P(x_d|y=c)$$

Define $\mathbf{X}^{(d-2)} = [x_1, \dots, x_{d-2}]$

$$= P(\mathbf{X}^{(d-2)}, x_{d-1}|y=c)P(x_d|y=c)$$

Features are conditionally independent given class label

$$= P(\mathbf{X}^{(d-2)}|y=c)P(x_{d-1}|y=c)P(x_d|y=c)$$

Recursively apply conditional independence

$$= P(x_1|y=c)P(x_2|y=c)\cdots P(x_d|y=c)$$

$$= \prod_{i=1}^{d} P(x_i|y=c)$$

Conditional Independence (cont.)

A more general case:

- Let **A**, **B**, and **C** be three sets of random variables.
- The variables in **A** are said to be conditionally independent of the variables in **B**, given the variables in **C** observed, if the following condition holds:

$$P(\mathbf{A}|\mathbf{B}, \mathbf{C}) = P(\mathbf{A}|\mathbf{C})$$

$$P(\mathbf{x}|y=c) = \prod_{i=1}^{d} P(x_i|y=c) \; ?$$

- The conditional independence between **A** and **B** given **C** can also be written as follows:

$$P(\mathbf{A}, \mathbf{B}|\mathbf{C}) = \frac{P(\mathbf{A}, \mathbf{B}, \mathbf{C})}{P(\mathbf{C})} \quad \text{Product rule} \quad P(\mathbf{V}|\mathbf{C}) = \frac{P(\mathbf{V}, \mathbf{C})}{P(\mathbf{C})} \text{ where } \mathbf{V} = \{\mathbf{A}, \mathbf{B}\}$$

$$= \frac{P(\mathbf{A}, \mathbf{B}, \mathbf{C})}{P(\mathbf{B}, \mathbf{C})} \times \frac{P(\mathbf{B}, \mathbf{C})}{P(\mathbf{C})} \qquad \frac{P(\mathbf{A}, \mathbf{U})}{P(\mathbf{U})} = P(\mathbf{A}|\mathbf{U}) \text{ where } \mathbf{U} = \{\mathbf{B}, \mathbf{C}\}$$

$$= P(\mathbf{A}|\mathbf{B}, \mathbf{C}) \times P(\mathbf{B}|\mathbf{C}) \quad \text{Product rule}$$

$$= P(\mathbf{A}|\mathbf{C}) \times P(\mathbf{B}|\mathbf{C}) \quad \text{Conditional independence}$$

How Naïve Bayes Classifier Works

Naïve Bayes Classifier: $P(\mathbf{x}|y=c) = \prod_{i=1}^{d} P(x_i|y=c)$ — Easier to estimate

- To classify a test record $\mathbf{x}^*$, we need to compute the posteriors for each class by using

$$P(y=c|\mathbf{x}^*) = \frac{\left(\prod_{i=1}^{d} P(x_i^*|y=c)\right)P(y=c)}{P(\mathbf{x}^*)}$$

- $P(\mathbf{x}^*)$ is constant for each class c, it is sufficient to choose the class that maximizes the numerator

$$P(y=c|\mathbf{x}^*) \propto \left(\prod_{i=1}^{d} P(x_i^*|y=c)\right)P(y=c)$$

P(A|B) = P(A)

P(A,B|C) = P(A|C) × P(B|C)

P(A₀…Aₙ|B) = P(A₀|B) × … × P(Aₙ|B)

μ(mean) = Σx / N

σ(variance²) = Σ(x−μ)² / (n−1)

Example of Naïve Bayes Classifier

→ see method below

Naive Bayes Classifier:

P(HomeO=Yes|No) = 3/7
P(HomeO=No|No) = 4/7
P(HomeO=Yes|Yes) = 0
P(HomeO=No|Yes) = 1

P(Marital Status = Single|No) = 2/7
P(Marital Status = Divorced|No) = 1/7
P(Marital Status = Married|No) = 4/7
P(Marital Status = Single|Yes) = 2/3
P(Marital Status = Divorced|Yes) = 1/3
P(Marital Status = Married|Yes) = 0

For taxable income:
If class=No: sample mean=110, sample variance=2975
If class=Yes: sample mean=90, sample variance=25

P(Class = No) = 7/10
P(Class = Yes) = 3/10

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$$P(\mathbf{x}|y=c) = \prod_{i=1}^{d} P(x_i|y=c)$$

Test example:

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	120K	?

$$P(\mathbf{x}^*|y=c) = \prod_{i=1}^{d} P(x_i^*|y=c)$$

P(HomeO=Yes|No) = 3/7
P(HomeO=No|No) = 4/7
P(HomeO=Yes|Yes) = 0
P(HomeO=No|Yes) = 1

P(Marital Status = Single|No) = 2/7
P(Marital Status = Divorced|No) = 1/7
P(Marital Status = Married|No) = 4/7
P(Marital Status = Single|Yes) = 2/3
P(Marital Status = Divorced|Yes) = 1/3
P(Marital Status = Married|Yes) = 0

For taxable income:
If class=No: sample mean=110, sample variance=2975
If class=Yes: sample mean=90, sample variance=25

P(Class = No) = 7/10
P(Class = Yes) = 3/10

P(x*|Class=No) = P(HomeO=No|Class=No) × P(Status=Married|Class=No) × P(Income=120K|Class=No)

$$= \frac{4}{7} \times \frac{4}{7} \times 0.0072 = 0.0024$$

P(x*|Class=Yes) = P(HomeO=No|Class=Yes) × P(Status=Married|Class=Yes) × P(Income=120K|Class=Yes)

one of the conditional probabilities is 0, the entire expression is 0

$$= 1 \times 0 \times (1.2 \times 10^{-9}) = 0$$

$$P(\mathbf{x}^*|\text{No}) \times P(\text{No}) = 0.0024 \times 0.7 = 0.00168$$

$$> P(\mathbf{x}^*|\text{Yes}) \times P(\text{Yes}) = 0 \times 0.3 = 0$$

Therefore $P(\text{No}|\mathbf{x}^*) > P(\text{Yes}|\mathbf{x}^*)$ Class = No

Illustrative Example

- Consider the problem of predicting whether a loan applicant will repay his/her loan obligation (no cheat) or become delinquent (cheat).

Predefined categories

Example

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training data

$P(y|\mathbf{x})$

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	120K	?

Test application $\mathbf{x}^*$

- To classify the application, we need to compute the posterior probabilities $P(\text{Yes}|\mathbf{x}^*)$ and $P(\text{No}|\mathbf{x}^*)$
- If $P(\text{Yes}|\mathbf{x}^*) > P(\text{No}|\mathbf{x}^*)$ then classified as Yes
- Otherwise classified as No

Estimate Priors

- Class: #instances in class c

$$P(y=c) = \frac{|y=c|}{N}$$

#training instances

- e.g.,

P(No) = 0.7

P(Yes) = 0.3

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Estimate Conditional Probabilities for Discrete Features

#instances in class c, whose values of feature x_i are k

$$P(x_i = k|y=c) = \frac{|(x_i = k) \wedge (y=c)|}{|y=c|}$$

A specific value k of the feature x_i

$$P(\text{Status} = \text{Married}|\text{Cheat} = \text{No}) = \frac{\#(\text{Status} = \text{Married} \wedge \text{Cheat} = \text{No})}{\#(\text{Cheat} = \text{No})} = \frac{4}{7}$$

$$P(\text{Home Owner} = \text{Yes}|\text{Cheat} = \text{Yes}) = \frac{\#(\text{Home Owner} = \text{Yes} \wedge \text{Cheat} = \text{Yes})}{\#(\text{Cheat} = \text{Yes})} = \frac{0}{3} = 0$$

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Estimate Conditional Probabilities for Continuous Features

- For continuous features:
  - Probability density estimation (more details will be introduced in the 2nd half of the semester):
    - Assume the values of a feature given a class label follow a Guassian distribution, i.e., assume $P(x_i|y=c)$ is a Guassian distribution
    - Use training data in the class c to estimate parameters of distribution (e.g., mean μ and variance σ^2)
    - Once probability density function is known, we can use it to estimate the conditional probability
- For each class c, assume values of the feature x_i follow a Gaussian distribution:

$$P(x_i|y=c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i-\mu_{ic})^2}{2\sigma_{ic}^2}}$$

The mean of x_i of the training data in class c

The variance of x_i of the training data in class c

- Suppose there are N_c instances in class c, then

$$\mu_{ic} = \frac{1}{N_c}\sum_{j=1}^{N_c} x_{i_j} \qquad \sigma_{ic}^2 = \frac{1}{N_c - 1}\sum_{j=1}^{N_c}\left(x_{i_j} - \mu_{ic}\right)^2$$

Value of feature x_i of the j-th training data in class c

$$P(x_i|y=c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i-\mu_{ic})^2}{2\sigma_{ic}^2}}$$

Tid	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- For (Income, Cheat = No):
  - sample mean = 110
  - sample variance = 2975 (standard deviation = 54.54)

$$P(\text{Income}|\text{Cheat} = \text{No}) = \frac{1}{\sqrt{2\pi} \times 54.54} e^{-\frac{(\text{Income}-110)^2}{2 \times 2975}}$$

- The estimated Gaussian distribution for (Income, Cheat = No):

A function of x_i

$$P(\text{Income}|\text{Cheat} = \text{No}) = \frac{1}{\sqrt{2\pi} \times 54.54} e^{-\frac{(x_i-110)^2}{2 \times 2975}}$$

Tid	Home Owner	Marital Status	Taxable Income	Cheat
4	Yes	Married	120K	No

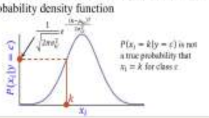
$$P(\text{Income} = 120|\text{No}) = \frac{1}{\sqrt{2\pi} \times 54.54} e^{-\frac{(120-110)^2}{2 \times 2975}} = 0.0072$$

Note: in practice, 0.0072 can be used as to approximate the conditional probability, but in theory it is not a true probability

Additional Notes

Probability density function $P(x_i|y=c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i-\mu_{ic})^2}{2\sigma_{ic}^2}}$

- The probability density function is continuous, the probability is defined as the area under the curve of the probability density function

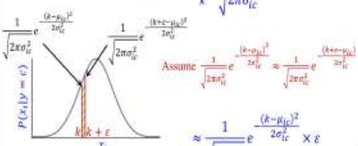


$P(x_i = k|y=c)$ is not a true probability that $x_i = k$ for class c

- Instead, we should compute

$$P(k \le x_i \le k + \varepsilon|y=c) = \int_k^{k+\varepsilon} \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(x_i-\mu_{ic})^2}{2\sigma_{ic}^2}} dx_i$$

Small positive constant



Assume $\frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}} \approx \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k+\varepsilon-\mu_{ic})^2}{2\sigma_{ic}^2}}$

$$\approx \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}} \times \varepsilon$$

- Since ε appears as a constant multiplicative factor for each class, it cancels out when comparing posterior probabilities $P(y=c|\mathbf{x})$ for each class
- E.g., consider binary classification and instance is represented by a single feature of continuous values

$$P(y=0|x=k) \quad \text{vs.} \quad P(y=1|x=k)$$

$$P(x=k|y=0)P(y=0) \quad \text{vs.} \quad P(x=k|y=1)P(y=1)$$

$$\frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{(k-\mu_0)^2}{2\sigma_0^2}} \times \varepsilon \times P(y=0) \quad \text{vs.} \quad \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(k-\mu_1)^2}{2\sigma_1^2}} \times \varepsilon \times P(y=1)$$

- Therefore, we can still apply the following equation to approximate the probability of $x_i = k$ for class c

$$P(x_i = k|y=c) = \frac{1}{\sqrt{2\pi\sigma_{ic}^2}} e^{-\frac{(k-\mu_{ic})^2}{2\sigma_{ic}^2}}$$

Laplace Estimate

- Alternative probability estimation (discrete features):

Original: $P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)|}{|y = c|}$

Laplace: $P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)| + 1}{|y = c| + n_i}$ ← #distinct values of x_i

$P(\text{Married}|\text{Yes}) = \frac{\#(\text{Married} \wedge \text{Yes})}{\#(\text{Yes})} = \frac{0}{3}$

$P(\text{Married}|\text{Yes}) = \frac{\#(\text{Married} \wedge \text{Yes}) + 1}{\#(\text{Yes}) + 3} = \frac{1}{6}$

The same to $P(\text{Single}|\text{Yes})$ and $P(\text{Divorced}|\text{Yes})$

Extreme case - no training data:

$P(\text{Single}|\text{Yes}) = P(\text{Married}|\text{Yes}) = P(\text{Divorced}|\text{Yes}) = \frac{1}{3}$

	Home Owner	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

M-estimate

- A more general estimation:

Original: $P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)|}{|y = c|}$

M-estimate: $P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)| + m \times p}{|y = c| + m}$ ← User-specified parameters

e.g., if prior information of $P(x_i = k|y = c)$ is available, then we can set p as the prior

For example, based on domain knowledge, you have the prior information:

Domain knowledge, not learned from data: $\hat{P}(\text{Single}|\text{Yes}) = \frac{1}{2}$ $\hat{P}(\text{Divorced}|\text{Yes}) = \frac{1}{3}$ $\hat{P}(\text{Married}|\text{Yes}) = \frac{1}{6}$

Extreme case - no training data:

$P(\text{Single}|\text{Yes}) = \frac{\#(\text{Single} \wedge \text{Yes}) + m \times \hat{P}(\text{Single}|\text{Yes})}{\#(\text{Yes}) + m} = \frac{m \times \hat{P}(\text{Single}|\text{Yes})}{m} = \hat{P}(\text{Single}|\text{Yes})$

27

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	120K	?

$m = 3$

$p = 1/3$ for all discrete features of class **Yes**

$p = 2/3$ for all discrete features of class **No**

P(HomO=Yes|No) = 3/7
P(HomO=No|No) = 4/7
P(HomO=Yes|Yes) = 0/3
P(HomO=No|Yes) = 1

P(Marital Status = Single|No) = 2/7
P(Marital Status = Divorced|No)=1/7
P(Marital Status = Married|No) = 4/7
P(Marital Status = Single|Yes) = 2/3
P(Marital Status = Divorced|Yes)= 1/3
P(Marital Status = Married|Yes) = 0/3

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

P(Class = No) = 7/10
P(Class = Yes) = 3/10

P(HomO=Yes|No) = ?
P(HomO=No|No) = ?
P(HomO=Yes|Yes) = ?
P(HomO=No|Yes) = ?

P(Marital Status = Single|No) = ?
P(Marital Status = Divorced|No) = ?
P(Marital Status = Married|No) = ?
P(Marital Status = Single|Yes) = ?
P(Marital Status = Divorced|Yes)=?
P(Marital Status = Married|Yes) = ?

M-estimate

$P(x_i = k|y = c) = \frac{|(x_i = k) \wedge (y = c)| + m \times p}{|y = c| + m}$

$P(\mathbf{x}^*|\text{Class} = \text{No}) = ?$ $P(\mathbf{x}^*|\text{Class} = \text{Yes}) = ?$

Tutorial

28

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	120K	?

$m = 3$

$p = 1/3$ for all discrete features of class **Yes**

$p = 2/3$ for all discrete features of class **No**

P(HomO=Yes|No) = (3+2)/(7+3) = 5/10
P(HomO=No|No) = (4+2)/(7+3) = 6/10
P(HomO=Yes|Yes) = (0+1)/(3+3) = 1/6
P(HomO=No|Yes) = (3+1)/(3+3) = 4/6

P(Status=Single|No) = (2+2)/(7+3) = 4/10
P(Status=Divorced|No)= (1+2)/(7+3) = 3/10
P(Status=Married|No) = (4+2)/(7+3) = 6/10
P(Status=Single|Yes) = (2+1)/(3+3) = 3/6
P(Status=Divorced|Yes) = (1+1)/(3+3) = 2/6
P(Status=Married|Yes) = (0+1)/(3+3) = 1/6

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

P(Class = No) = 7/10
P(Class = Yes) = 3/10

$\frac{|(x_i = k) \wedge (y = c)| + m \times p}{|y = c| + m}$

$\sum_k P(x_i = k|y = c) = 1$

$\hat{P}(\text{HomO} = \text{Yes}|\text{No}) = \frac{P(\text{HomO} = \text{Yes}|\text{No})}{P(\text{HomO} = \text{Yes}|\text{No}) + P(\text{HomO} = \text{No}|\text{No})} = \frac{\frac{5}{10}}{\frac{5}{10} + \frac{6}{10}} = \frac{5}{11}$

$\hat{P}(\text{HomO} = \text{No}|\text{No}) = \frac{P(\text{HomO} = \text{No}|\text{No})}{P(\text{HomO} = \text{Yes}|\text{No}) + P(\text{HomO} = \text{No}|\text{No})} = \frac{\frac{6}{10}}{\frac{5}{10} + \frac{6}{10}} = \frac{6}{11}$

9

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	120K	?

$m = 3$

$p = 1/3$ for all discrete features of class **Yes**

$p = 2/3$ for all discrete features of class **No**

P(HomO=Yes|No) = (3+2)/(7+3) = 5/10
P(HomO=No|No) = (4+2)/(7+3) = 6/10
P(HomO=Yes|Yes) = (0+1)/(3+3) = 1/6
P(HomO=No|Yes) = (3+1)/(3+3) = 4/6

P(Status=Single|No) = (2+2)/(7+3) = 4/10
P(Status=Divorced|No)= (1+2)/(7+3) = 3/10
P(Status=Married|No) = (4+2)/(7+3) = 6/10
P(Status=Single|Yes) = (2+1)/(3+3) = 3/6
P(Status=Divorced|Yes) = (1+1)/(3+3) = 2/6
P(Status=Married|Yes) = (0+1)/(3+3) = 1/6

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

P(Class = No) = 7/10
P(Class = Yes) = 3/10

$m = 3$

$p = 1/3$ for all discrete features of class **Yes**

$p = 2/3$ for all discrete features of class **No**

P(HomO=Yes|No) = (5/10)/(5/10+6/10) = 5/11
P(HomO=No|No) = (6/10)/(5/10+6/10) = 6/11
P(HomO=Yes|Yes) = (1/6)/(1/6+4/6) = 1/5
P(HomO=No|Yes) = (4/6)/(1/6+4/6) = 4/5

P(Status=Single|No) = (4/10)/(4/10+3/10+6/10) = 4/13
P(Status=Divorced|No)= (3/10)/(4/10+3/10+6/10) = 3/13
P(Status=Married|No) = (6/10)/(4/10+3/10+6/10) = 6/13
P(Status=Single|Yes) = (2+1)/(3+3) = 3/6
P(Status=Divorced|Yes) = (1+1)/(3+3) = 2/6
P(Status=Married|Yes) = (0+1)/(3+3) = 1/6

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

P(Class = No) = 7/10
P(Class = Yes) = 3/10

9

Home Owner	Marital Status	Taxable Income	Cheat
No	Married	120K	?

$m = 3$

$p = 1/3$ for all discrete features of class **Yes**

$p = 2/3$ for all discrete features of class **No**

P(HomO=Yes|No) = 5/11
P(HomO=No|No) = 6/11
P(HomO=Yes|Yes) = 1/5
P(HomO=No|Yes) = 4/5

P(Status=Single|No) = 4/13
P(Status=Divorced|No) = 3/13
P(Status=Married|No) = 6/13
P(Status=Single|Yes) = 3/6
P(Status=Divorced|Yes) = 2/6
P(Status=Married|Yes) = 1/6

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

P(Class = No) = 7/10
P(Class = Yes) = 3/10

$P(\mathbf{X}|\text{Class=No}) = P(\text{HomO=No} \mid \text{Class=No}) \times P(\text{Status=Married} \mid \text{Class=No}) \times P(\text{Income=120K} \mid \text{Class=No}) = 6/11 \times 6/13 \times 0.0072 = 0.0018$

$P(\mathbf{X}|\text{Class=Yes}) = P(\text{HomO=No} \mid \text{Class=Yes}) \times P(\text{Status=Married} \mid \text{Class=Yes}) \times P(\text{Income=120K} \mid \text{Class=Yes}) = 4/5 \times 1/6 \times 1.2 \times 10^{-9} = 1.6 \times 10^{-10}$

$P(\mathbf{X}|\text{No}) \times P(\text{No}) > P(\mathbf{X}|\text{Yes}) \times P(\text{Yes})$

Thus, $P(\text{No}|\mathbf{X}) > P(\text{Yes}|\mathbf{X})$

Cheat = No

10

Week 4

Bayesian Classifiers: Recall

- Estimate $P(y|\boldsymbol{x})$ via Bayes rule:

$$P(y=c|\boldsymbol{x}) = \frac{P(\boldsymbol{x}, y=c)}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}|y=c)P(y=c)}{P(\boldsymbol{x})}$$

Prior, Post

- Make predictions based on maximum posterior:

$$y^* = c^* \text{ if } c^* = \arg\max_c P(y=c|\boldsymbol{x}) \quad \text{the 0/1 loss}$$

$$\Updownarrow$$

$$y^* = c^* \text{ if } c^* = \arg\max_c P(\boldsymbol{x}|y=c)P(y=c)$$

Easy to estimate

Still difficult to estimate. $\boldsymbol{x}$ contains many input variables. Some are discrete, and others are continuous.

Naïve Bayes Classifier

- To make estimation of $P(\boldsymbol{x}|y)$ from training data tractable, Naïve Bayes classifiers assume features are conditionally independent given the class label:

$$P(\boldsymbol{x}|y=c) = \prod_{i=1}^{d} P(x_i|y=c)$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]$

- The conditional independence assumption may not hold in practice.
  - Correlated features can degrade the performance.

An Example

- Suppose the probability of a person having a specific disease D is 50%.
- There are two medical tests, T_1 and T_2, of binary values (positive or negative). The outcomes of T_2 are perfectly positively correlated with T_1 if a person has the disease, but are independent of T_1 if the person does not have the disease
  - When a person has the disease, the outcomes of T_1 and T_2 are both positive (or negative).
- If a person has the disease, the probabilities of tests T_1 and T_2 being negative are 40%, respectively.
- If a person does not have the disease, the probabilities of T_1 and T_2 being negative are 60% and 65%, respectively.
- If the two tests T_1 and T_2 are both negative for a particular patient, diagnose whether the patient has the disease?

Variables Definition

- Let X_1 denote the outcome of T_1
  - $X_1 = 1$: positive
  - $X_1 = 0$: negative
- Let X_2 denote the outcome of T_2
  - $X_2 = 1$: positive
  - $X_2 = 0$: negative
- Let Y denote whether a person has the disease D
  - $Y = 1$: yes
  - $Y = 0$: no

Probabilities

- Suppose that the probability of a person having a specific disease D is 50%.

$$P(Y=0) = 50\% \text{ and } P(Y=1) = 50\%$$

- The outcomes of T_2 are perfectly positively correlated with T_1 if a person has the disease.
  - Given that a person has the disease, if the outcome of T_1 is positive then the outcome of T_2 is always positive, and if the outcome of T_1 is negative then the outcome of T_2 is always negative.

$$P(X_1=1, X_2=1|Y=1) = \frac{|(X_1=1) \wedge (X_2=1) \wedge (Y=1)|}{|Y=1|} = \frac{|(X_1=1) \wedge (Y=1)|}{|Y=1|} = P(X_1=1|Y=1)$$

T_1, T_2: + +, − −

$$P(X_1=0, X_2=0|Y=1) = P(X_1=0|Y=1)$$

- The outcomes of T_2 are perfectly positively correlated with T_1 if a person has the disease, but are independent of T_1 if the person does not have the disease.
  - Given that a person does not have the disease, the outcomes of T_1 and T_2 are independent.

$$P(X_1=1, X_2=1|Y=0) = P(X_1=1|Y=0)P(X_2=1|Y=0)$$
$$P(X_1=1, X_2=0|Y=0) = P(X_1=1|Y=0)P(X_2=0|Y=0)$$
$$P(X_1=0, X_2=1|Y=0) = P(X_1=0|Y=0)P(X_2=1|Y=0)$$
$$P(X_1=0, X_2=0|Y=0) = P(X_1=0|Y=0)P(X_2=0|Y=0)$$

  - The conditional independence assumption in Naïve Bayes Classifiers holds when a person does not have the disease.

Conditional Probability

- If a person has the disease, the probabilities of tests T_1 and T_2 being negative are 40%, respectively.

$$P(X_1=0|Y=1) = 0.4 \longrightarrow P(X_1=1|Y=1) = 0.6$$
$$P(X_2=0|Y=1) = 0.4 \longrightarrow P(X_2=1|Y=1) = 0.6$$

- If a person does not have the disease, the probabilities of T_1 and T_2 being negative are 60% and 65%, respectively.

$$P(X_1=0|Y=0) = 0.6 \longrightarrow P(X_1=1|Y=0) = 0.4$$
$$P(X_2=0|Y=0) = 0.65 \longrightarrow P(X_2=1|Y=0) = 0.35$$

- Given a patient with $X_1 = 0$, and $X_2 = 0$, to estimate

$$P(Y=0|X_1=0, X_2=0) \text{ and } P(Y=1|X_1=0, X_2=0)$$

Using a Naïve Bayes Classifier

$$P(Y=0|X_1=0, X_2=0) = \frac{P(X_1=0, X_2=0|Y=0)P(Y=0)}{P(X_1=0, X_2=0)} \quad \text{Bayes rule}$$

Using Naïve Bayes assumption

$$= \frac{P(X_1=0|Y=0)P(X_2=0|Y=0)P(Y=0)}{P(X_1=0, X_2=0)}$$

Prediction: $Y = 0$

$$= \frac{0.6 \times 0.65 \times 0.5}{P(X_1=0, X_2=0)} = \frac{0.195}{P(X_1=0, X_2=0)}$$

$$P(Y=1|X_1=0, X_2=0) = \frac{P(X_1=0, X_2=0|Y=1)P(Y=1)}{P(X_1=0, X_2=0)} \quad \text{Bayes rule}$$

Using Naïve Bayes assumption

$$= \frac{P(X_1=0|Y=1)P(X_2=0|Y=1)P(Y=1)}{P(X_1=0, X_2=0)}$$

$$= \frac{0.4 \times 0.4 \times 0.5}{P(X_1=0, X_2=0)} = \frac{0.08}{P(X_1=0, X_2=0)}$$

10

Features are Correlated When $Y = 1$

- However, because X_1 and X_2 are perfectly positively correlated when $Y = 1$,

$$P(X_1=0, X_2=0|Y=1) = P(X_1=0|Y=1) = 0.4$$

$$P(Y=1|X_1=0, X_2=0) = \frac{P(X_1=0, X_2=0|Y=1)P(Y=1)}{P(X_1=0, X_2=0)}$$

Perfectly correlated

$$= \frac{P(X_1=0|Y=1)P(Y=1)}{P(X_1=0, X_2=0)} = \frac{0.4 \times 0.5}{P(X_1=0, X_2=0)}$$

- X_1 and X_2 are independent if the person does not have the disease.

$$P(Y=0|X_1=0, X_2=0) = \frac{P(X_1=0|Y=0)P(X_2=0|Y=0)P(Y=0)}{P(X_1=0, X_2=0)}$$

$$= \frac{0.195}{P(X_1=0, X_2=0)}$$

(Same) Prediction: $Y = 1$

Bayesian Belief Networks

- A more general approach to modeling the independence and conditional independence among $\boldsymbol{x}$ and y, s.t. the computation of $P(\boldsymbol{x}, y) = P(\boldsymbol{x}|y)P(y)$ is tractable.
  - Suppose all features are discrete (if there are both continuous and discrete features, the estimation is much more difficult).
- Representation: a Bayesian network provides a graphical representation of the probabilistic relationships among a set of random variables including features and output class.
- Two key elements:
  - A directed acyclic graph (DAG) encoding the dependence relationships among a set of variables
  - A probability table associating each node to its immediate parent nodes

Characterized

A DAG Example

- Consider three random variables A, B and C, where A and B are independent variables and each has a direct influence on a third variable, C

$P(A|B) = P(A)$

$P(A, B) = P(A)P(B)$

node → variable

If there is a directed arc from B to C, then B is the parent of C, and C is the child of B

Directed arc → dependence relationship (C depends on A and B)

Another DAG Example

no path from B to D

If there is a directed path from B to B, then D is an ancestor of B and B is a descendant of D

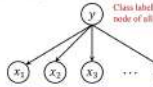
Non-descendants of B

Parent of B

no path from B to A

A Special Case: Naïve Bayes

- A Naïve Bayes classifier can be represented using a special DAG:



Once a class label is given, all the features are conditionally independent to each other, because each feature is a non-descendant to any other features

$$P(x_1, x_2, \ldots, x_d|y=c) = \prod_{i=1}^{d} P(x_i|y=c)$$

DAG: Conditional Independence

- Property (conditional independence): a node in a Bayesian network is conditionally independent of its non-descendants, if its parents are known.

Non-descendants of B

Parent of B

If C is observed, then B is conditionally independent of A and D

$$P(B|C, A, D) = P(B|C)$$

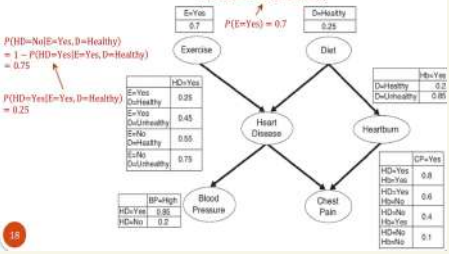
$$P(B, A, D|C) = P(B|C)P(D|C)P(A|C)$$

BBN Representation

- Besides the conditional independence conditions imposed by the network topology, each node is also associated with a probability table.
  - If a node X does not have any parents, then the table contains only the prior probability $P(X)$
  - If a node X has only one parent, Z, then the table contains the conditional probability $P(X|Z)$.
  - If a node X has multiple parents, $\{Z_1, Z_2, \ldots, Z_k\}$, then the table contains the conditional probability $P(X|Z_1, Z_2, \ldots, Z_k)$

BBN Representation: Example

A Bayesian network for modeling patients with heart disease or heartburn problems



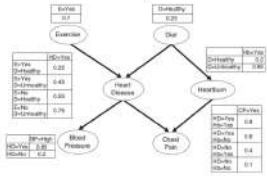
BBN Model Building

- Two steps in the training phase:
  - Creating the structure of the network, i.e., DAG
    - Network topology can be obtained by encoding the subjective knowledge of domain experts
    - Or can be learned from data (structure learning) --- still an open problem
  - Estimating the probability values in the table associated with each node
    - Counting based on the definition of the corresponding probabilities
- Note: in this module we only focus on how to use a BBN to make predictions (or inference)

Suppose Heart Disease is our target variable to make prediction on (i.e., output) the other variables are input features, whose values can be observed or missing.

Inference: Example 1

- Without any additional information, to determine whether the person is likely to have heart disease.

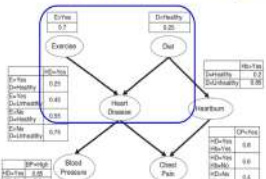


$P(\text{HD=No})$ *vs.* $P(\text{HD=Yes})$

$P(\text{HD=Yes})$? $\quad P(\text{HD=Yes}) = \sum_{\alpha}\sum_{\beta} P(\text{HD=Yes}, \text{E}=\alpha, \text{D}=\beta)$

Sum Rule

$\alpha = \{\text{Yes, No}\} \quad \beta = \{\text{Healthy, Unhealthy}\}$



$$P(\text{HD=Yes}) = \sum_{\alpha}\sum_{\beta} P(\text{HD=Yes}, \text{E}=\alpha, \text{D}=\beta)$$

Product Rule

$$= \sum_{\alpha}\sum_{\beta} P(\text{HD=Yes}|\text{E}=\alpha, \text{D}=\beta)P(\text{E}=\alpha, \text{D}=\beta)$$

$$= \sum_{\alpha}\sum_{\beta} P(\text{HD=Yes}|\text{E}=\alpha, \text{D}=\beta)P(\text{E}=\alpha)P(\text{D}=\beta)$$

Independence

$\alpha = \{\text{Yes, No}\} \quad \beta = \{\text{Healthy, Unhealthy}\}$

$$P(\text{HD=Yes}) = \sum_{\alpha}\sum_{\beta} P(\text{HD=Yes}|\text{E}=\alpha, \text{D}=\beta)P(\text{E}=\alpha)P(\text{D}=\beta)$$

$$= P(\text{HD=Yes}|\text{E=Yes, D=Healthy})P(\text{E=Yes})P(\text{D=Healthy})$$
$$+ P(\text{HD=Yes}|\text{E=Yes, D=Unhealthy})P(\text{E=Yes})P(\text{D=Unhealthy})$$
$$+ P(\text{HD=Yes}|\text{E=No, D=Healthy})P(\text{E=No})P(\text{D=Healthy})$$
$$+ P(\text{HD=Yes}|\text{E=No, D=Unhealthy})P(\text{E=No})P(\text{D=Unhealthy})$$

Look up probability tables

$$= 0.25 \times 0.7 \times 0.25 + 0.45 \times 0.7 \times 0.75 + 0.55 \times 0.3 \times 0.25 + 0.75 \times 0.3 \times 0.75 = 0.49$$

$P(\text{HD=Yes}) = 0.49$

$P(\text{HD=No}) = 1 - P(\text{HD=Yes}) = 0.51$

- Therefore, the person has a slightly higher chance of not getting the heart disease.

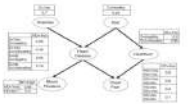
Inference: Example 2

- If the person has high blood pressure, chest pain and heartburn, but does regular exercise and eats a healthy diet, to diagnose whether the patient has heart disease:

$P(\text{HD=Yes}|\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})$

vs.

$P(\text{HD=No}|\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})$



$$P(\text{HD=Yes}|\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes}) = \frac{P(\text{HD=Yes, BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})}{P(\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})}$$

How to estimate the joint probability in the numerator?

Keep in mind that the goal is to rewrite the joint probability into an equivalent form, such that for all the probabilities in the rewritten equivalent form, their values can be found from the tables in the BBN.

There are many ways to rewrite the above joint probability, e.g.,
$P(\text{D, E} \mid \text{HD, BP, Hb, CP})P(\text{HD, BP, Hb, CP})$, or
$P(\text{D, BP, CP} \mid \text{E, Hb, HD})P(\text{E, Hb, HD})$, many others

Which one is useful for the joint probability simplification?

From the BBN, we found that the variables BP and CP are child nodes of HD and Hb, but not a parent node for any other variables. That means we could not find any conditional probabilities in the tables, which involves BP and CP in the condition, like $P(\text{D, E} \mid \text{HD, BP, Hb, CP})$. In other words, if in the rewritten equivalent form, there is a probability where BP or (and) CP is (are) in the condition, we still have to further rewrite it such that BP and CP are not in the condition of any conditional probability term.

The above analysis motivates us to transform the joint probability (numerator) using the following form based on the product rule:

$$P(\text{HD=Yes, BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})$$
$$= P(\text{BP=High, CP=Yes} \mid \text{HD=Yes, Hb=Yes, E=Yes, D=Healthy})P(\text{HD=Yes, Hb=Yes, E=Yes, D=Healthy})$$

Not in the condition of the conditional probability

Denote by $\mathbf{U} = \{\text{BP, CP}\}$ and $\mathbf{V} = \{\text{HD, Hb, E, D}\}$

$P(\mathbf{U}, \mathbf{V}) = P(\mathbf{U}|\mathbf{V})P(\mathbf{V})$

$$P(\text{HD=Yes}|\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes}) = \frac{P(\text{HD=Yes, BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})}{P(\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})}$$

Denote by $\mathbf{U} = \{\text{BP, CP}\}$ and $\mathbf{V} = \{\text{HD, Hb, E, D}\}$

$P(\mathbf{U}, \mathbf{V}) = P(\mathbf{U}|\mathbf{V})P(\mathbf{V})$

$$P(\text{BP=High, CP=Yes} \mid \text{HD=Yes, Hb=Yes, E=Yes, D=Healthy})P(\text{HD=Yes, Hb=Yes, E=Yes, D=Healthy})$$

Slide 29 Slide 30

Recall: if **A** and **B** are conditionally independent given **C**, we have

$P(\mathbf{A}|\mathbf{B}, \mathbf{C}) = P(\mathbf{A}|\mathbf{C})$ or $P(\mathbf{A}, \mathbf{B}|\mathbf{C}) = P(\mathbf{A}|\mathbf{C})P(\mathbf{B}|\mathbf{C})$

$P(\text{BP=High, CP=Yes} \mid \text{HD=Yes, Hb=Yes, E=Yes, D=Healthy})$

- HD and Hb are parents of BP and CP
- E and D are non-descendants of BP and CP
- Therefore, BP and CP are conditionally independent of E and D given HD and Hb

$P(\text{BP=High, CP=Yes} \mid \text{HD=Yes, Hb=Yes})$

- HD and Hb are parents of CP, and BP is a non-descendant of CP
- Therefore, BP and CP are conditionally independent given HD and Hb

$P(\text{BP=High} \mid \text{HD=Yes, Hb=Yes})P(\text{CP=Yes} \mid \text{HD=Yes, Hb=Yes})$

- HD is parent of BP
- Hb is a non-descendant of BP
- Therefore, BP and Hb are conditionally independent given HD

$P(\text{BP=High} \mid \text{HD=Yes})P(\text{CP=Yes} \mid \text{HD=Yes, Hb=Yes})$

$= 0.85 \times 0.8 = 0.68$

$P(\text{HD=Yes, Hb=Yes, E=Yes, D=Healthy})$

Denote by $\mathbf{U} = \{\text{HD, Hb}\}$ and $\mathbf{V} = \{\text{E, D}\}$, $P(\mathbf{U}, \mathbf{V}) = P(\mathbf{U}|\mathbf{V})P(\mathbf{V})$

$P(\text{HD=Yes, Hb=Yes} \mid \text{E=Yes, D=Healthy})P(\text{E=Yes, D=Healthy})$

Independence

Given E and D, HD and Hb are conditionally independent

$P(\text{HD=Yes} \mid \text{E=Yes, D=Healthy})P(\text{Hb=Yes} \mid \text{E=Yes, D=Healthy})$

Given D, Hb and E are conditionally independent

$P(\text{HD=Yes} \mid \text{E=Yes, D=Healthy})P(\text{Hb=Yes} \mid \text{D=Healthy})$

$= P(\text{HD=Yes} \mid \text{E=Yes, D=Healthy})P(\text{Hb=Yes} \mid \text{D=Healthy})P(\text{E=Yes})P(\text{D=Healthy})$

$= 0.25 \times 0.2 \times 0.7 \times 0.25 = 0.00875$

$$P(\text{HD=Yes} \mid \text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes}) = \frac{P(\text{BP=High, CP=Yes} \mid \text{HD=Yes, Hb=Yes, E=Yes, D=Healthy})P(\text{HD=Yes, Hb=Yes, E=Yes, D=Healthy})}{P(\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})} = \frac{0.68 \times 0.00875}{0.00595} = \frac{0.00595}{P(\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})}$$

$$P(\text{HD=No} \mid \text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes}) = \frac{P(\text{BP=High, CP=Yes} \mid \text{HD=No, Hb=Yes, E=Yes, D=Healthy})P(\text{HD=No, Hb=Yes, E=Yes, D=Healthy})}{P(\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})} = \frac{0.01 \times 0.02625}{0.0021} = \frac{0.0021}{P(\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})}$$

$P(\text{HD=Yes}|\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})$

$>$

$P(\text{HD=No}|\text{BP=High, Hb=Yes, CP=Yes, D=Healthy, E=Yes})$

The person has a higher chance of getting the heart disease.

Inference: Example 3

- If the person has high blood pressure, but exercises regularly and eats a healthy diet, to diagnose about heart disease (estimate the probabilities).

$P(\text{HD=Yes}|\text{BP=High, D=Healthy, E=Yes})$

vs.

$P(\text{HD=No}|\text{BP=High, D=Healthy, E=Yes})$

Tutorial

Hint: Using BBNs for Inference

- Given a BBN, and an inference (prediction) task:
  1. Translate the problem into a probabilistic language, i.e., what probabilities to be estimated?
  2. If the probabilities to be estimated cannot be obtained from the probability tables of the BBN, then
     - A. Identify a subgraph which captures the dependence between input variables (features) and the output variable (class)
     - B. Based on the network topology, apply product rule, sum rule and the properties of conditional independence and independence to induce equivalent forms of the probabilities until all probabilities can be found from the probability tables

Bayesian Belief Networks: Summary

- BBNs use a (directed) graphical model to model dependence among variables
  - Other directed graphical models: Hidden Markov Models, Dynamic Bayesian Networks, etc.
  - Undirected graphical models: e.g., Markov Random Fields, Conditional Random Fields, etc.
- Network structure construction is difficult
  - Use domain knowledge – not complete, may not accurate
  - Learn structure from data – computationally expensive, greedy algorithm → not optimal

week5 decision tree

Illustrative Example

- Consider the problem of predicting whether a loan applicant will repay his/her loan obligation (no cheat) or become delinquent (cheat).

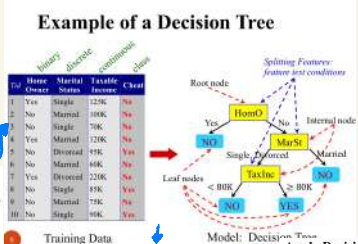
Example

Predefined categories

Decision Tree Classifiers: Summary

- Easy to interpret
- Efficient in both training and testing
- Effective for datasets with a lot of categorical features
- Used as a base classifier in many ensemble learning approaches (will be introduced in the 2nd half)

Example of a Decision Tree



Model: Decision Tree

Motivation of Decision Trees

- Suppose a new applicant submits a loan application. How do we decide whether to approve or reject the application?
- To pose a series of questions about the profile of the applicant:
  - Whether the applicant is a home owner? If yes, then he/she may repay the loan obligation with a high probability.
  - After that, we may ask a follow-up question: what is the applicant's marital status? …

Bayesian classifiers: $P(y|\mathbf{x})$

$f(\mathbf{x}) = y$

Decision tree

Training Data

Induction (induce a tree)

Deduction (apply the tree)

Apply Decision Tree to Test Data

High-level Algorithm

Initial tree

Default class: **Don't Cheat** (most of the borrowers successfully repaid their loans)

Home Owner test condition is selected

Marital Status test condition is selected

Taxable Income test condition is selected

Another Example of Decision Tree

There could be more than one tree that fits the same data!

Apply Model to Test Data

Decision Tree Induction Algorithms

- CART, ID3, C4.5, etc.
- High-level basic idea:
  - Given a new application, the goal of asking a series of questions (checking properties of the profile) one by one is to find similar profiles (applicants) in the past until it is confident to make a decision based on the labels (whether cheat or no cheat) of the similar applicants.
  - A tree can be learned by splitting training data into subsets based on outcomes of a feature test.
  - This process is recursively applied on each derived subset until the subset at a node has all the same class or there is no improvement for prediction.

Tree induction algorithm

Induction (induce a tree)

Decision Tree

Deduction (apply the tree)

Tree Induction

- Determine how to split the data
  - How to specify the feature test condition?
  - How to determine the best split?
- Determine when to stop splitting

Stopping Criteria for Tree Induction

- Stop expanding a node when all the data instances belong to the same class
  - Ideal case but not always possible
- Stop expanding a node when all the data instances have similar feature values
- Early termination
  - Useful to avoid the overfitting issue (will be introduced next week)

Tree Induction

- Greedy strategy
  - Split the records based on a feature test that optimizes certain criterion
- Issues
  - Determine how to split the records
    - How to specify the feature test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to Specify Test Condition?

- Depends on feature types
  - Discrete
  - Continuous
- Depends on number of ways to split
  - Binary split
  - Multi-way split

Splitting Based on Discrete Features (more than two distinct values)

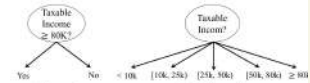
- Multi-way split: Use as many partitions as distinct values

- Binary split: Divide values into two subsets. Need to find optimal partitioning

Splitting Based on Binary Features

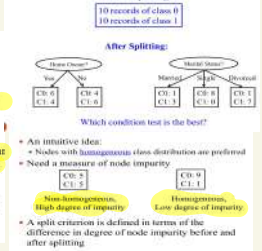
- Generate two potential outcomes

Splitting Based on Continuous Features



- Consider all possible splits and find the best cut
- Can be very computationally intensive

How to Determine the Best Split



- An intuitive idea:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity

Non-homogeneous, High degree of impurity

Homogeneous, Low degree of impurity

- A split criterion is defined in terms of the difference in degree of node impurity before and after splitting

High-level Algorithm: Summary

Let D_t be the set of training records that reach a node t

General Procedure:

- If D_t contains records that belong to the same class y_t, then t is a leaf node labeled as y_t (e.g., Yes or No)
- Otherwise if D_t is an empty set, then t is a leaf node labeled by the default class, y_d (e.g., No)
- Otherwise if D_t contains records that belong to more than one class, then a feature is selected to conduct condition test to split the data into smaller subsets.
  - A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes
  - Recursively apply the procedure to each subset

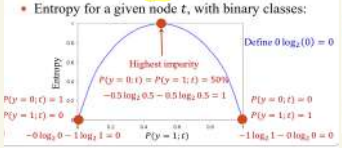
Measure of Impurity: Entropy

- Entropy at a given node t:

$$\text{Entropy}(t) = -\sum_c P(y=c\,|\,t)\log_2 P(y=c\,|\,t)$$

(The probability of class c at the node t)

- Entropy for a given node t, with binary classes:



Entropy Properties

- Entropy at a given node t:

$$\text{Entropy}(t) = -\sum_c P(y=c;t)\log_2 P(y=c;t)$$

- Maximum: $\log_2 C$ (Total number of all possible values of y, i.e., #classes) when records are equally distributed among all classes
- Minimum: 0 when all records belong to one class

Splitting Based on Entropy (cont.)

- Gain Ratio:

$$\Delta_{\text{GainR}} = \frac{\Delta_{\text{info}}}{\text{SplitINFO}}$$

(Penalty on large number of small partitions)

Maximum: $\log_2 P$

where $\text{SplitINFO} = -\sum_{i=1}^{P}\frac{n_i}{n}\log_2\left(\frac{n_i}{n}\right)$

- Parent node t of n instances is split into P partitions (children), n_i is the number of instances in partition i
- Higher entropy partitioning (large number of small partitions) is penalized!

Examples of Computing Entropy

$$\text{Entropy}(t) = -\sum_c P(y=c;t)\log_2 P(y=c;t)$$

Y_1	0
Y_2	6

$P(Y_1)=\frac{0}{6}=0 \quad P(Y_2)=\frac{6}{6}=1$

Entropy $= -0\log_2(0) - 1\log_2(1) = -0-0=0$

Y_1	1
Y_2	5

$P(Y_1)=\frac{1}{6} \quad P(Y_2)=\frac{5}{6}$

Entropy $= -\left(\frac{1}{6}\right)\log_2\left(\frac{1}{6}\right)-\left(\frac{5}{6}\right)\log_2\left(\frac{5}{6}\right)=0.65$

Y_1	2
Y_2	4

$P(Y_1)=\frac{2}{6} \quad P(Y_2)=\frac{4}{6}$

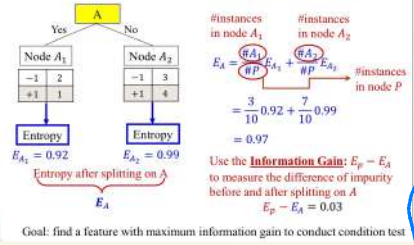
Entropy $= -\left(\frac{2}{6}\right)\log_2\left(\frac{2}{6}\right)-\left(\frac{4}{6}\right)\log_2\left(\frac{4}{6}\right)=0.92$

A motivating example of Information Gain

Node P

−1	5
+1	5

Entropy before splitting: Entropy $E_p = 1$



$E_A = \frac{\#A_1}{\#P}E_{A_1} + \frac{\#A_2}{\#P}E_{A_2}$ (#instances in node A_1, #instances in node A_2, #instances in node P)

$= \frac{3}{10}0.92 + \frac{7}{10}0.99$

$= 0.97$

Use the **Information Gain**: $E_p - E_A$ to measure the difference of impurity before and after splitting on A

$E_p - E_A = 0.03$

Goal: find a feature with maximum information gain to conduct condition test

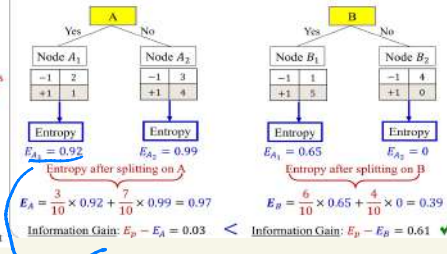
Refer to Question 1 in tutorial for practice

A motivating example of Information Gain

Node P

−1	5
+1	5

Entropy before splitting: Entropy $E_p = 1$



$E_A = \frac{3}{10}\times 0.92 + \frac{7}{10}\times 0.99 = 0.97$

Information Gain: $E_p - E_A = 0.03$ <

$E_B = \frac{6}{10}\times 0.65 + \frac{4}{10}\times 0 = 0.39$

Information Gain: $E_p - E_B = 0.61$ ✓

Information Gain: Motivation

- Recall: a split criterion should be defined in terms of the difference in degree of node impurity before and after splitting
- Information Gain:

Δ_{info} = Entropy(parent node) − Entropy(children nodes)

- Choose a feature whose condition test maximizes the gain (minimizes the weighted average impurity measures of the child nodes)

Information Gain: Definition

- Suppose a parent node t is split into P partitions (children)
- Information Gain:

$$\Delta_{\text{info}} = \text{Entropy}(t) - \sum_{j=1}^{P}\frac{n_j}{n}\text{Entropy}(j)$$

(n: Number of examples at node t; n_j: Number of examples at child j)

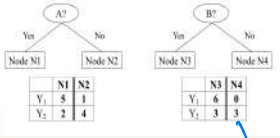
- To choose a feature whose condition test maximizes the gain (minimizes the weighted average impurity measures of the children nodes)

$$-\frac{2}{3}\log_2\left(\frac{2}{3}\right)-\left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right)$$

Information Gain: Practice

- Suppose features A and B are binary
- Based on Information Gain, which feature should be selected to split data?

	Parent
Y_1	6
Y_2	6



	N1	N2
Y_1	5	1
Y_2	2	4

	N3	N4
Y_1	6	0
Y_2	3	3

Entropy(Parent) $= -\left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right)-\left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right)=1$

Entropy(N1) $= -\left(\frac{5}{7}\right)\log_2\left(\frac{5}{7}\right)-\left(\frac{2}{7}\right)\log_2\left(\frac{2}{7}\right)=0.8631$

Entropy(N2) $= -\left(\frac{1}{5}\right)\log_2\left(\frac{1}{5}\right)-\left(\frac{4}{5}\right)\log_2\left(\frac{4}{5}\right)=0.7219$

Entropy(Split$_A$) $= \left(\frac{7}{12}\right)\times 0.8631 + \left(\frac{5}{12}\right)\times 0.7219 = 0.8043$ **Split on A**

Entropy(N3) $= -\left(\frac{3}{9}\right)\log_2\left(\frac{3}{9}\right)-\left(\frac{6}{9}\right)\log_2\left(\frac{6}{9}\right)=0.9183$

Entropy(N4) $= -\left(\frac{0}{3}\right)\log_2\left(\frac{0}{3}\right)-\left(\frac{3}{3}\right)\log_2\left(\frac{3}{3}\right)=0$

Entropy(Split$_B$) $= \left(\frac{9}{12}\right)\times 0.9183 + \left(\frac{3}{12}\right)\times 0 = 0.6887$ **Split on B**

Entropy(Parent) $= -\left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right)-\left(\frac{6}{12}\right)\log_2\left(\frac{6}{12}\right)=1$

Split on A

Entropy(Split$_A$) $= \left(\frac{7}{12}\right)\times 0.8631 + \left(\frac{5}{12}\right)\times 0.7219 = 0.8043$

Split on B

Entropy(Split$_B$) $= \left(\frac{9}{12}\right)\times 0.9183 + \left(\frac{3}{12}\right)\times 0 = 0.6887$

$\Delta_{\text{info}}(A) = 1 - 0.8043 = 0.1957$

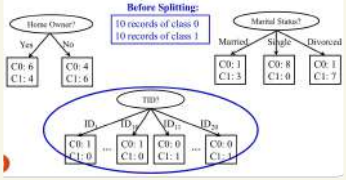
$\Delta_{\text{info}}(B) = 1 - 0.6887 = 0.3113$

Choose B to conduct condition test to split data

y gain ratio?

Information Gain: Limitation

- Disadvantage: tends to prefer splits that result in large number of partitions, each being small but pure



An Example

```
>>> from sklearn import tree
>>> ...
    Data preprocessing
>>> dtC = tree.DecisionTreeClassifier()
>>> dtC.fit(X, y)
>>> pred= dtC.predict(X_t)
```

A: $P(N_1)=\frac{7}{12}$, $P(N_2)=\frac{5}{12}$

B: $P(N_3)=\frac{9}{12}$, $P(N_4)=\frac{3}{12}$

entropy

$E_{N1} = -\frac{5}{7}\log_2\left(\frac{5}{7}\right)-\frac{2}{7}\log_2\left(\frac{2}{7}\right) = 0.863$

$E_{N2} = -\frac{1}{5}\log_2\left(\frac{1}{5}\right)-\frac{4}{5}\log_2\left(\frac{4}{5}\right) = 0.722$

$E_{N3} = -\frac{6}{9}\log_2\frac{6}{9}-\left(\frac{3}{9}\log_2\frac{3}{9}\right) = 0.92$

$E_{N4} = -0\log_2 0-(1\log_2 1) = 0$

$\left(\frac{7}{12}\times 0.863\right)+\left(\frac{5}{12}\times 0.722\right) = 0.8$

$0.92\times\frac{9}{12} = 0.688$

Choose (B)

	Parent
Y_1	6
Y_2	6

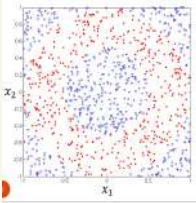
$-\frac{6}{12}\log_2\frac{6}{12}-\frac{6}{12}\log_2\frac{6}{12} = 1$

A: $1-0.8 = 0.2$

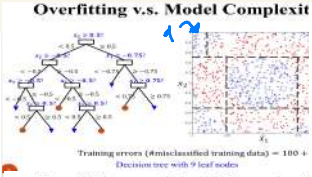
B: $1-0.688 = 0.312$

Lesson 6a: Generalization

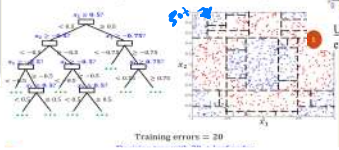
Underfitting and Overfitting (Example)



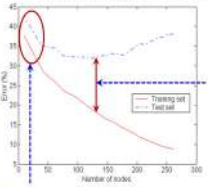
Overfitting v.s. Model Complexit



Overfitting v.s. Model Complexit



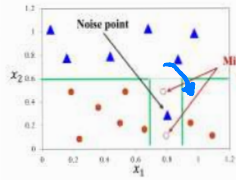
Underfitting and Overfitting



Overfitting: when test error rate begins to increase even though training error rate continues to decrease

Underfitting: when model is too simple, both training and test error rates are large

Overfitting due to Noise (cont.)



Decision boundary is distorted by noise point

Estimation of Generalization Errors

- Training errors: error on the training set: $e(T)$
- Generalization errors: error on previously unseen testing set: $e'(T)$
- Approaches to estimating generalization errors:
  - Optimistic Estimate: $e'(T) = e(T)$
  - Incorporating model complexity
    - Occam's Razor
    - Pessimistic Error Estimate
  - Using Validation Set

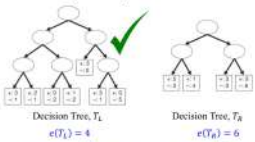
Using a Validation Set

- Divide the original training data set into two smaller subsets
- One is for training, the other (known as the validation set) is for estimating the generalization error
- The complexity of the best model can be estimated based on the performance of the model on the validation set

Optimistic Estimate

- Assume that the training set is a good representation of the overall data
- The training error can be used to provide an optimistic estimate for the generalization error
  - $e'(T) = e(T)$
- A decision tree induction algorithm selects the model that produces the lowest training error rate

An Example of Optimistic Estimate



Occam's Razor

- Definition: Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.
- For complex models, there is a greater chance that it was fitted accidentally by errors in data.
- Therefore, one should include model complexity when evaluating a model.

"Everything should be made as simple as possible, but no simpler." – Albert Einstein

Pessimistic Error Estimate

- Idea: explicitly computes generalization error as the sum of training error and a penalty term for model complexity
  $$e'(T) = e(T) + \Omega(T)$$
- In a decision tree, we can define a penalty term of $k > 0$ on each leaf node, i.e.,
  $$e'(t) = e(t) + \Omega(t) = e(t) + k$$
- Then,
  $$e'(T) = e(T) + N \times k$$
  N: Total number of leaf nodes; $k > 0$, e.g., $k = 0.5$

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors

Overfitting v.s. Model Complexity (cont.)

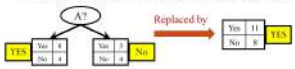
- How do we determine the right model complexity?
- A model with ideal complexity is the one that produces the lowest generalization error
- No knowledge of the test data and how well the model will perform on unseen data
- The best it can do is to **estimate** the generalization error of the induced model

Example

- For a tree with 30 leaf nodes and 10 errors on training (out of 1,000 instances), $k = 0.5$:
  - Training errors = 10
  - Training error rate = $\frac{10}{1000} = 1\%$
  - Generalization errors = $10 + 30 \times 0.5 = 25$
  - Generalization error rate = $\frac{25}{1000} = 2.5\%$

How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the feature values are the same
  - Pre-Pruning (Early Stopping Rule)
    - More restrictive conditions:
      - Stop if number of instances is less than some user-specified threshold
      - Stop if expanding the current node does not improve generalization errors
- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a new leaf node
  - Class label of leaf node is determined from majority class of instances in the sub-tree



criterion : {"gini", "entropy"}, default="gini"
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

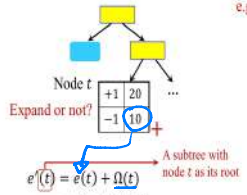
max_depth : int, default=None
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

max_features : int, float or {"auto", "sqrt", "log2"}, default=None
The number of features to consider when looking for the best split:
- If int, then consider max_features features at each split.
- If float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.
- If "auto", then max_features=sqrt(n_features).
- If "sqrt", then max_features=sqrt(n_features).
- If "log2", then max_features=log2(n_features).
- If None, then max_features=n_features.

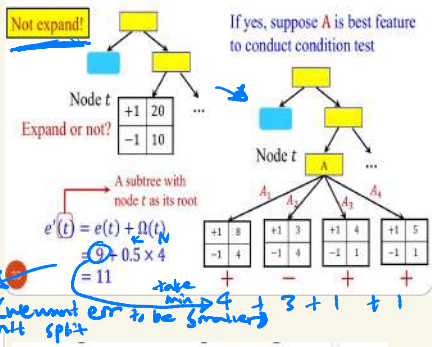
Pre-Pruning Example

Generalization errors: $e'(T) = e(T) + \Omega(T) = e(T) + N \times k$

e.g., $k = 0.5$



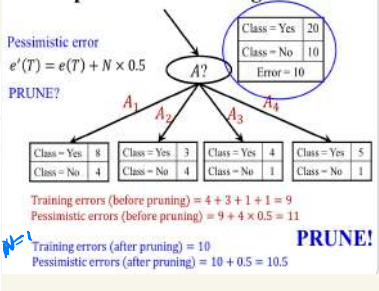
Generalization errors: $e'(T) = e(T) + N \times 0.5$



11 > 10.5 (current err to be smaller) so don't split

take min 4 + 3 + 1 + 1

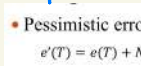
Example of Post-Pruning



- Pessimistic error?
  $e'(T) = e(T) + N \times 0.5$

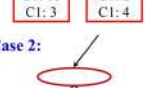
PRUNE?

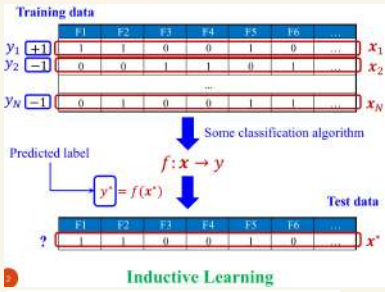
Tutorial

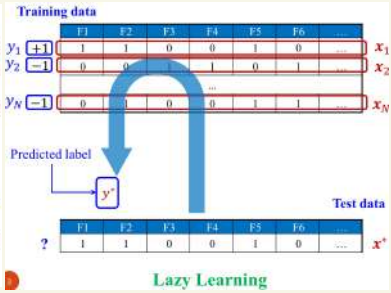


Case 1:

Case 2:







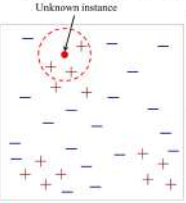
Summary of NN Classifier

- The K-NN classifier is a lazy learner
  - It does not build models explicitly.
  - "Training" is very efficient.
  - Classifying unknown test instances is relatively expensive.

Instance Based Classifiers

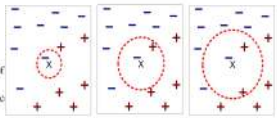
- K-Nearest Neighbors classifier
  - Uses K "closest" points (nearest neighbors) for performing classification

Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored labeled instances
  - Distance metric to compute distance between instances
  - The value of K, the number of nearest neighbors to retrieve
- To classify an unknown instance
  - Compute distance to all the training instances
  - Identify K nearest neighbors
  - Use class labels of nearest neighbors to determine the cla label of the unknown instance (e.g., by taking majority vote)

Definition of Nearest Neighbors



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of an instance x are data points that have the K smallest distance to x

Distance Metric

- Compute distance between two data points in a d-dimensional space:

- Euclidean distance

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j) \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j)}$$

$$= \|(\boldsymbol{x}_i - \boldsymbol{x}_j)\|_2$$

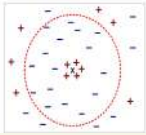
Inner product between x_i and x_j: $x_i \cdot x_j = \sum_{k=1}^{d} x_{ik} x_{jk}$

L_2 norm of x_i: $\|x_i\|_2 = \sqrt{\sum_{k=1}^{d} x_{ik}^2}$

L_2 norm distance

Value of K

- Choosing the value of K:
  - If K is too small, sensitive to noise points
  - If K is too large, neighborhood may include points from other classes



Determine Class Label

- Determine the class from nearest neighbor list
  - Take the majority vote of class labels among the K-nearest neighbors
- Given test data x^*, majority voting:

$$y^* = \arg\max_c \sum_{(x_i, y_i) \in N_{x^*}} I(c = y_i)$$

Indicator function that returns 1 if its input is true, otherwise 0

Nearest neighbors of the test instance x^*

- Every neighbor has the same impact on the classification
- This indeed makes the algorithm sensitive to the choice of K

Revised Voting Scheme

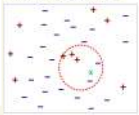
- Solution: distance-weighted voting
  - Weight the influence of each nearest neighbor x_i according to its distance to the test instance x^*:

$$w_i = \frac{1}{d(\boldsymbol{x}^*, \boldsymbol{x}_i)^2}$$

$$y^* = \arg\max_c \sum_{(x_i, y_i) \in N_{x^*}} w_i \times I(c = y_i)$$

Example

Consider a binary classification problem, and a 5-NN classifier



Training record	Class label	Distance to test instance
1	+	3
2	+	1.5
3	+	4
4	-	1.5
5	-	2

- Majority voting:

- Distance-weighted voting:

Tutorial

Other Issues

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

- Scaling issues
  - Feature may need to be scaled to prevent distance from being dominated by some features
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 40kg to 200kg
    - income of a person may vary from $10K to $1M
  - Solution: normalization on features of different scales

Normalization

- Min-max normalization: to $[min_{new}, max_{new}]$
  - Example: To normalize income ranging from $12,000 to $98,000 to [0.0, 1.0], what is the value for $73,600 after normalization?

$$v_{new} = \frac{v_{old} - min_{old}}{max_{old} - min_{old}} (max_{new} - min_{new}) + min_{new}$$

73,600 ➡ $\frac{73600 - 12000}{98000 - 12000} (1.0 - 0) + 0 = 0.716$

- Standardization (z-score normalization) (μ: mean, σ: standard deviation):

$$v_{new} = \frac{v_{old} - \mu_{old}}{\sigma_{old}} \quad \Rightarrow \quad \mu_{new} = 0, \text{and } \sigma_{new} = 1$$

- Example: Let $\mu = 54{,}000$, $\sigma = 16{,}000$. What is the value for $73,600 after standardization?

$$\frac{73600 - 54000}{16000} = 1.225$$

Implementation

```
>>> from sklearn.neighbors import KNeighborsClassifier
...
                                        set number of neighbors
>>> knnC = KNeighborsClassifier(n_neighbors=3)
>>> knnC.fit(X, y)
>>> pred= knnC.predict(X)
```

Build indices s.t. it is more efficient when making predictions on test data

Lesson 7: Artificial Neural Networks

Part 2

I have questions …

- Full-time students: Find me after the tutorials (but not the lectures)
- Part-time students: Dedicated QA time on Thursdays
- Send questions via email boyang.li@ntu.edu.sg or via Microsoft Teams
- Make an appointment

Full-time Arrangement

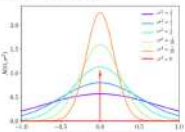
- Lectures
  - 2.30-5.30pm on Thursdays, LT2A
  - Weeks 7-12
- Tutorials
  - 2.30-4.30pm on Mondays, LT2A
  - Weeks 9, 11 and 13
  - Week 12 (3.30-4.30pm, 4 Apr), guest lecture from Geo Tech Analysing FeedBack on Municipal Issues with NLP and Deep Learning.

Purpose of Machine Learning

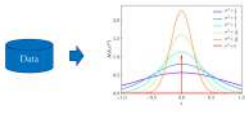
- Make Accurate Predictions
  - Which team will win a soccer match?
  - Which stock will see its price skyrocket?
  - Which patient is at higher risk?
- Usually it is difficult to write down rules manually
- Rather, we learn to make the predictions from paired data $(\boldsymbol{x}_i, y_i)$

- Make Accurate Predictions
  - Artificial Neural Networks (Week 7)
  - Support Vector Machines (Week 8)
  - Regression (Week 8)
  - Ensemble Learning (Week 9)

- Model Uncertainty
  - Germany will probably beat Japan, but what are the odds? 60/40, 70/30, or 80/20?
  - I'm willing to bet more money if the odds are in my favor.
  - Often translates to: what is the shape of the probability distribution?



- Model Uncertainty
  - Density Estimation (Week 11)



- Pattern Discovery
  - We often have little prior experience, knowledge, or insight into the causal mechanisms that generated the data.
  - Still, with only statistical tools, we can identify many important data characteristics
  - Obviously, domain knowledge can enrich and complement statistical tools.

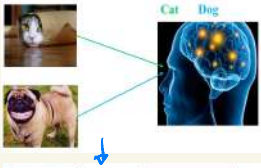
- Pattern Discovery
  - Imagine you are an alien from another planet. You watch a soccer match. What do you observe?
  - Two groups of humans. One ball.
  - Behavior change when the ball goes into the net.

- Pattern Discovery
  - Clustering (Week 10)
  - Dimensionality Reduction (Week 12)

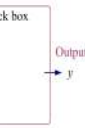
Artificial Neural Networks: Perceptron

Artificial Neural Networks (ANN)

- The study of ANN was inspired by biological neural systems

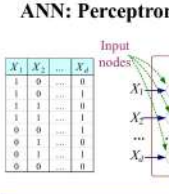


X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	0	-1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output y is 1 if at least two of the three inputs are equal to 1

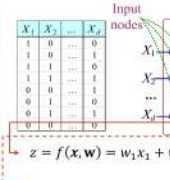
ANN: Perceptron



X_1	X_2	...	X_d	
1	0	...	0	
1	0	...	1	
1	1	...	0	
1	1	...	1	
0	0	...	1	
0	1	...	0	
0	1	...	1	
0	0	...	0	

Each input node is connected via a weighted link to the output node. Weights can be positive, negative or zero (no connection)

Perceptron model



$$z = f(\boldsymbol{x}, \mathbf{w}) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d - \theta$$

$$y = a(z), \text{ where } a(z) = \text{sign}(z) = \begin{cases} 1, & z \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

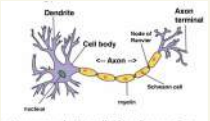
- Model is an assembly of inter-connected nodes and weighted links
- Output node first sums up each of its input value according to the weights of its links
- Compare the weighted sum against some threshold θ
- Produce an output based on the sign of the result

$$z = \sum_{i=1}^{d} w_i x_i - \theta \longrightarrow y = a(z) = \text{sign}(z)$$

$$y = \text{sign}\left(\sum_{i=1}^{d} w_i x_i - \theta\right)$$

"Biology"

- Human brain is a densely interconnected network of neurons, connected to others via dendrites and axons.
- Dendrites and axons transmit electrical signals from one neuron to another
- The human brain learns by changing the strength of the synaptic connection between neurons
- An ANN is composed of an interconnected assembly of nodes and directed links.



- A neuron sends out a spike from the axon after receiving enough input from the dendrites.

- Mathematically, the output of a perceptron model can be expressed in a more compact form

$$y = \text{sign}\left(\sum_{i=1}^{d} w_i x_i - \theta\right)$$

$$y = \text{sign}(\mathbf{w} \cdot \boldsymbol{x}) \quad \text{(Inner product)}$$

where $\boldsymbol{x} = (x_0, x_1, x_2, \ldots, x_d)$

$\mathbf{w} = (w_0, w_1, w_2, \ldots, w_d)$

bias part

$w_0 = -\theta$, and $x_0 = 1$

Inner Product: Review

- Given two vectors $\boldsymbol{x}$ and $\boldsymbol{z}$, which are both of d dimensions, the inner product between $\boldsymbol{x}$ and $\boldsymbol{z}$ is defined as

$$\boldsymbol{x} \cdot \boldsymbol{z} = \sum_{i=1}^{d} x_i z_i$$

$\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ $\quad$ $\boldsymbol{z} = (z_1, z_2, \ldots, z_d)$

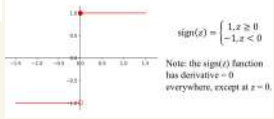
$y = \text{sign}(\mathbf{w} \cdot \boldsymbol{x})$ $\quad$ $\boldsymbol{x} = (x_0, x_1, x_2, \ldots, x_d)$ $\quad$ $\mathbf{w} = (w_0, w_1, w_2, \ldots, w_d)$

$$\mathbf{w} \cdot \boldsymbol{x} = \sum_{i=0}^{d} (w_i x_i) = \sum_{i=1}^{d} (w_i x_i) + (w_0 x_0)$$

$w_0 = -\theta$, and $x_0 = 1$

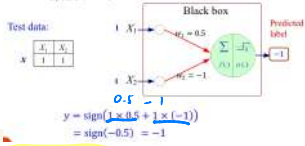
$$y = \text{sign}\left(\sum_{i=1}^{d} w_i x_i - \theta\right) \longleftrightarrow y = \text{sign}(\mathbf{w} \cdot \boldsymbol{x})$$

ANN: Sign Function



Perceptron: Making Prediction

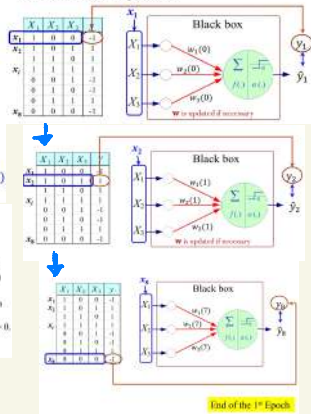
- Given a learned perceptron with $w_1 = 0.5$, $w_2 = -1$, and $\theta = 0$



$$y = \text{sign}(1 \times 0.5 + 1 \times (-1)) = \text{sign}(-0.5) = -1$$

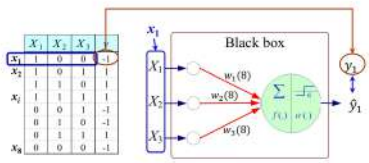
Perceptron: Learning

- During training, the weight parameters $\mathbf{w}$ are adjusted until the outputs of the perceptron become consistent with the true outputs of training data
- The weight parameters $\mathbf{w}$ are updated iteratively or in an online learning manner



The 2nd Epoch starts

Perceptron: Learning (cont.)



Perceptron: Learning (cont.)

- Algorithm: (d dimensions)
1. Let $D = \{(\boldsymbol{x}_i, y_i) \mid i = 1,2,\dots,N\}$ be the set of training examples, $t = 0$
2. Initialize $\mathbf{w}$ with random values $\mathbf{w}_0$
3. **Repeat**
4. **for** each training example $(\boldsymbol{x}_i, y_i)$ **do**
5. Compute the predicted output $\hat{y}_i$
6. Update $\mathbf{w}_t$ by $\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$
7. $t = t + 1$
8. **end for**
9. **Until** stopping condition is met

- Why use the following weight update rule?

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$$

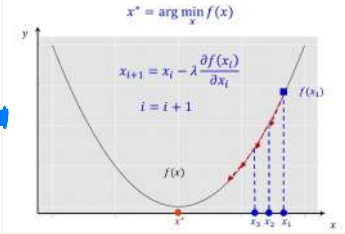
- Induced based on a gradient descent method

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$$

Function to be minimized, e.g., the error loss function

Learning rate $\lambda \in (0,1]$ (not 0)

Gradient Descent



- Weight update rule

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i \quad \Leftarrow \quad \mathbf{w}_{t+1} = \mathbf{w}_t - \lambda\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}}$$

- Consider the loss function $\mathcal{L}$ for each training example as $e_i \triangleq y_i - \hat{y}_i$ $\quad \hat{y}_i = \text{sign}(\mathbf{w}_t \cdot \boldsymbol{x}_i)$

$$\mathcal{L} = \frac{1}{2}e_i^2 = \frac{1}{2}(y_i - \hat{y}_i)^2 = \frac{1}{2}\left(y_i - \text{sign}(\mathbf{w}_t \cdot \boldsymbol{x}_i)\right)^2$$

- Update the weight using a gradient descent method

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w}_t - \lambda\frac{\partial \mathcal{L}(\hat{y})}{\partial \hat{y}}\frac{\partial \hat{y}(z)}{\partial z}\frac{\partial z(\mathbf{w})}{\partial \mathbf{w}}$$

Chain rule

$\mathcal{L} = \frac{1}{2}(y - \hat{y})^2 \quad \hat{y} = \text{sign}(x) \quad z = \mathbf{w} \cdot \boldsymbol{x}$

Chain Rule of Calculus (Review)

- Suppose that $y = g(x)$ and $z = f(y) = f(g(x))$
- Chain rule of calculus:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

- Generalized to the vector case: suppose $\boldsymbol{x} \in R^m$, $\boldsymbol{y} \in R^n$, $\boldsymbol{y} = g(\boldsymbol{x})$ and $z = f(\boldsymbol{y})$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

Perceptron: Learning (cont.)

- The weight update formula for perceptron:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \lambda\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w}_t - \lambda\frac{\partial \mathcal{L}(\hat{y})}{\partial \hat{y}}\frac{\partial \hat{y}(z)}{\partial z}\frac{\partial z(\mathbf{w})}{\partial \mathbf{w}}$$

$$\mathcal{L} = \frac{1}{2}(y_i - \hat{y}_i)^2$$

$\hat{y}_i = \text{sign}(z_i)$ ~~$\hat{y}_i = z_i$~~

$(y_i - \hat{y}_i) \times -1$

$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$

Note: the sign() function has derivative = 0 except at $z = 0$. Therefore, to compute meaningful derivative, we remove sign() from consideration

$\frac{1}{z_i} = \mathbf{w}_t \cdot \boldsymbol{x}_i$

$\boldsymbol{x}_i$

Approximating the derivative?

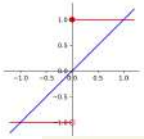
- The equation used to compute $\hat{y}_i$ from z_i

$$\hat{y}_i = \text{sign}(z_i)$$

- The equation used to compute $\frac{\partial \hat{y}(z)}{\partial z}$

$$\hat{y}_i = z_i$$

- Why? This is approximating the step function with a linear function.



While the derivative itself is incorrect, its direction is correct.
That is, if you want to increase $\hat{y}_i$, you should increase z_i, and vice versa.

Perceptron Weights Update

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$$

- If the prediction is correct, $(y - \hat{y}) = 0$, then weight remains unchanged $\mathbf{w}_{t+1} = \mathbf{w}_t$
- If $y = +1$ and $\hat{y} = -1$, then $(y - \hat{y}) = 2$
- The weights of all links with positive inputs need to be updated by increasing their values
- The weights of all links with negative inputs need to be updated by decreasing their weights

$\boldsymbol{x}_i$	x_{i1}	...	x_{ik}	...	x_{id}
	>0		$=0$		<0
$\mathbf{w}_{t+1}$	$w_1 \uparrow$	...	w_k	...	$w_d \downarrow$

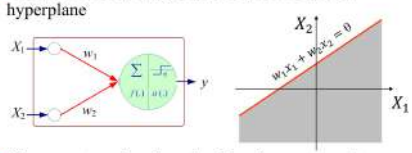
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda(y_i - \hat{y}_i)\boldsymbol{x}_i$$

- If $y = -1$ and $\hat{y} = +1$, then $(y - \hat{y}) = -2$
- The weights of all links with positive inputs need to be updated by decreasing their values
- The weights of all links with negative inputs need to be updated by increasing their weights

$\boldsymbol{x}_i$	x_{i1}	...	x_{ik}	...	x_{id}
	>0		$=0$		<0
$\mathbf{w}_{t+1}$	$w_1 \downarrow$	...	w_k	...	$w_d \uparrow$

Convergence

- The decision boundary of a perceptron is a linear hyperplane



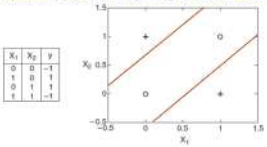
- The perceptron learning algorithm is guaranteed to converge to an optimal solution for linear classification problems

Perceptron Limitation

- If the problem is not linearly separable, the algorithm fails to converge

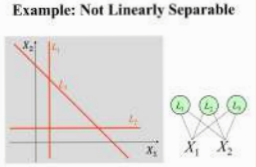
Nonlinearly separable data given by the XOR function

X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1

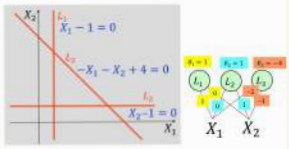


Multi-layer Perceptron

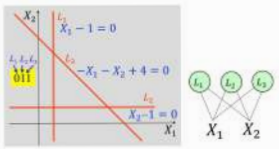
Example: Not Linearly Separable



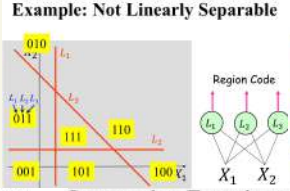
Example: Not Linearly Separable



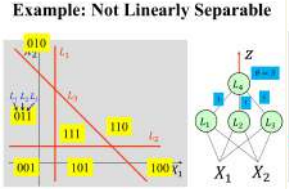
Example: Not Linearly Separable



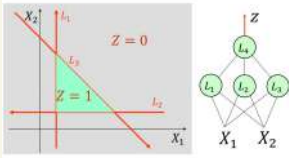
Example: Not Linearly Separable



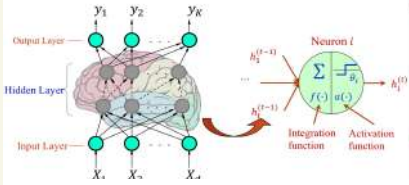
Example: Not Linearly Separable



Example: Not Linearly Separable ...



General Structure: Multilayer ANN



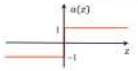
Integration Functions

- Weighted sum: $\sum_{i=1}^{d} w_i X_i - \theta$
- Quadratic function $\sum_{i=1}^{d} w_i X_i^2 - \theta$
- Spherical function $\sum_{i=1}^{d} (X_i - w_i)^2 - \theta$

for this course

Activation Functions

- Sign function (Threshold function) $a(z) = \text{sign}(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$



- Unipolar sigmoid function: $a(z) = \frac{1}{1 + e^{-\lambda z}}$

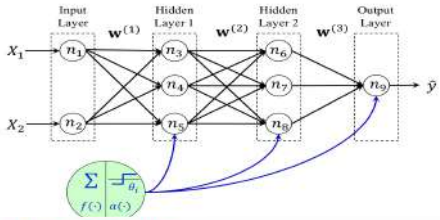
When $\lambda = 1$, it is called sigmoid function

Update Weights for Multi-layer NNs

- Initialize the weights in each layer $(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(k)}, \ldots, \mathbf{w}^{(m)})$
- Adjust the weights such that the output of ANN is consistent with class labels of training examples
  - Loss function for each training instance: $\mathcal{L} = \frac{1}{2}(y_i - \hat{y}_i)^2$
  - For each layer k, update the weights, $\mathbf{w}^{(k)}$, by gradient descent at each iteration t: $\mathbf{w}_{t+1}^{(k)} = \mathbf{w}_t^{(k)} - \lambda \frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(k)}}$
  - Computing the gradient w.r.t. weights in each layer is computationally expensive!

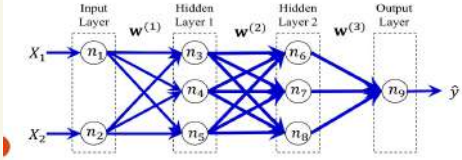
The Backpropagation Algorithm

A Multi-layer Feed-forward NN

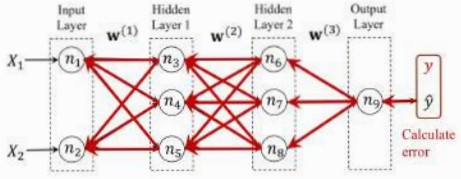


Backpropagation: Basic Idea

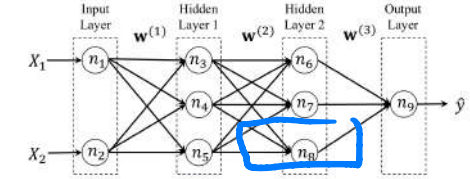
- Initialize the weights $(\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(3)})$
- **Forward pass:** each training examples $(\boldsymbol{x}_i, y_i)$ is used to compute outputs of each hidden layer and generate the final output $\hat{y}_i$ based on the ANN

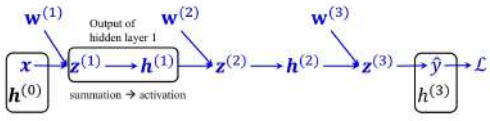


- **Backpropagation:** Starting with the output layer, to propagate error back to the previous layer in order to update the weights between the two layers, until the earliest hidden layer is reached



The Computational Graph





Backpropagation (BP)

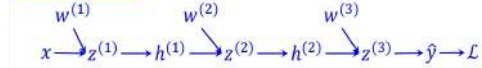
- Gradient of $\mathcal{L}$ w.r.t. $w^{(3)}$: $\frac{\partial \mathcal{L}}{\partial w^{(3)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial w^{(3)}}$
- Gradient of $\mathcal{L}$ w.r.t. $w^{(2)}$:

$$\frac{\partial \mathcal{L}}{\partial w^{(2)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial w^{(2)}}$$

- Gradient of $\mathcal{L}$ w.r.t. $w^{(1)}$:

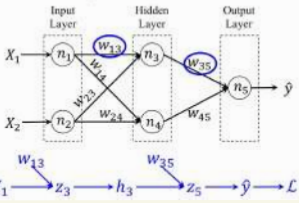
$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial w^{(1)}}$$

Consider each layer contains a single unit



An Example

- Consider an ANN of 1 hidden layer as follows. Suppose the sign function and the weighted sum function are used for both hidden and output nodes



$w'_{35} = w_{35} + \lambda e_i h_3$

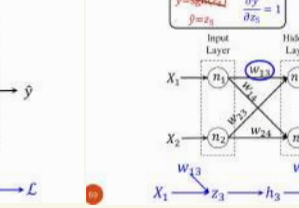
$w'_{35} = w_{35} - \lambda \frac{\partial \mathcal{L}}{\partial w_{35}} = w_{35} - \lambda \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_5} \frac{\partial z_5}{\partial w_{35}}$

$\mathcal{L} = \frac{1}{2} e_i^2 = \frac{1}{2} (y_i - \hat{y}_i)^2$

$-1 \times (y_i - \hat{y}_i) = -e_i$

$\hat{y} = \text{sign}(z_5)$ $\quad \frac{\partial \hat{y}}{\partial z_5} = 1$ $\quad \hat{y} = z_5$

$z_5 = w_{35} h_3 + w_{45} h_4$



$w'_{13} = w_{13} + \lambda e_i w_{35} X_1$

$w'_{13} = w_{13} - \lambda \frac{\partial \mathcal{L}}{\partial w_{13}} = w_{13} - \lambda \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_5} \frac{\partial z_5}{\partial h_3} \frac{\partial h_3}{\partial z_3} \frac{\partial z_3}{\partial w_{13}}$

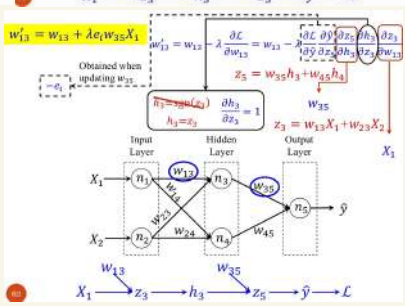
Obtained when updating w_{35}

$-e_i$

$z_5 = w_{35} h_3 + w_{45} h_4$

$h_3 = \text{sign}(z_3)$ $\quad \frac{\partial h_3}{\partial z_3} = 1$ $\quad h_3 = z_3$

$z_3 = w_{13} X_1 + w_{23} X_2$



$w'_{23} = w_{23} + \lambda e_i w_{35} X_2$

$w'_{23} = w_{23} - \lambda \frac{\partial \mathcal{L}}{\partial w_{23}} = w_{23} - \lambda \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_5} \frac{\partial z_5}{\partial h_3} \frac{\partial h_3}{\partial z_3} \frac{\partial z_3}{\partial w_{23}}$

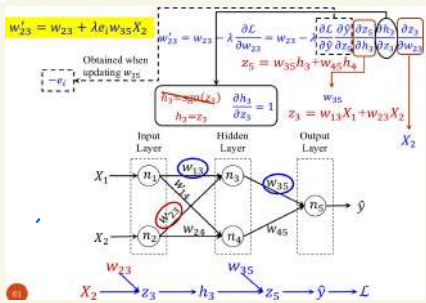
Obtained when updating w_{35}

$-e_i$

$z_5 = w_{35} h_3 + w_{45} h_4$

$h_3 = \text{sign}(z_3)$ $\quad \frac{\partial h_3}{\partial z_3} = 1$ $\quad h_3 = z_3$

$z_3 = w_{13} X_1 + w_{23} X_2$

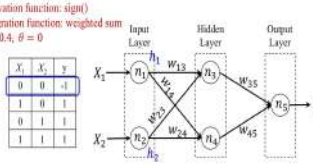


BP Algorithm: Example

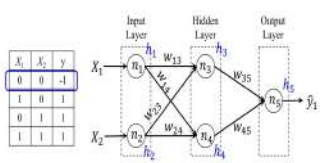
Activation function: sign()

Integration function: weighted sum

$\lambda = 0.4$; $\theta = 0$



X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1



X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1

- **Initialization:**

$(w_{13} = 1, w_{14} = 1, w_{23} = 1, w_{24} = 1, w_{35} = 1, w_{45} = 1)$

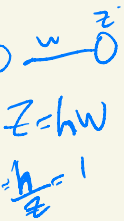
For the 1st example: $h_1 = 0$ and $h_2 = 0$

Forward pass:

$h_3 = \text{sign}(0 \times 1 + 0 \times 1) = 1$ and $h_4 = \text{sign}(0 \times 1 + 0 \times 1) = 1$

Then $\hat{y}_1 = h_5 = \text{sign}(1 \times 1 + 1 \times 1) = 1$

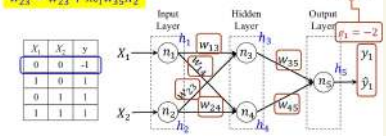
Back-propagation is for computing gradient
gradient descent is for optimization



$w'_{13} = w_{13} + \lambda e_i w_{35} X_1$ $\quad$ $w'_{35} = w_{35} + \lambda e_i h_3$

$w'_{23} = w_{23} + \lambda e_i w_{35} X_2$

BP Algorithm: Example (cont.)



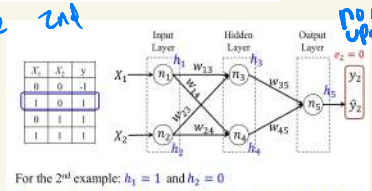
X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1

Backpropagation:

$w_{35} = 1 + 0.4 \times (-2) \times 1 = 0.2$ $\quad w_{45} = 1 + 0.4 \times (-2) \times 1 = 0.2$

$w_{13} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$ $\quad w_{14} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$

$w_{23} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$ $\quad w_{24} = 1 + 0.4 \times (-2) \times 1 \times 0 = 1$

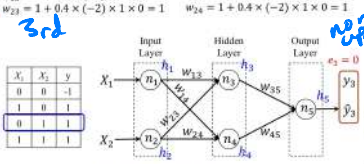


X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1

For the 2nd example: $h_1 = 1$ and $h_2 = 0$

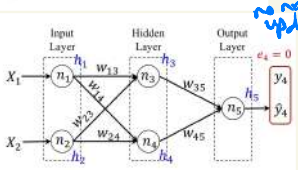
$h_3 = \text{sign}(1 \times 1 + 0 \times 1) = 1$ and $h_4 = \text{sign}(1 \times 1 + 0 \times 1) = 1$

Then $\hat{y}_2 = h_5 = \text{sign}(1 \times 0.2 + 1 \times 0.2) = 1$



X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1

For the 3rd example: $h_1 = 0$ and $h_2 = 1$



X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1

For the 4th example: $h_1 = 1$ and $h_2 = 1$

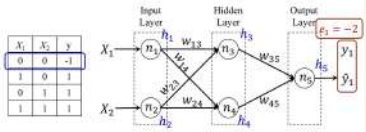
$h_3 = \text{sign}(1 \times 1 + 1 \times 1) = 1$ and $h_4 = \text{sign}(1 \times 1 + 1 \times 1) = 1$

Then $\hat{y}_4 = h_5 = \text{sign}(1 \times 0.2 + 1 \times 0.2) = 1$

End of the 1st Epoch

The 2nd Epoch starts

BP Algorithm: Example (cont.)



X_1	X_2	y
0	0	-1
1	0	1
0	1	1
1	1	1

For the 1st example again: $h_1 = 0$ and $h_2 = 0$

$h_3 = \text{sign}(0 \times 1 + 0 \times 1) = 1$ and $h_4 = \text{sign}(0 \times 1 + 0 \times 1) = 1$

Then $\hat{y}_1 = h_5 = \text{sign}(0.2 \times 1 + 0.2 \times 1) = 1$

Weights need to be further updated via backpropagation

Design Issues for ANN

- The number of nodes in the input layer
  - Assign an input node to each numerical or binary input variable
- The number of nodes in the output layer
  - Binary class problem → single node
  - C-class problem → C output nodes
- How many nodes in the hidden layer(s)?
  - Too many parameters result in networks that are too complex and overfit the data
- If the network underfits
  - Try to increase the number of hidden units
- If the network overfits
  - Try to decrease the number of hidden units

- How many nodes in the hidden layer(s)?
  - Too many parameters result in networks that are too complex and overfit the data
- If the network underfits
  - Try to increase the number of hidden units
- If the network overfits
  - Try to decrease the number of hidden units

Lesson 8a: Support Vector Machines

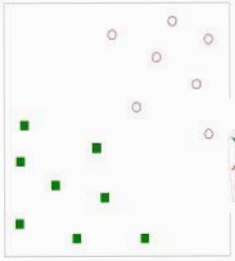
light weight, robust
Optimization easy
→ deep learning if huge
small dataset can use SVM

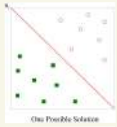
Support Vector Machines (SVMs)

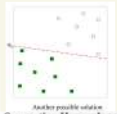
- SVMs have shown promising empirical results in many practical applications, such as computer vision, sensor networks and text mining
- The motivation behind SVMs is from the geometry perspective of linear algebra
- The objective of SVMs is to learn a maximum margin hyperplane
  - Based on statistical learning theory

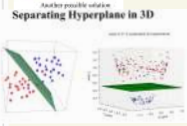
Separating Hyperplane

- To learn a binary classifier
- To find a hyperplane (linear decision boundary) so that all the squares reside on one side of the hyperplane and all the circles reside on the other



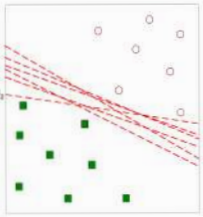


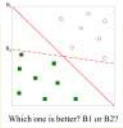




Maximum Margin

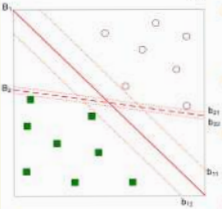
- Although all the hyperplanes shown in the figure can separate training examples perfectly, their generalization errors may differ.
- How to choose one of these hyperplanes to construct a classifier's decision boundary with small generalization errors?





Maximum Margin (cont.)

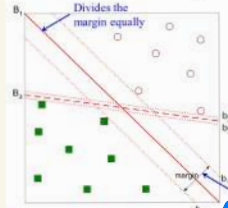
- Each decision boundary B_i is associated with a pair of parallel hyperplanes: b_{i1} and b_{i2}
- b_{i1} is obtained by moving the hyperplane until it touches the closest circle(s)
- b_{i2} is obtained by moving a hyperplane away from the decision boundary until it touches the closest square(s)
- The distances from b_{i1} and b_{i2} to B_i are the same



Maximum Margin (cont.)

- Assumption: larger margins imply better generalization errors
- The margin of B_1 is much larger than that of B_2. Therefore, B_1 is better than B_2

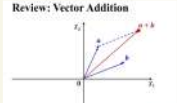
The distance between these two hyperplanes is known as the margin of the classifier



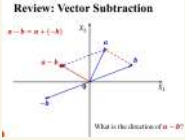
B1 better than B2

larger margin = better
wider = lesser error
Can tolerance disturbance

Review: Vector Addition



Review: Vector Subtraction



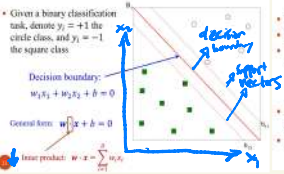
Decision Boundary

- Given a binary classification task, denote $y_i = +1$ the circle class, and $y_i = -1$ the square class

Decision boundary: $w_1x_1 + w_2x_2 + b = 0$

General form: $\mathbf{w}^T\mathbf{x} + b = 0$

Inner product: $\mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{n} w_i x_i$



Review: Inner Products of Vectors

- We use bold letters to denote vectors, such as $\boldsymbol{a}$ and $\boldsymbol{b}$.
- A vector can have many dimensions: $\boldsymbol{a} = (a_1, a_2, .., a_n)$
- The inner product of two D-dimensional vectors (D-vectors), $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as

$$\boldsymbol{a} \cdot \boldsymbol{b} = a_1 b_1 + a_2 b_2 + \cdots + a_D b_D = \sum_i a_i b_i$$

- Also, $\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\| \|\boldsymbol{b}\| \cos\theta$, where θ is the angle between $\boldsymbol{a}$ and $\boldsymbol{b}$
- $\|\boldsymbol{a}\|$ is the Euclidean norm of $\boldsymbol{a}$

$$\|\boldsymbol{a}\| = \sqrt{a_1 a_1 + a_2 a_2 + \cdots + a_D a_D} = \sqrt{\boldsymbol{a} \cdot \boldsymbol{a}}$$

Review: Geometry of Inner Products

$\boldsymbol{a} \cdot \boldsymbol{b} = \|\boldsymbol{a}\| \|\boldsymbol{b}\| \cos\theta$, where θ is the angle between $\boldsymbol{a}$ and $\boldsymbol{b}$

$\frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{b}\|} = \|\boldsymbol{a}\| \cos\theta$ is the length of the projection of $\boldsymbol{a}$ on (or onto) $\boldsymbol{b}$

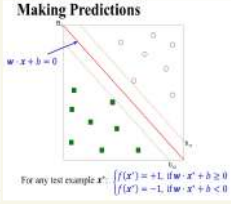
What if $\boldsymbol{a}$ and $\boldsymbol{b}$ form an obtuse angle?

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are orthogonal, $\theta = 90°, \cos\theta = 0, \boldsymbol{a} \cdot \boldsymbol{b} = 0$

Making Predictions

$\mathbf{w} \cdot \mathbf{x} + b = 0$

For any test example $\mathbf{x}'$: $\begin{cases} f(\mathbf{x}') = +1, & \text{if } \mathbf{w} \cdot \mathbf{x}' + b \geq 0 \\ f(\mathbf{x}') = -1, & \text{if } \mathbf{w} \cdot \mathbf{x}' + b < 0 \end{cases}$

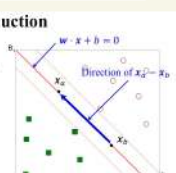


Margin – Induction

- Suppose $\mathbf{x}_a$ and $\mathbf{x}_b$ are two points located on the decision boundary

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_a + b = 0 \\ \mathbf{w} \cdot \mathbf{x}_b + b = 0 \end{cases}$$

$$\mathbf{w} \cdot (\mathbf{x}_a - \mathbf{x}_b) = 0$$



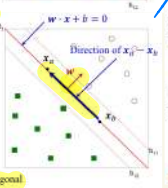
- Suppose $\mathbf{x}_a$ and $\mathbf{x}_b$ are two points located on the decision boundary,

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_a + b = 0 \\ \mathbf{w} \cdot \mathbf{x}_b + b = 0 \end{cases}$$

$$\mathbf{w} \cdot (\mathbf{x}_a - \mathbf{x}_b) = 0$$

Based on definition of inner product

The direction of $\mathbf{w}$ is orthogonal to the decision boundary



- For any circle $\mathbf{x}_c$ located above the decision boundary:

$$\mathbf{w} \cdot \mathbf{x}_c + b \geq k, \text{ where } k > 0$$

- For any square $\mathbf{x}_s$ located below the decision boundary:

$$\mathbf{w} \cdot \mathbf{x}_s + b \leq k', \text{ where } k' < 0$$

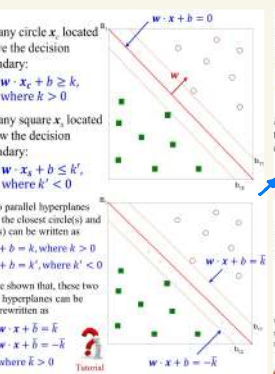
The two parallel hyperplanes passing the closest circle(s) and square(s) can be written as

$$\mathbf{w} \cdot \mathbf{x} + b = k, \text{ where } k > 0$$
$$\mathbf{w} \cdot \mathbf{x} + b = k', \text{ where } k' < 0$$

It can be shown that, these two parallel hyperplanes can be further rewritten as

$$\mathbf{w} \cdot \mathbf{x} + \bar{b} = \bar{k}$$
$$\mathbf{w} \cdot \mathbf{x} + \bar{b} = -\bar{k}$$
$$\text{where } \bar{k} > 0$$

Tutorial



The two parallel hyperplanes passing the closest circle(s) and square(s) can be written as

$$\mathbf{w} \cdot \mathbf{x} + \bar{b} = \bar{k} \quad \text{where } \bar{k} > 0$$
$$\mathbf{w} \cdot \mathbf{x} + \bar{b} = -\bar{k}$$

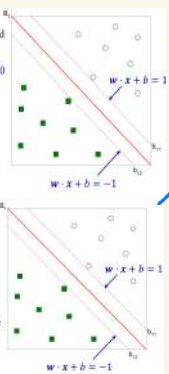
After rescaling $\mathbf{w}$ and $\bar{b}$, the two parallel hyperplanes can be further rewritten as

$$\mathbf{w} = \frac{\mathbf{w}}{\bar{k}}, \quad b = \frac{\bar{b}}{\bar{k}}$$
$$\mathbf{w} \cdot \mathbf{x} + b = 1$$
$$\mathbf{w} \cdot \mathbf{x} + b = -1$$

After rescaling $\mathbf{w}$ and $\bar{b}$, the two parallel hyperplanes can be further rewritten as

$$\mathbf{w} \cdot \mathbf{x} + b = 1$$
$$\mathbf{w} \cdot \mathbf{x} + b = -1$$

We should use different letters for the rescaled $\mathbf{w}$ and b, but we abuse notations for simplicity.



b_{11}: $\mathbf{w} \cdot \mathbf{x}_1 + b = 1$

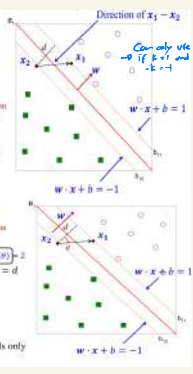
b_{12}: $\mathbf{w} \cdot \mathbf{x}_2 + b = -1$

$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2$$

Based on definition of inner product

$$\|\mathbf{w}\|_2 d = 2$$

Margin



$$\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2) = 2$$

Based on definition of inner product

$$\|\mathbf{w}\|_2 \times \|\mathbf{x}_1 - \mathbf{x}_2\|_2 \times \cos(\theta) = 2$$

Based on definition of $\cos(\cdot)$: $= d$

$$\|\mathbf{w}\|_2 \times d = 2$$

Important: this holds only for the rescaled $\mathbf{w}$.

Margin Maximization

$\|\boldsymbol{w}\|_2 \times d = 2 \longrightarrow d = \frac{2}{\|\boldsymbol{w}\|_2}$ $\quad \|\boldsymbol{w}\|_2^2 = \sum_{i=1}^{d} (w_i \times w_i)$

Maximize margin $d = \frac{2}{\|\boldsymbol{w}\|_2} \longrightarrow$ Minimize $\frac{\|\boldsymbol{w}\|_2^2}{2}$

For convenience in computation

Constraints: $\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1$, if $y_i = 1$

$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1$, if $y_i = -1$

OR

$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$

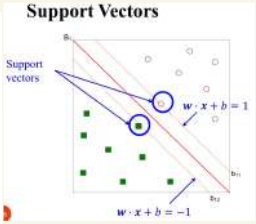
Optimization Problem for SVMs

- Optimization problem of linear SVMs

$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}$$
$$\text{s.t.} \quad y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, \; i = 1, \ldots, N$$

- The optimization is convex
  - Many numerical approaches can be applied to find the solution
  - Convex is easy. Non-convex is hard.
  - The exact optimization algorithms are beyond the scope of this course.

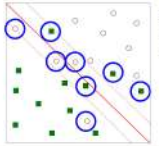
Support Vectors



Non-separable Case

Additional Notes (Optional)

- What if data of two classes cannot be perfectly separated?



Slack variables need to be introduced to absorb errors

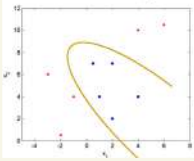
Implementation Example

```
>>> from sklearn import svm
...
>>> svmC = svm.LinearSVC()
>>> svmC.fit(X, y)
>>> pred= svmC.predict(X)
```

Nonlinear SVMs

Additional Notes (Optional)

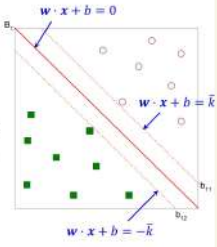
- What if decision boundary is not linear?



Kernel trick in the dual form

Question 2

- $\boldsymbol{w}$ determines the orientation (slope) of the decision boundary.
- The support vectors determine how the decision boundary moves in parallel motion.
- Together, they determine b.

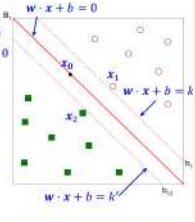


Question 2 (cont.)

b_{11}: $\boldsymbol{w} \cdot \boldsymbol{x}_1 + b = k$, where $k > 0$

b_{12}: $\boldsymbol{w} \cdot \boldsymbol{x}_2 + b = k'$, where $k' < 0$

Given two support vectors (or two points on b_{11} and b_{12} repsectively), I can choose b such that $k = -k'$

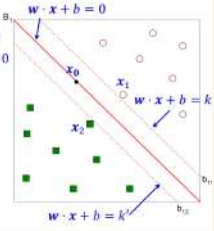


Question 2 (cont.)

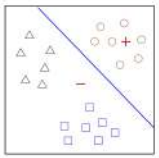
b_{11}: $\boldsymbol{w} \cdot \boldsymbol{x}_1 + b = k$, where $k > 0$

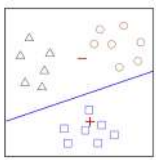
b_{12}: $\boldsymbol{w} \cdot \boldsymbol{x}_2 + b = k'$, where $k' < 0$

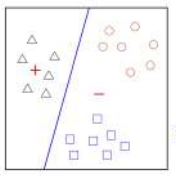
$$\boldsymbol{w} \cdot \boldsymbol{x}_1 + b = -(\boldsymbol{w} \cdot \boldsymbol{x}_2 + b)$$
$$2b = -\boldsymbol{w} \cdot \boldsymbol{x}_1 - \boldsymbol{w} \cdot \boldsymbol{x}_2$$
$$b = -\frac{1}{2}\boldsymbol{w} \cdot (\boldsymbol{x}_1 + \boldsymbol{x}_2)$$



Multi-Class Classification







- Give a 3-class classification problem: C
- General approaches: 1 v.s. rest
  - Binary classification 1: positive (C_1) v.s. ne
  - Binary classification 2: positive (C_2) v.s. ne
  - Binary classification 3: positive (C_3) v.s. ne
  - For a test instance $\boldsymbol{x}^*$, apply binary classifie to make predictions on $\boldsymbol{x}^*$
  - Combine predicted results of $f_1(\boldsymbol{x}^*)$, $f_2(\boldsymbol{x}^*)$ make a final prediction

Linear SVMs for Multi-Class

- f_i only generates $-1/1$:
  1: belong to C_i, and -1: not belong to C_i
- Given a test data $\boldsymbol{x}^*$, suppose

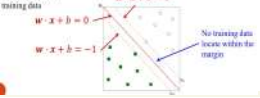
	C_1	C_2	C_3
$f_1(\boldsymbol{x}^*) = -1$	0	1	1
$f_2(\boldsymbol{x}^*) = 1$	0	1	0
$f_3(\boldsymbol{x}^*) = -1$	1	1	0
Total Votes:	1	3	1

Linear SVMs: Separable Case

- Optimization problem of linear SVM

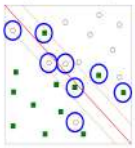
$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2} \quad \text{s.t. } y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1,\ i = 1, \ldots, N$$

Maximize margin



Linear SVMs: Nonseparable Case

- What if the problem is not separable?



Slack variables $\xi_i \geq 0$ need to be introduced to absorb errors

Slack Variables

- For Separable Case:

$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1$, if $y_i = 1$
$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1$, if $y_i = -1$

OR $y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$

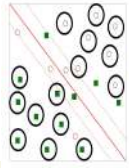
- For Nonseparable Case:

$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1 - \xi_i$, if $y_i = 1$
$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1 + \xi_i$, if $y_i = -1$

$\xi_i \geq 0$ Slack variables

OR

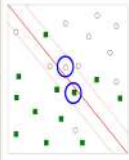
$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$



If $\xi_i = 0$, there is no problem with $\boldsymbol{x}_i$

$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$

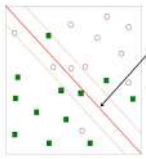
$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$



If $0 < \xi_i < 1$, $\boldsymbol{x}_i$ is correctly classified but in the margin

$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$

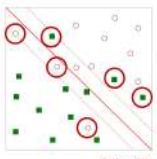
$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq k$
$0 < k < 1$



If $\xi_i = 1$, $\boldsymbol{x}_i$ is on the decision boundary (random guess)

$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$

$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 0$



If $\xi_i > 1$, $\boldsymbol{x}_i$ is misclassified

$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i$

$y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq -k$
$k > 0$

Can be negative

$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq -k$, if $y_i = 1$
$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq k$, if $y_i = -1$

Can be positive

Soft Error

- The number of misclassifications is $\#\{\xi_i > 1\}$
- The number of nonseparable points is $\#\{\xi_i > 0\}$
- Soft errors:

$$\sum_i \xi_i$$

Linear SVMs: Nonseparable Case

- Linear SVMs with soft errors:

$$\min_{\boldsymbol{w},b,\xi_i} \frac{\|\boldsymbol{w}\|_2^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right) \quad \text{s.t. } y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 - \xi_i,\ i = 1, \ldots, N,\ \xi_i \geq 0$$

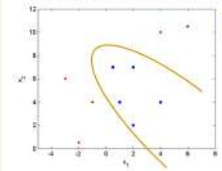
Penalize the decision boundary with large values of slack variables

$C \geq 0$ is a parameter to tradeoff the impact of margin maximization and tolerable errors

Nonnegative ξ_i provides an estimate of the error of the decision boundary on the training example $\boldsymbol{x}_i$

Nonlinear SVMs

- What if decision boundary is not linear?



Kernel trick in the dual form

- How to generalize linear decision boundary to become nonlinear?
- Key idea: transform $\boldsymbol{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point $\boldsymbol{x}_i$ are located
  - Feature space: the space of $\varphi(\boldsymbol{x}_i)$ after transformation
- Assumption: in a higher dimensional space, it is easier to find a linear hyperplane to classify data

- Optimization problem of nonlinear SVMs

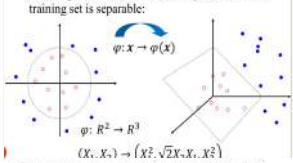
$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2} \quad \text{s.t. } y_i \times (\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) \geq 1,\ i = 1, \ldots, N$$

$\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b = 0$ Hyperplane in feature space

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically very high dimensional!
- The kernel trick comes to rescue

Feature Mapping

- The original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Nonlinear SVM: Kernel Trick

- Suppose $\varphi(\cdot)$ is given as follows, mapping an instance from 2-dimensional space to 6-dimensional space:

$$\varphi([X_1, X_2]) = \left[1, \sqrt{2}X_1, \sqrt{2}X_2, X_1^2, X_2^2, \sqrt{2}X_1X_2\right]$$

- Given two data instances: $\boldsymbol{a} = [A_1, A_2]$ and $\boldsymbol{b} = [B_1, B_2]$

$$\varphi(\boldsymbol{a}) = \left[1, \sqrt{2}A_1, \sqrt{2}A_2, A_1^2, A_2^2, \sqrt{2}A_1A_2\right]$$
$$\varphi(\boldsymbol{b}) = \left[1, \sqrt{2}B_1, \sqrt{2}B_2, B_1^2, B_2^2, \sqrt{2}B_1B_2\right]$$

- Inner product of the two instances after feature mapping:

$$\varphi(\boldsymbol{a}) \cdot \varphi(\boldsymbol{b}) = 1 + 2A_1B_1 + 2A_2B_2 + A_1^2B_1^2 + A_2^2B_2^2 + 2A_1A_2B_1B_2$$
$$= (1 + A_1B_1 + A_2B_2)^2$$

- Inner product of the two instances after feature mapping:

$$\varphi(\boldsymbol{a}) \cdot \varphi(\boldsymbol{b}) = 1 + 2A_1B_1 + 2A_2B_2 + A_1^2B_1^2 + A_2^2B_2^2 + 2A_1A_2B_1B_2$$
$$= (1 + A_1B_1 + A_2B_2)^2$$

- So, if we define the kernel function as follows, there is no need to carry out $\varphi(\cdot)$ explicitly

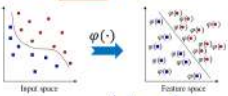
$$k(\boldsymbol{a}, \boldsymbol{b}) = (1 + A_1B_1 + A_2B_2)^2 = (1 + \boldsymbol{a} \cdot \boldsymbol{b})^2$$

- This use of kernel function to avoid carrying out $\varphi(\cdot)$ explicitly is known as the kernel trick

Kernel Trick: General Idea

- If $\varphi(\cdot)$ satisfies some conditions, then we can find a function $k(\cdot, \cdot)$ such that

Kernel function → $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$



$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2}\right)$$

How to Apply Kernel Trick?

- Optimization problem for nonlinear SVMs (separable)

$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2} \quad \text{s.t. } y_i \times (\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) \geq 1,\ i = 1, \ldots, N$$

Primal Form

Instances in the feature space do not appear in the form of inner products

- The kernel trick is not applicable
- How about its dual form?

Lagrange multiplier method

Lagrange Multiplier Method: Idea

- Given: an objective $f(\boldsymbol{w})$ to be minimized, with a set of inequality constraints to be satisfied $h_i(\boldsymbol{w}) \leq 0, i = 1, 2, \ldots, q$

$$\min_{\boldsymbol{w}} f(\boldsymbol{w}) \quad \text{s.t. } h_i(\boldsymbol{w}) \leq 0,\ i = 1, \ldots, q$$

- The Lagrangian for the optimization problem:

$$L(\boldsymbol{w}, \boldsymbol{\lambda}) = f(\boldsymbol{w}) + \sum_{i=1}^{q} \lambda_i h_i(\boldsymbol{w})$$

$\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_q)$ The Lagrange multipliers

The Dual Form (Separable)

- By using Lagrangian Multiplier method

$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2} \quad \text{s.t. } y_i \times (\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) \geq 1,\ i = 1, \ldots, N$$

Primal Form

$$\max_{\boldsymbol{\lambda}} L_D(\boldsymbol{\lambda}) = -\left(\frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j (\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)) - \sum_{i=1}^{N} \lambda_i\right)$$

Dual Form

Dual Optimization Problem

- The dual Lagrangian involves only the Lagrange multipliers and the training data
- The negative sign in the dual Lagrangian transforms a minimization problem of the primal form to a maximization problem of the dual form
- The objective is to maximize $L_D(\boldsymbol{\lambda})$
  - Can be solved using numerical techniques such as quadratic programming

- Once the λ_i's are found, we can obtain the feasible solutions for $\boldsymbol{w}$ and b from

$$\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \varphi(\boldsymbol{x}_i) \quad \text{AND} \quad \lambda_i(y_i(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) - 1) = 0$$

- The decision boundary can be expressed as

$$\boldsymbol{w} \cdot \varphi(\boldsymbol{x}) + b = \left(\sum_{i=1}^{N} \lambda_i y_i \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x})\right) + b = 0$$

If $\boldsymbol{x}_i$ is a support vector, then the corresponding $\lambda_i > 0$, otherwise, $\lambda_i = 0$

- For a test instance $\boldsymbol{x}^*$, it can be classified using

$$f(\boldsymbol{x}^*) = \text{sign}\left(\sum_{i=1}^{N} \lambda_i y_i \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}^*) + b\right)$$

Nonlinear SVM via Kernel Trick

Training: $\max_{\boldsymbol{\lambda}} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j (\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j))\right)$

Decision boundary: $\sum_{i=1}^{N} \lambda_i y_i (\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}^*)) + b = 0$

- The data points only appear as inner product
- As long as the inner product in the feature space can be calculated, no need for the explicit mapping

- Replace inner product in feature space by kernel function

Training: $\max_{\boldsymbol{\lambda}} \left(\sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)$

Decision boundary: $\sum_{i=1}^{N} \lambda_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}^*) + b = 0$

If $\boldsymbol{x}_i$ is a support vector, then the corresponding $\lambda_i > 0$, otherwise, $\lambda_i = 0$

$k(\boldsymbol{x}_i, \boldsymbol{x}^*) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}^*)$

$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$

Kernel Functions: Examples

- Linear kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$$

- Radial basis function kernel with width σ

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2}\right)$$

- Polynomial kernel with degree d

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j + 1)^d$$

Soft Margin Dual Form

- By using Lagrangian Multiplier method

$$\min_{\boldsymbol{w},b,\xi_i} \frac{\|\boldsymbol{w}\|_2^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right) \quad \text{s.t. } y_i(\boldsymbol{w} \cdot \varphi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i,\ i = 1, \ldots, N,\ \xi_i \geq 0,\ i = 1, \ldots, N,$$

$$\min_{\boldsymbol{\lambda}} L_D(\boldsymbol{\lambda}) = -\left(\frac{1}{2}\sum_{i,j} \lambda_i\lambda_j y_i y_j (\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)) - \sum_{i=1}^{N} \lambda_i\right)$$

s.t. $0 \leq \lambda_i \leq C$ Kernel trick can be applied

$$\boldsymbol{w} \cdot \varphi(\boldsymbol{x}) + b = \left(\sum_{i=1}^{N} \lambda_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) (\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}))\right) + b = 0$$

Kernel trick can be applied

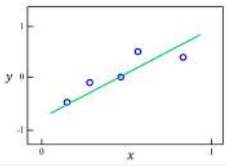
- For a test instance $\boldsymbol{x}^*$, it can be classified using

$$f(\boldsymbol{x}^*) = \text{sign}\left(\sum_{i=1}^{N} \lambda_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}^*) (\varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}^*)) + b\right)$$

$k(\boldsymbol{x}_i, \boldsymbol{x}^*)$

Regression: Example

Linear Fitting

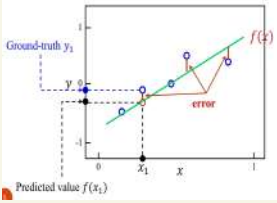


Linear Regression Model

- A special case, an instance is represented by one input feature
- To learn a linear function $f(x)$ in terms of w (drop bias term b for simplicity) from $\{x_i, y_i\}, i = 1, ..., N$

$$f(x) = w \cdot x$$

- Such that the difference (i.e., error) between the predicted values $f(x_i)$'s and the ground-truth values y_i's is as small as possible



- Suppose sum-of-squares error is used

$$L(w) = \frac{1}{2}\sum_{i=1}^{N}(f(x_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(w \times x_i - y_i)^2$$

- Learn the linear model in terms of w by minimizing the error

$$w^* = \arg\min_w L(w)$$

- To solve the unconstrained minimization problem, we can set the derivative of $E(w)$ w.r.t. w to zero

$$\frac{\partial L(w)}{\partial w} = \frac{\partial\left(\frac{1}{2}\sum_{i=1}^{N}(w \times x_i - y_i)^2\right)}{\partial w} = 0$$

$$\sum_{i=1}^{N}(w \times x_i - y_i) \times x_i = 0$$

$$w\sum_{i=1}^{N}x_i^2 - \sum_{i=1}^{N}y_i \times x_i = 0$$

$$w = \frac{\sum_{i=1}^{N} y_i \times x_i}{\sum_{i=1}^{N} x_i^2}$$

- To learn a linear function $f(x)$ in more general form in terms of w and b,

$$f(x) = w \cdot x + b$$

- By defining $w_0 = b$, and $X_0 = 1$, w and x are of $d + 1$ dimensions

$$f(x) = w \cdot x$$

- Suppose sum-of-squares error is used

$$L(w) = \frac{1}{2}\sum_{i=1}^{N}(f(x_i) - y_i)^2 = \frac{1}{2}\sum_{i=1}^{N}(w \cdot x_i - y_i)^2$$

- Learn the linear model in terms of w by minimizing the error

$$w^* = \arg\min_w L(w) + \frac{\lambda}{2}\|w\|_2^2$$

Positive tradeoff parameter

A regularization term to control the complexity of the model

Regularized Linear Regression

- To solve the unconstrained minimization problem, we can set the derivative of $L(w) + \frac{\lambda}{2}\|w\|_2^2$ w.r.t. w to zero

$$\frac{\partial\left(L(w) + \frac{\lambda}{2}\|w\|_2^2\right)}{\partial w} = 0$$

- We can obtain a closed-form solution for w by the above equations

Closed-Form Solution

- Denote by $\mathbf{X} = (x_1, x_2, ..., x_N)^T$

$$\mathbf{X} = \begin{pmatrix} x_{10} & \cdots & x_{N0} \\ \vdots & \ddots & \vdots \\ x_{1d} & \cdots & x_{Nd} \end{pmatrix}^T = \begin{pmatrix} x_{10} & \cdots & x_{1d} \\ \vdots & \ddots & \vdots \\ x_{N0} & \cdots & x_{Nd} \end{pmatrix}$$

- And by $\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}$
- The closed-form solution for w:

$$w = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

How to induce this closed-form solution?

Tutorial

Implementation Example

```
>>> from sklearn.linear_model import Ridge
...
>>> rlr = Ridge(alpha=0.1)
>>> rlr.fit(X, y)
>>> pred_train_rlr= rlr.predict(X)
```

Referred to as λ on our lecture notes

Evaluation

- Root Mean Square Error (RMSE)

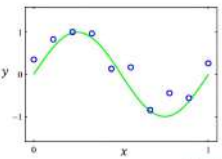
$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2}$$

- Mean Absolute Error (MAE)

$$\frac{1}{N}\sum_{i=1}^{N}|f(x_i) - y_i|$$

Nonlinear Fitting

Additional Notes (Optional)



kernel regression notes!

Regularized Linear Regression with Kernels

- Primal objective

$$\mathcal{J}(w) = \frac{1}{2}\sum_{i=1}^{N}(w \cdot \phi(x_i) - y_i)^2 + \frac{\lambda}{2}\|w\|_2^2$$

- It can be reformulated in terms of a dual form where kernel function arises naturally

$$\frac{\partial\mathcal{J}(w)}{\partial w} = 0 \Rightarrow \sum_{i=1}^{N}(w \cdot \phi(x_i) - y_i)\phi(x_i) + \lambda w = 0$$

Closed Form Solution

- By using kernel trick $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$

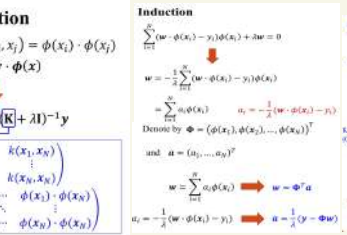
$$f(x) = w \cdot \phi(x)$$

$$k_i(x) = k(x_i, x)$$

$$f(x) = \mathbf{k}(x)^T(\mathbf{K} + \lambda\mathbf{I})^{-1}y$$

$$\mathbf{K} = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{pmatrix} = \begin{pmatrix} \phi(x_1)\cdot\phi(x_1) & \cdots & \phi(x_1)\cdot\phi(x_N) \\ \vdots & \ddots & \vdots \\ \phi(x_N)\cdot\phi(x_1) & \cdots & \phi(x_N)\cdot\phi(x_N) \end{pmatrix}$$

Induction



Lesson 9: Ensemble Learning

Ensemble Methods

- Objective:
  - To improve model performance in terms of accuracy by aggregating the predictions of multiple models
- How to do it?
  - Construct a set of base models from the training data
  - Make predictions by combing the predicted results made by each base model

"Two heads are better than one"

Why Ensemble Work?

- Suppose there are 3 base binary classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$ or accuracy acc = 0.65
  - Given a test instance, if we choose any one of these classifiers to make prediction, the probability that the classifier makes a wrong prediction is 35%

Base classifiers: f_1 f_2 f_3

A test instance: x

- Consider to combine the 3 base classifiers to make a prediction on a test instance using a majority vote
- The ensemble makes a wrong prediction only if more than 1 (i.e. 2 or 3) of the base classifiers predict incorrectly

majority

Truth label: -1

f_1	f_2	f_3	f_M	
+1	+1	+1	+1	✗
+1	+1	-1	+1	✗
+1	-1	+1	+1	✗
-1	+1	+1	+1	✗
+1	-1	-1	-1	✓
-1	+1	-1	-1	✓
-1	-1	+1	-1	✓
-1	-1	-1	-1	✓

The combined model makes a wrong prediction if at least two of the three base classifiers make a wrong prediction at the same time

- Assuming the three base classifiers are independent
  - The decision of one classifier does not tell us anything about another classifier.
- Each classifier has an error rate of $\epsilon = 35\%$
- What is the overall error rate of the ensemble?

Stories of Success

- Data mining competitions on Kaggle
  - Winning teams employ ensembles of classifiers

Feature Engineering ⟶ Ensemble Learning

independent

Truth label: -1

	f_1		f_2		f_3	f_M		
+1	35%	+1	35%	+1	35%	+1	✗	1st
+1	35%	+1	35%	-1	65%	+1	✗	2nd
+1		-1		+1		+1	✗	3rd
-1		+1		+1		+1	✗	4th
+1		-1		-1		-1	✓	
-1		+1		-1		-1	✓	
-1		-1		+1		-1	✓	
-1		-1		-1		-1	✓	

3 lines with the same probability

1st → probability of this happen 0.35, 0.35, 0.35

The combined model makes a wrong prediction if at least two of the three base classifiers make a wrong prediction at the same time

- Therefore, probability that the ensemble classifier makes a wrong prediction is:

$$\sum_{i=2}^{3}\binom{3}{i}\varepsilon^i(1-\varepsilon)^{3-i} = 3 \times 0.35^2 \times 0.65 + 1 \times 0.35^3 \times 1 = 0.2817$$

2,3,4 case — 1st case — $\binom{3}{2}\varepsilon^2(1-\varepsilon)^1$ — $\binom{3}{3}\varepsilon^3(1-\varepsilon)^0$

- Case 1: when there are two exact classifiers make wrong predictions, the probability is $\binom{3}{2}\varepsilon^2(1-\varepsilon)^{3-2}$
  - All possible combination; Two classifiers make wrong predictions; The rest one makes correct prediction
- Case 2: when all the three classifiers make wrong predictions, the probability is $\binom{3}{3}\varepsilon^3(1-\varepsilon)^{3-3}$

1-0.2817

- That is the accuracy of the ensemble classifier is 71.83%

$\varepsilon_{f_i} = 35\%$ ⟶ $\varepsilon_M = 28.17\%$

$\binom{3}{1} = 3$ $\binom{3}{2} = 3$ $\binom{3}{3} = 1$

- Suppose there are 25 independent base classifiers
  - Therefore, probability that the ensemble classifier makes a wrong prediction is:

$$\sum_{i=13}^{25}\binom{25}{i}\varepsilon^i(1-\varepsilon)^{25-i} = 0.06$$

- That is the accuracy of the ensemble classifier is 94%

$\varepsilon_{f_i} = 35\%$

$\varepsilon_M = 6\%$

Review: Joint Probability

- $P(X = 0, Y = 0)$ is the probability of both toss getting tail.
- What about $P(X = 1, Y = 0)$?
- What about $P(Y = 0, Y = 1)$? 0

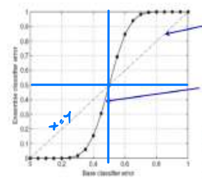
Review: Conditional Probability

- $P(X = 0|Y = 0)$ is the probability of the first toss being a tail, knowing the second toss is a tail.
- What about $P(Y = 0|X = 1)$?
- What about $P(X = 0|X = 0)$? 1

- Conditional and joint probabilities are related as
- $P(X = x, Y = y) = P(Y = y|X = x)P(X = x)$
- $P(X = x, Y = y) = P(X = x|Y = y)P(Y = y)$

Knowing that Y takes on value y, the probability of X takes on value x — The probability of Y taking on value y — The probability that Y takes on value y and X takes on value x

Error Rate of Base Classifiers



The base classifiers are identical (perfectly correlated)

The base classifiers are independent

Observation: the ensemble classifier performs worse than the base classifiers when the base classifier error rate is larger than 0.5

Error rate of an ensemble of 25 binary classifiers for different base classifier error rates

Correlation between Base Classifiers

- Sometimes the base classifiers are not completely independent.
- An extreme case is when they are perfectly correlated. That is, they always have the same predictions.

f_1	f_2
+1	+1
-1	-1
-1	-1
+1	+1
+1	+1

Perfectly positively correlated

USELESS

f_1	f_2
+1	-1
-1	+1
-1	+1
+1	-1
+1	-1

Perfectly negatively correlated

Review: Random Variables

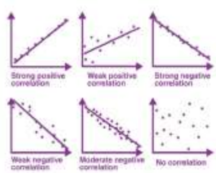
- Consider two random variables X and Y, which may take on values from {0, 1}.
  - Random variables are variables whose values are randomly assigned (from some unspecified random experiment).
- Imagine X and Y are the results of two coin tosses.
  - Head for the first toss ⇒ $X = 1$
  - Tail for the first toss ⇒ $X = 0$
  - Head for the second toss ⇒ $Y = 1$
  - Tail for the second toss ⇒ $Y = 0$
- The probabilities are denoted as $P(X = 0)$, $P(X = 1)$, $P(Y = 0)$, $P(Y = 0)$

Review: Independence

- If X and Y are independent, then we have
- $P(X = x|Y = y) = P(X = x)$
  - Y provides no knowledge about X
  - E.g., no matter the outcome of the first coin toss, the second toss is still 50/50
  - OR, no matter the second toss, the first toss is still 50/50
- Equivalently, $P(X = x, Y = y) = P(X = x)P(Y = y)$
- Why is this equivalent?

$$P(X = x, Y = y) = P(X = x|Y = y)P(Y = y) = P(X = x)P(Y = y)$$

Review: Correlation between RVs



One variable tells us something about the other variable.

Difficult Input Causes Correlation

- These are supposedly digits from 0 to 9.
- Most models probably make incorrect predictions on these data points.
- Having multiple models would not help too much here.

Necessary Conditions

- Two necessary conditions for an ensemble classifier to perform better than a single classifier:
  1. The base classifiers are not perfectly correlated with each other.
     - In practice, the base classifiers are usually somewhat correlated.
  2. The base classifiers should do better than a classifier that performs random guessing (e.g., for binary classification, accuracy should be better than 0.5)

Tutorial

Ensemble Methods

- How to generate a set of base classifiers?
  - By manipulating the training set: multiple training sets are created by resampling the original data according to some sampling distribution. A classifier is then trained from each training set, such as Bagging, Boosting
  - By manipulating the input features: a subset of input features is chosen to form each training set. A classifier is then built from each training set, such as Random forest
  - By manipulating the learning algorithm(s): applying the algorithm several times on the same training data using different parameters or applying different algorithms
- How to combine the base classifiers for predictions?

General Procedure

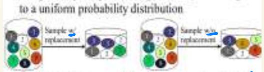
1. Let D denote the original training data, k denote the number of base classifiers, and T be the test dataset.
2. **for** $i = 1$ to k **do**
3. Train a base classifier f_i from D
4. **end for**
5. **for** each test instance $x \in T$ **do**
6. Generate $f_1(x)$, $f_2(x)$, ..., and $f_k(x)$
7. Calculate $f_M(x)$ = Merge($f_1(x), f_2(x), ..., f_k(x)$)
8. **end for**

For example, majority voting (can be other schemes)

Bagging

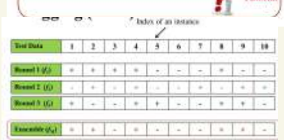
bootstrapping: Create multi copy of training set that r not identical

- Known as bootstrap aggregating (bootstrapping), to repeatedly sample with replacement according to a uniform probability distribution



- Build classifier on each bootstrap sample, which is of the same size of the original data
- Use majority voting to determine the class label of ensemble classifier

Bagging (cont.)



Majority Voting

[optional] $\lim_{n\to\infty}(1+\frac{x}{n})^n = e^x$

- Suppose a training set D contains N examples
- A training instance has a probability of $1 - \frac{1}{N}$ of *not* being selected
- Its probability of ending up *not* in a training set D_i is $\left(1-\frac{1}{N}\right)^N \approx \frac{1}{e} = 0.368$
- A bootstrap sample D_i contains approximately 63.2% of the original training data

Implementation Example

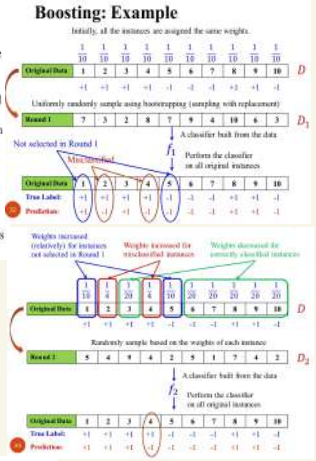
Boosting

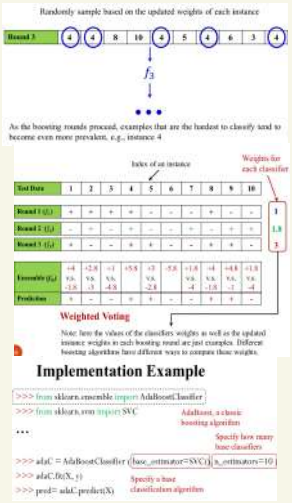
- Principles:
  - Boost a set of weak learners to a strong learner
  - Make instances currently misclassified more important
- Generally,
  - To adaptively change the distribution of training data so that the base classifiers will focus more on previously misclassified records
- Specifically,
  - Initially, all N instances are assigned equal weights
  - Unlike bagging, weights may change at the end of each boosting round
  - In each boosting round, after the weights are assigned to the training instances,
    - Draw a bootstrap sample from the original data by using the weights as a sampling distribution to build a model

Boosting: Procedure

1. Initially, all instances are assigned equal weights $\frac{1}{N}$ so that they are equally likely to be chosen for training. A sample is drawn uniformly to obtain a new training set.
2. A classifier is induced from the training set, and used to classify all the examples in the original training set.
3. The weights of the training instances are updated at the end of each boosting round
   - Instances that are wrongly classified will have their weights increased
   - Instances that are classified correctly will have their weights decreased
4. Repeat Steps 2 and 3 until the stopping condition is met
5. Finally, the ensemble is obtained by aggregating the base classifiers obtained from each boosting round

Boosting: Example





Implementation Example

```
>>> from sklearn.ensemble import AdaBoostClassifier
>>> from sklearn.svm import SVC
...
>>> adaC = AdaBoostClassifier(base_estimator=SVC(), n_estimators=10)
>>> adaC.fit(X, y)
>>> pred= adaC.predict(X)
```

AdaBoost, a classic boosting algorithm

Specify how many base classifiers

Specify a base classification algorithm

Random Forests

- A class of ensemble methods specifically designed for decision tree classifiers
- Random Forests grow many trees
- Each tree is generated based on a random subset of features
- Final result on classifying a new instance – voting
  - Forest chooses the classification result having the most votes (over all the trees in the forest)

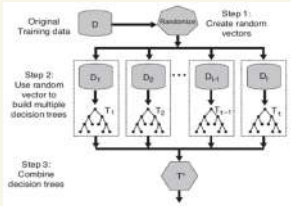
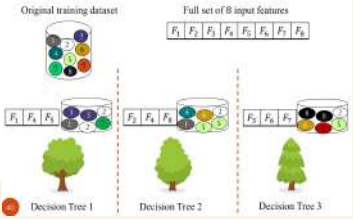


Illustration of random forests

Random Forests: Algorithm

- Choose T: number of trees to grow
- Choose $m' < m$ (m is the number of total features): number of features used to calculate the best split at each node (typically 20%)
- For each tree
  - Choose a training set via bootstrapping
  - For each node, randomly choose m' features and calculate the best split
  - Fully grown and not pruned
- Use majority vote among all the trees

Example



Random Forests: Discussions

- Bagging + random features
- Improve Accuracy
  - Incorporate more diversity
- Improve Efficiency
  - Searching among subsets of features is much faster than searching among the complete set

Implementation Example

```
>>> from sklearn.ensemble import RandomForestClassifier
...
>>> rfC = RandomForestClassifier(n_estimators = 20, max_depth = 3)
>>> rfC.fit(X, y)
>>> pred= rfC.predict(X)
```

Set parameter for the base tree

Specify how many base classifiers

Combination Methods

- Voting
  - Majority voting
  - Weighted voting
- Average
  - Simple average
  - Weighted average
- Combining by learning

Voting

- Majority voting:
  - Takes the class label that receives the largest number of votes as the final winner
- Weighted voting:
  - A generalized version of majority voting by introducing weights for each classifier

Average

classification (for binary)

- Simple average:

$$f_M(x) = \frac{1}{T}\sum_{i=1}^{T} f_i(x)$$

- Weighted average:

$$f_M(x) = \sum_{i=1}^{T} w_i f_i(x)$$

where $w_i \geq 0$, and $\sum_{i=1}^{T} w_i = 1$

- Prediction:

$$\hat{y}(x) = \begin{cases} 1, & f_M(x) \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

Combining by Learning

- Stacking:
  - A general procedure where a learner is trained to combine the individual learners
  - Individual learners: first-level learners
  - Combiner: second-level learner, or meta-learner
  - Suppose given a binary classification problem,

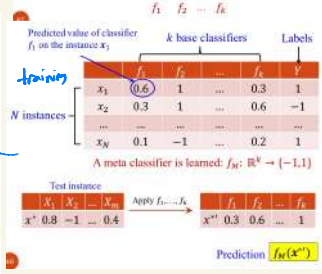
Combining by Learning: Summary

- Represent each training instances using classifier-generated outputs

$$x'_i = (f_1(x_i), f_2(x_i), \ldots, f_k(x_i))$$

- Learn a meta classifier f_M from $\{x'_i, y_i\}, i = 1, \ldots, N$.
- For a test instance x^*
  - Represent it by $x^{*\prime} = (f_1(x^*), f_2(x^*), \ldots, f_k(x^*))$
  - Use f_M to make a prediction $f_M(x^{*\prime})$

Similar to multi layer perceptron

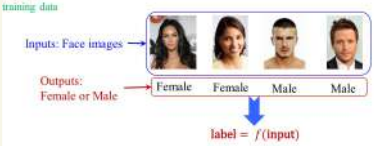


Lesson 10: Clustering

Supervised Learning

- The examples presented to computers are pairs of inputs and the corresponding outputs, the goal is to "learn" a mapping or model from inputs to labels

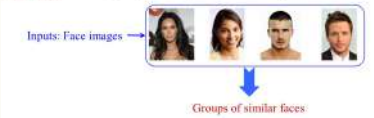
Labeled training data



Unsupervised Learning

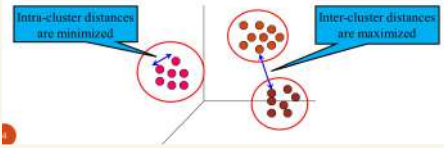
- The examples presented to computers are a set of inputs without any outputs, the goal is to "learn" an intrinsic structure of the examples, e.g., clusters of examples, density of the examples

Unlabeled training data



What is Cluster Analysis?

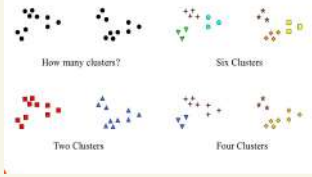
- Finding groups of data instances such that the data instances in a group are
  - similar to one another
  - different from data instances in other groups



What is not Cluster Analysis?

- Supervised classification
  - Have class label information
- Simple segmentation
  - Dividing customers into different groups alphabetically, by last name

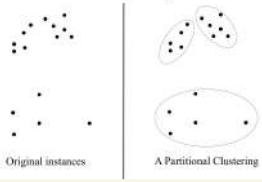
Notion of a Cluster can be Ambiguous



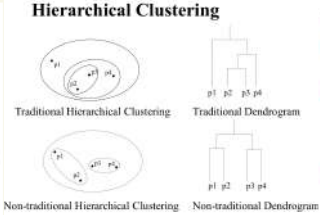
Types of Clusterings

- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
- Partitional clustering
  - Divide data instances into **non-overlapping** subsets (clusters) so that each data instance is in exactly one subset
- Hierarchical clustering
  - A set of **nested** clusters organized as a hierarchical tree

Partitional Clustering



Hierarchical Clustering



Other Distinctions Between Clusterings

- Exclusive versus non-exclusive
  - In non-exclusive clustering, instances may belong to multiple clusters
- Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
- Partial versus complete
  - In partial clustering, only some of the instances are clustered

Clustering Algorithms

- *K*-means and its variants
- Hierarchical clustering

K-means Clustering

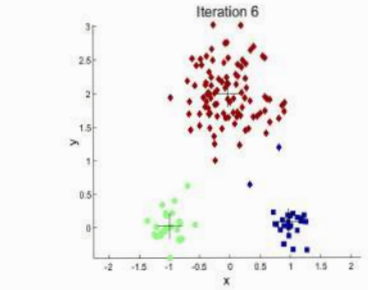
k = number of clus

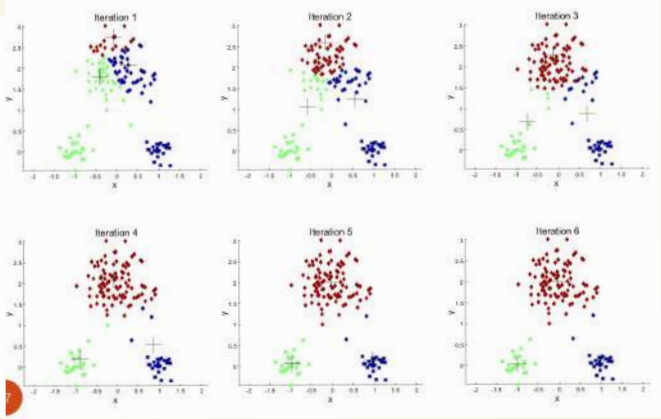
- Partitional clustering approach
- Number of clusters, *K*, must be specified
- Each cluster is associated with a centroid (center point) ← close to
- Each point is assigned to the cluster with the closest centroid
- Basic algorithm:
  1. Select *K* data instances as the initial centroids
  2. **Repeat**
  3. Form *K* clusters by assigning all data instances to the closest centroid
  4. Recompute the centroid of each cluster → Calculate average
  5. **Until** The centroids do not change

K-means Clustering – Details

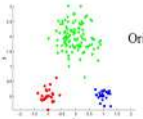
- Initial centroids are often chosen randomly
- The centroid is (typically) the mean of the data instances in the cluster
- 'Closeness' is measured by a proximity
  - Distance: Euclidean distance, etc
  - Similarity: cosine similarity, correlation, etc
- *K*-means will converge for common similarity measures mentioned above
  - In practice, it converges in the first few iterations
  - Often the stopping condition is changed to 'Until relatively few instances change clusters'

K-means Illustration

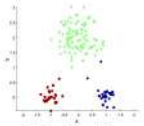




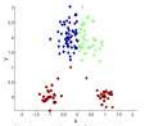
Two different K-means Clusterings



Original Points



Optimal Clustering



Sub-optimal Clustering

Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each data instance, the "error" is the distance to the nearest cluster that is represented by a centroid
  - To get an overall SSE, we square these errors and sum them

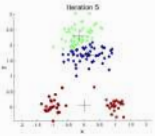
$$\text{Total SSE} \quad \text{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} \text{dist}(c_i, x)^2$$

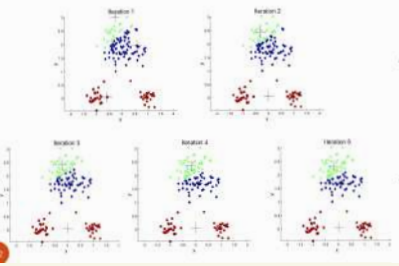
c_i → Centroid of the cluster C_i

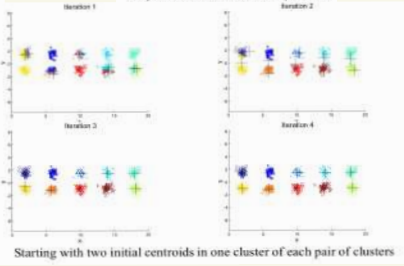
$$\text{Cluster SSE for cluster } C_i = \sum_{x \in C_i} \text{dist}(c_i, x)^2$$

- Given two different runs of K-means, we can choose the one with the smallest Total SSE
- One easy way to reduce SSE is to increase K, the number of clusters
  - However, a good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Importance of Initial Centroids

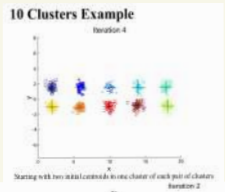


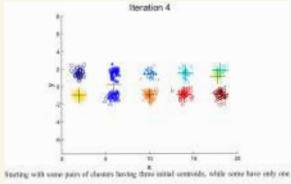


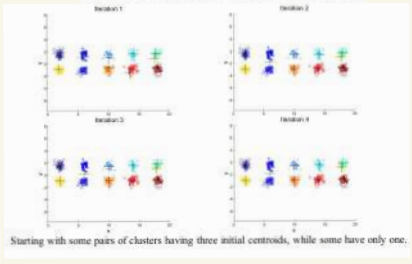


Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example







Starting with some pairs of clusters having three initial centroids, while some have only one.

Initial Centroids Issue

- Multiple runs
- Postprocessing
  - Decompose a cluster with high Cluster SSE,
  - Merge clusters with low Cluster SSE, which are close to each other
- Bisecting K-means

Pre-processing and Post-processing

- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Post-processing
  - Eliminate small clusters that may represent outliers
  - Split 'loose' clusters, i.e., clusters with relatively high SSE
  - Merge clusters that are 'close' and that have relatively low SSE

Empty Clusters Issue

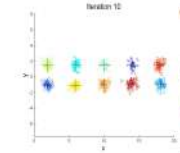
- Basic K-means algorithm can yield empty clusters
- Several strategies to choose a replacement centroid
  - Choose the point that contributes most to SSE
  - Choose a point from the cluster with the highest Cluster SSE
  - If there are several empty clusters, the above can be repeated several times

Bisecting K-means

- Basic algorithm:

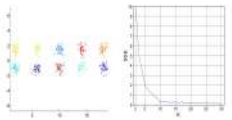
1. Initialize the list of clusters to be a single cluster that contains all points
2. **Repeat**
3. Select a cluster with highest SSE from the list of clusters
4. **For** $i = 1$ to T **do**
5. Bisect the selected cluster using basic K-means
6. **End**
7. Add the two clusters from the bisection with lowest SSE to the list of clusters
8. **Until** the list of clusters contains K clusters
9. (After bisecting) run K-means with K clusters found so far

Bisecting K-means



Estimation of K

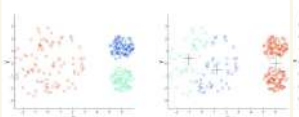
- SSE can be used to estimate the number of clusters



Limitations of K-means

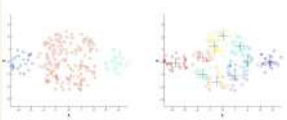
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has problems when the data contains outliers

Limitations of K-means: Differing Density



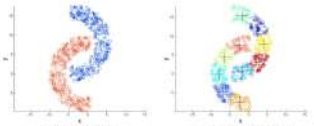
Original Points K-means (3 Clusters)

Solution



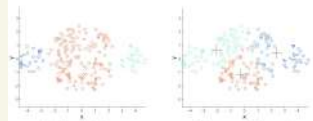
Original Points K-means Clusters

One solution is to use many clusters. Find parts of clusters, but need to put together.



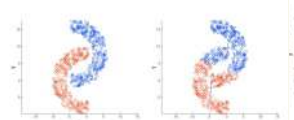
Original Points K-means Clusters

Limitations of K-means: Differing Sizes

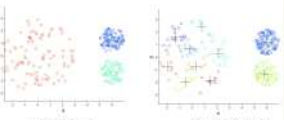


Original Points K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes



Original Points K-means (2 Clusters)

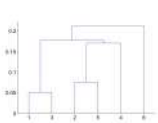


Original Points K-means Clusters

Hierarchical Clustering

don't know how to select

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits



Dendrogram

Nested cluster diagram

Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- They may correspond to meaningful taxonomies
  - Examples in document organization, biological sciences

Hierarchical Clustering (cont.)

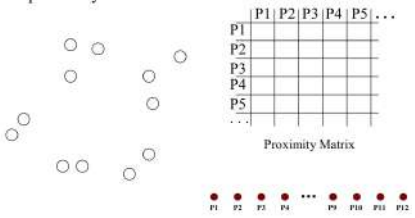
- Two main types of hierarchical clustering
  - Agglomerative (bottom-up):
    - Start with the instances as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or K clusters) left
  - Divisive (top-down):
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are K clusters)
- Traditional hierarchical algorithms use a proximity matrix (similarity or distance) to merge or split one cluster at a time

Agglomerative Clustering Algorithm

- Basic algorithm:
  1. Compute the proximity matrix — distance or similarity
  2. Let each data instance be a cluster
  3. **Repeat** — smallest distance or largest similarity
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is to compute the proximity of two clusters
  - Different approaches to defining the proximity between clusters lead to different clustering results

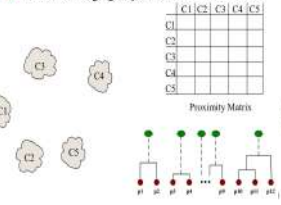
Starting Situation

- Start with clusters of individual instances and a proximity matrix



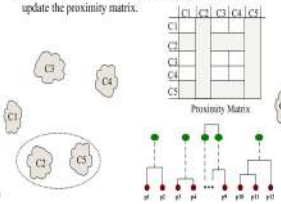
Intermediate Situation

- After some merging steps, we have some clusters



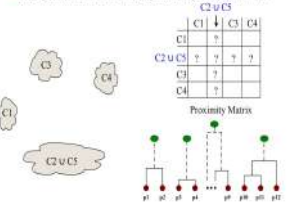
Intermediate Situation...

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.

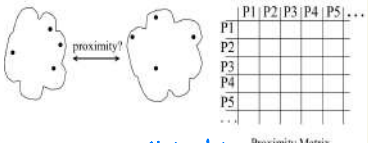


After Merging

- The question is "How do we update the proximity matrix?"

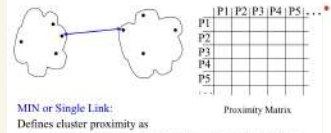


Define Inter-Cluster Proximity



- MIN or Single Link — Shortest dist
- MAX or Complete Link
- Group Average

Inter-Cluster Similarity (I)



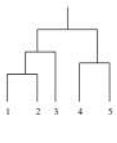
MIN or Single Link:
Defines cluster proximity as

- the proximity between the closest two data points that are in different clusters
- or the shortest edge (single link) between two nodes in different subsets (using graph terms)

MIN or Single Link

- Similarity of two clusters is based on the two closest points (most similar) in the different clusters

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00



Clustering with MIN

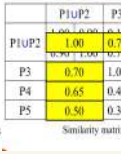
Step 1: Merge the two closest clusters (largest similarity) — merge threshold = 1 = ulers; 0.9 = highest

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00

Clustering with MIN (cont.)

- Step 2: Update proximity matrix based on MIN: proximity of two clusters is based on the two closest points in different clusters (largest similarity) — get max

	P1∪P2	P3	P4	P5
P1∪P2	1.00	0.70	0.65	0.50
P3	0.70	1.00	0.40	0.30
P4	0.65	0.40	1.00	0.80
P5	0.50	0.30	0.80	1.00



Clustering with MIN (cont.)

Step 1: Merge the two closest clusters (largest similarity)

	P1∪P2	P3	P4	P5
P1∪P2	1.00	0.70	0.65	0.50
P3	0.70	1.00	0.40	0.30
P4	0.65	0.40	1.00	0.80
P5	0.50	0.30	0.80	1.00

Clustering with MIN (cont.)

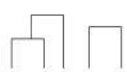
- Step 2: Update proximity matrix based on MIN: proximity of two clusters is based on the two closest points in different clusters (largest similarity)

	P1∪P2	P3	P4∪P5
P1∪P2	1.00	0.70	0.65
P3	0.70	1.00	0.40
P4∪P5	0.65	0.40	1.00

Clustering with MIN (cont.)

Step 1: Merge the two closest clusters (largest similarity)

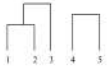
	P1∪P2	P3	P4∪P5
P1∪P2	1.00	0.70	0.65
P3	0.70	1.00	0.40
P4∪P5	0.65	0.40	1.00



Clustering with MIN (cont.)

- Step 2: Update proximity matrix based on MIN: proximity of two clusters is based on the two closest points in different clusters (largest similarity)

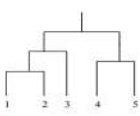
	P1∪P2∪P3	P4∪P5
P1∪P2∪P3	1.00	0.65
P4∪P5	0.65	1.00



Clustering with MIN (cont.)

Step 1: Merge the two closest clusters (largest similarity)

	P1∪P2∪P3	P4∪P5
P1∪P2∪P3	1.00	0.65
P4∪P5	0.65	1.00



What about the proximity matrix is a distance matrix?

Tutorial

attention — pick min — pick max

distance/similarity matrix

Column on Join ?

single = smallest distance / max similarity

Complete = max distance / min similarity

Strengths of Hierarchical Clustering

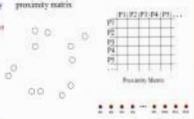
- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- They may correspond to meaningful taxonomies
  - Examples in document organization, biological sciences

Agglomerative Clustering Algorithm

- Basic algorithm:
  1. Compute the proximity matrix (distance or similarity)
  2. Let each data instance be a cluster
  3. **Repeat**
  4. Merge the two closest clusters (smallest distance or largest similarity)
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is to compute the proximity of two clusters
  - Different approaches to defining the proximity between clusters lead to different clustering results

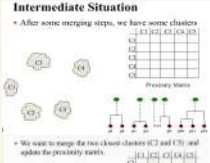
Starting Situation

- Start with clusters of individual instances and a proximity matrix

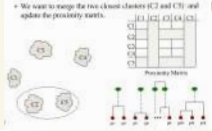


Intermediate Situation

- After some merging steps, we have some clusters

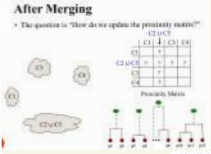


- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.

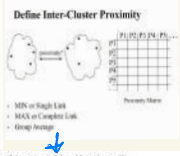


After Merging

- The question is "How do we update the proximity matrix?"

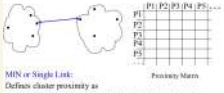


Define Inter-Cluster Proximity



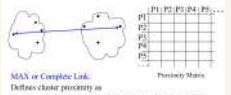
- MIN or Single Link
- MAX or Complete Link
- Group Average

Inter-Cluster Similarity (I)



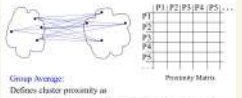
MIN or Single Link:
Defines cluster proximity as
- the proximity between the closest two data points that are in different clusters
- or the shortest edge (single link) between two nodes in different subsets (using graph terms)

Inter-Cluster Similarity II



MAX or Complete Link:
Defines cluster proximity as
- the proximity between the farthest two points that are in different clusters
- or the longest edge (complete link) between two nodes in

Inter-Cluster Similarity (III)

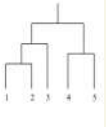


Group Average:
Defines cluster proximity as
- the average pairwise proximities of all pairs of points from different clusters
- Or average length of edges between nodes in different subsets (using graph terms)

MIN or Single Link

- Similarity of two clusters is based on the two closest points (most similar) in the different clusters

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00



Similarity matrix

Clustering with MIN

Step 1: Merge the two closest clusters (largest similarity)

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00

Similarity matrix

- Step 2: Update proximity matrix based on MIN: proximity of two clusters is based on the two closest points in different clusters (largest similarity)

	P1∪P2	P3	P4	P5
P1∪P2	1.00	0.70	0.65	0.50
P3	0.70	1.00	0.40	0.30
P4	0.65	0.40	1.00	0.80
P5	0.50	0.30	0.80	1.00

Similarity matrix

Step 1: Merge the two closest clusters (largest similarity)

	P1∪P2	P3	P4	P5
P1∪P2	1.00	0.70	0.65	0.50
P3	0.70	1.00	0.40	0.30
P4	0.65	0.40	1.00	0.80
P5	0.50	0.30	0.80	1.00

Similarity matrix

- Step 2: Update proximity matrix based on MIN: proximity of two clusters is based on the two closest points in different clusters (largest similarity)

	P1∪P2	P3	P4∪P5
P1∪P2	1.00	0.70	0.65
P3	0.70	1.00	0.40
P4∪P5	0.65	0.40	1.00

Similarity matrix

Step 1: Merge the two closest clusters (largest similarity)

	P1∪P2	P3	P4∪P5
P1∪P2	1.00	0.70	0.65
P3	0.70	1.00	0.40
P4∪P5	0.65	0.40	1.00

Similarity matrix

- Step 2: Update proximity matrix based on MIN: proximity of two clusters is based on the two closest points in different clusters (largest similarity)

	P1∪P2∪P3	P4∪P5
P1∪P2∪P3	1.00	0.65
P4∪P5	0.65	1.00

Similarity matrix

Step 1: Merge the two closest clusters (largest similarity)

	P1∪P2∪P3	P4∪P5
P1∪P2∪P3	1.00	0.65
P4∪P5	0.65	1.00

Similarity matrix

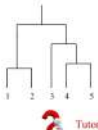
What about the proximity matrix is a distance matrix?

Tutorial

MAX or Complete Link

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00



Similarity matrix

Tutorial

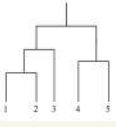
Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters

$$\text{Proximity}(C_i, C_j) = \frac{\sum_{x_i \in C_i, x_j \in C_j} \text{Proximity}(x_i, x_j)}{|C_i| \times |C_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00

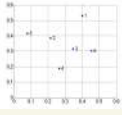


Agglomerative Clustering: Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty in handling different-sized clusters

Divisive Hierarchical Clustering

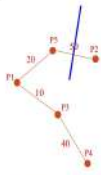
- Basic algorithm:
  1. Generate a minimum spanning tree to connect all data instances as a single cluster
  2. **Repeat**
  3. Create a new cluster by breaking the link corresponding to the largest distance (smallest similarity)
  4. **Until** only singleton clusters remain
- Minimum Spanning Tree (MST)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points (x_i, x_j) such that one point (x_i) is in the current tree but the other (x_j) is not
  - Add (x_j) to the tree and put an edge between x_i and x_j



An Example of DHC

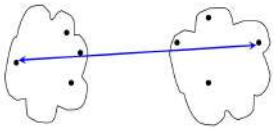
The distance matrix between 5 points

	P1	P2	P3	P4	P5
P1	0	90	10	65	20
P2	90	0	70	60	50
P3	10	70	0	40	30
P4	65	60	40	0	80
P5	20	50	30	80	0



Suppose $K = 2$, and P3 is chosen at the beginning for constructing the MST

Inter-Cluster Similarity II



	P1	P2	P3	P4	P5	...
P1						
P2						
P3						
P4						
P5						
...						

Proximity Matrix

MAX or Complete Link:

Defines cluster proximity as

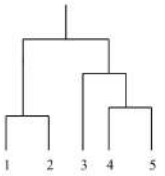
- the proximity between the farthest two points that are in different clusters
- or the longest edge (complete link) between two nodes in different subsets (using graph terms)

MAX or Complete Link

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters

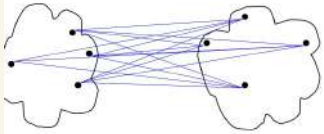
	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00

Similarity matrix



Tutorial

Inter-Cluster Similarity (III)



	P1	P2	P3	P4	P5	...
P1						
P2						
P3						
P4						
P5						
...						

Proximity Matrix

Group Average:

Defines cluster proximity as

- the average pairwise proximities of all pairs of points from different clusters
- Or average length of edges between nodes in different subsets (using graph terms)

due to Outlier bias

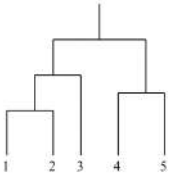
Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters

$$\text{Proximity}(C_i, C_j) = \frac{\sum_{x_i \in C_i, x_j \in C_j} \text{Proximity}(x_i, x_j)}{|C_i| \times |C_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters

	P1	P2	P3	P4	P5
P1	1.00	0.90	0.10	0.65	0.20
P2	0.90	1.00	0.70	0.60	0.50
P3	0.10	0.70	1.00	0.40	0.30
P4	0.65	0.60	0.40	1.00	0.80
P5	0.20	0.50	0.30	0.80	1.00



Agglomerative Clustering: Limitations

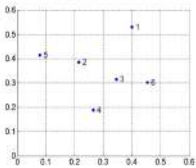
- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty in handling different-sized clusters

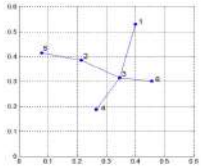
Divisive Hierarchical Clustering

- Basic algorithm:
  1. Generate a minimum spanning tree to connect all data instances as a single cluster
  2. **Repeat**
  3. Create a new cluster by breaking the link corresponding to the largest distance (smallest similarity)
  4. **Until** only singleton clusters remain

Divisive Hierarchical Clustering (cont.)

- Minimum Spanning Tree (MST)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points (x_i, x_j) such that one point (x_i) is in the current tree but the other (x_j) is not
  - Add (x_j) to the tree and put an edge between x_i and x_j





An Example of DHC

no cycle

The distance matrix between 5 points

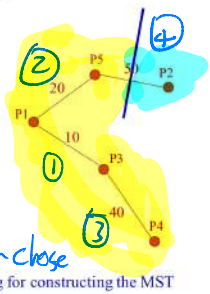
	P1	P2	P3	P4	P5
P1	~~0~~	90	~~70~~	~~65~~	~~20~~
P2	90	~~0~~	70	60	50
P3	~~70~~	70	~~0~~	~~40~~	~~30~~
P4	~~65~~	60	40	~~0~~	~~80~~
P5	~~20~~	50	~~30~~	~~80~~	~~0~~

See column cross all p when chose

Suppose $K = 2$, and P3 is chosen at the beginning for constructing the MST

pick longest distance

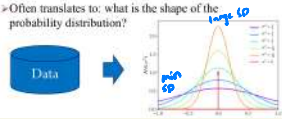
K=2 (2 clusters)



Lesson 11: Density Estimation

Purpose of Machine Learning

- Model Uncertainty
  - Germany will probably beat Japan, but what are the odds? 60/40, 70/30, or 80/20?
  - I'm willing to bet more money if the odds are in my favor.
  - Often translates to: what is the shape of the probability distribution?



Why Estimating Distributions

- Recall Naïve Bayes Classifiers

$$y_i = \arg\max_c P(\boldsymbol{x}|y=c)P(y=c)$$

$$= \arg\max_c \prod_{i=1}^{d} P(x_i|y=c)\,P(y=c)$$

A naïve, simplifying assumption, which we may remove if we can estimate the distribution properly.

Discrete vs Continuous Probability Distributions

Discrete Probability Distribution

- Usually a finite number of outcomes
- A 6-sided die ⇒ 6 possible outcomes
- The distribution can be described with six numbers
  - Non-negative
  - Sum up to 1

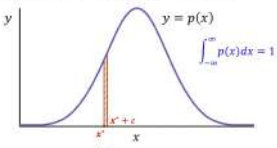
Outcome	1	2	3	4	5	6
Probability	0.1	0.2	0.1	0.3	0.05	0.25

Continuous Probability Distribution

- The outcome is a real number in a range
  - For example, $[0, 1]$, $(-\infty, \infty)$
- An infinite number of outcomes between any two real numbers.
- This is really tricky.
- There is a formal branch of mathematics (measure theory) that deal with this, though we will not introduce it here.

Probability Density Function

Key message: Probability is area under the curve



$$P(x^*) = \int_{x^*}^{x^*+\varepsilon} p(x)dx \approx p(x^*) \times \varepsilon$$

Density Estimation

- Density estimation aims to estimate an unobservable underlying probability density function based on observed data
- Denote by $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ the set of observed data points, drawn from an unknown $p(\boldsymbol{x})$,

$$\boldsymbol{x}_i \sim p(\boldsymbol{x}), \text{ for } i = 1, 2, \ldots, N$$

The goal is to estimate the probability density function $p(\boldsymbol{x})$

State-of-the-Art Density Estimation

- Generative Adversarial Networks
  - The competition drives the counterfeiter to learn the distribution of real data.

Stability AI, the startup behind Stable Diffusion, raises $101M

Stable diffusion is an image generating AI, similar to DALL-E and Midjourney.

Density Estimation Approaches

- Parametric density estimation
  - Assume a form for $p(\boldsymbol{x}|\boldsymbol{\theta})$, defined up to parameters, $\boldsymbol{\theta}$
    - E.g., Gaussian distribution $N(\mu, \sigma^2)$, $\boldsymbol{\theta} = \{\mu, \sigma^2\}$
  - Estimate $\boldsymbol{\theta}$ from the observed data points
    - Maximum Likelihood Estimation
- Nonparametric density estimation

The General Principle

- The observed data points $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ are assumed to be a sample of N random variables independent and identically distributed (i.i.d.)
- Identically distributed: for any $\boldsymbol{x}_i \in \mathcal{D}$, it is sampled from the same probability distribution
- Independent: all the data points $\boldsymbol{x}_i \in \mathcal{D}$ are independent events

Parametric Density Estimation

- Assume that $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ are drawn from some known probability density family, $P(\boldsymbol{x}|\boldsymbol{\theta})$, defined up to parameters, $\boldsymbol{\theta}$

$$\boldsymbol{x}_i \sim P(\boldsymbol{x}|\boldsymbol{\theta})$$

- We seek $\boldsymbol{\theta}$ that makes $\boldsymbol{x}_i$ as likely as possible under $P(\boldsymbol{x}|\boldsymbol{\theta})$
- Approach: maximum likelihood estimation

Maximum Likelihood Estimation

- Likelihood of parameter $\boldsymbol{\theta}$ given sample $\mathcal{D}$:

$$l(\boldsymbol{\theta}; \mathcal{D}) \triangleq P(\mathcal{D}|\boldsymbol{\theta})$$

- As $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ are i.i.d., the above likelihood is the product of the likelihoods of the individual data points

$$l(\boldsymbol{\theta}; \mathcal{D}) \triangleq P(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N} P(\boldsymbol{x}_i|\boldsymbol{\theta})$$

- In MLE, we aim to find $\boldsymbol{\theta}$ that makes $\mathcal{D}$ the most likely to be drawn from. Mathematically, we aim to search for $\hat{\boldsymbol{\theta}}$ such that

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} l(\mathcal{D}; \boldsymbol{\theta})$$

- Typically, we maximize the log-likelihood:

$$\mathcal{L}(\mathcal{D}; \boldsymbol{\theta}) \triangleq \ln l(\mathcal{D}; \boldsymbol{\theta})$$

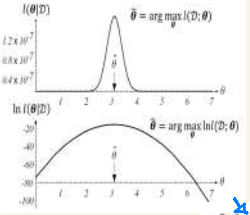
- Why?
  1. The $\ln(\cdot)$ function converts the product into a sum

$$\ln l(\mathcal{D}; \boldsymbol{\theta}) = \ln P(\mathcal{D}|\boldsymbol{\theta}) = \ln\left(\prod_{i=1}^{N} P(\boldsymbol{x}_i|\boldsymbol{\theta})\right) = \sum_{i=1}^{N} \ln P(\boldsymbol{x}_i|\boldsymbol{\theta})$$

  2. The $\ln(\cdot)$ function is a strictly increasing function, one can maximize the likelihood without changing the value where it takes its maximum

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} l(\mathcal{D}; \boldsymbol{\theta}) \Leftrightarrow \hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \ln l(\mathcal{D}; \boldsymbol{\theta})$$

MLE Illustration



Solution of MLE

- Suppose $\boldsymbol{\theta}$ contains p parameters,

$$\boldsymbol{\theta} = [\theta_1\ \theta_2\ \ldots\ \theta_p]^T$$

- $\max_{\boldsymbol{\theta}} \ln l(\mathcal{D}; \boldsymbol{\theta})$ is an unconstrained optimization problem.

To solve it, we first set the derivative of $\ln l(\mathcal{D}; \boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$ to zero

necessary condition

$$\nabla_{\boldsymbol{\theta}} \ln l(\mathcal{D}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\left(\sum_{i=1}^{N} \ln P(\boldsymbol{x}_i|\boldsymbol{\theta})\right) = \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \ln P(\boldsymbol{x}_i|\boldsymbol{\theta}) = \boldsymbol{0},$$

Gradient operator

$$\nabla_{\boldsymbol{\theta}} = \left[\frac{\partial}{\partial\theta_1}\ \frac{\partial}{\partial\theta_2}\ \ldots\ \frac{\partial}{\partial\theta_p}\right]^T$$

- We then obtain a solution $\hat{\boldsymbol{\theta}}$ by solving the above system of equations

Univariate Gaussian

- Suppose $\mathcal{D} = \{x_1, x_2, ..., x_N\}$. Each data point x_i is a scalar, and is drawn from a Guassian distribution with unknown μ and σ^2:

$$P(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

$$\mu = \mathbb{E}[x] = \int P(x;\mu,\sigma^2)x\,dx$$

$$\sigma^2 = \mathrm{Var}(x) = \mathbb{E}[(x-\mathbb{E}[x])(x-\mathbb{E}[x])^T]$$

Expectation, considering all possible values (infinite)

- The log-likelihood is

$$\ln l(\mathcal{D};\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln P(x_i|\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)\right) = \sum_{i=1}^{N} \ln\left(\exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)\right) - \sum_{i=1}^{N} \ln(\sqrt{2\pi\sigma^2})$$

$$= -\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{2\sigma^2} - \frac{N}{2}\ln(2\pi\sigma^2) = -\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{2\sigma^2} - \frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln\sigma^2$$

- The log-likelihood:

$$\ln l(\mathcal{D};\boldsymbol{\theta}) = -\frac{\sum_{i=1}^{N}(x_i-\mu)^2}{2\sigma^2} - \frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln\sigma^2$$

- The derivative of the log-likelihood:

$$\nabla_{\boldsymbol{\theta}} \ln l(\mathcal{D};\boldsymbol{\theta}) = \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \ln P(x_i|\boldsymbol{\theta}) = \begin{bmatrix} \sum_{i=1}^{N} \nabla_{\mu} \ln P(x_i|\boldsymbol{\theta}) \\ \sum_{i=1}^{N} \nabla_{\sigma} \ln P(x_i|\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i-\mu) \\ \frac{\sum_{i=1}^{N}(x_i-\mu)^2}{2(\sigma^2)^2} - \frac{N}{2\sigma^2} \end{bmatrix}$$

- By setting the derivative to be zero:

$$\begin{cases} \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i-\mu) = 0 \\ \frac{\sum_{i=1}^{N}(x_i-\mu)^2}{2(\sigma^2)^2} - \frac{N}{2\sigma^2} = 0 \end{cases}$$

- We have

$$\begin{cases} \hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \\ \hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i-\hat{\mu})^2 \end{cases}$$

Unbiased Estimator

- An estimator is a rule for estimating a quantity based on observations.
- The MLE estimator for the Gaussian mean is $\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$
- An estimator is unbiased if its expectation is the same as the true quantity.

Unbiased estimation

$$\mathbb{E}[\hat{\mu}] = \mu \leftarrow \text{True}$$

$$\mathbb{E}[\hat{\sigma}^2] = \frac{N-1}{N}\sigma^2$$

To correct bias

$$\tilde{\sigma}^2 = \frac{N}{N-1}\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\hat{\mu})^2$$

Wait... Expectation of an Estimator?

- How can you talk about an expectation without a probability distribution?
- Imagine you want to estimate the average height of Singaporeans.
- Conceptually simple: measure everyone!
- Actually feasible: measure 100 randomly selected Singaporeans.
- However, the estimate will change depending on who are selected.
- These different estimators form a distribution.
- The calculation of this distribution is out of the scope of this course.

Multivariate Gaussian

- Suppose $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, and each data point $\boldsymbol{x}_i$ has d dimensions, and is drawn from a Guassian distribution with unknown $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$

($\boldsymbol{\mu}$: d-dimensional mean vector; $\boldsymbol{\Sigma}$: $d \times d$ covariance matrix)

$$P(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

($|\boldsymbol{\Sigma}|$: the determinant of $\boldsymbol{\Sigma}$)

$$\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}] = \int P(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})\boldsymbol{x}\,d\boldsymbol{x}$$

$$\boldsymbol{\Sigma} = \mathbb{E}[(\boldsymbol{x}-\mathbb{E}[\boldsymbol{x}])(\boldsymbol{x}-\mathbb{E}[\boldsymbol{x}])^T]$$

- The log-likelihood of multivariate Gaussian is

$$\ln l(\mathcal{D};\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln P(\boldsymbol{x}_i;\boldsymbol{\mu},\boldsymbol{\Sigma}) = -\frac{Nd}{2}\ln(2\pi) - \frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})$$

- The derivative of the log-likelihood w.r.t. $\boldsymbol{\mu}$ is

$$\frac{\partial}{\partial\boldsymbol{\mu}} \ln l(\mathcal{D};\boldsymbol{\theta}) = \sum_{i=1}^{N} \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu})$$

- By setting the derivative to zero, we have

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$$

Unbiased estimation: $\mathbb{E}[\hat{\boldsymbol{\mu}}] = \boldsymbol{\mu}$

- By computing the derivative of the log-likelihood w.r.t.$\boldsymbol{\Sigma}$ (each Σ_{ij}, i.e., the entry of the i-th row and the j-th column), and setting to be zero, we have

$$\hat{\boldsymbol{\Sigma}}_{\text{MLE}} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})(\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})^T \quad \text{MLE estimator}$$

- The above estimation is biased!

$$\mathbb{E}[\hat{\boldsymbol{\Sigma}}] = \frac{N-1}{N}\boldsymbol{\Sigma}$$

- We can correct the bias by timing $\hat{\boldsymbol{\Sigma}}$ with $\frac{N}{N-1}$:

$$\tilde{\boldsymbol{\Sigma}}_{\text{Unbiased}} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})(\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})^T \quad \text{Unbiased estimator (not MLE)}$$

Covariance Matrix $\boldsymbol{\Sigma}$

Variance of the first dimension; Covariance of the first and second dimensions

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathrm{Var}(x^{(1)}) & \mathrm{Cov}(x^{(1)},x^{(2)}) & \cdots & \mathrm{Cov}(x^{(1)},x^{(d)}) \\ \mathrm{Cov}(x^{(2)},x^{(1)}) & \mathrm{Var}(x^{(2)}) & \cdots & \mathrm{Cov}(x^{(2)},x^{(d)}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(x^{(d)},x^{(1)}) & \mathrm{Cov}(x^{(d)},x^{(2)}) & \cdots & \mathrm{Var}(x^{(d)}) \end{bmatrix}$$

$$\mathrm{Var}(x^{(i)}) = \mathbb{E}\left[(x^{(i)}-\mathbb{E}[x^{(i)}])^2\right]$$

$$\mathrm{Cov}(x^{(i)},x^{(j)}) = \mathbb{E}[(x^{(i)}-\mathbb{E}[x^{(i)}])(x^{(j)}-\mathbb{E}[x^{(j)}])] = \int (x^{(i)}-\mathbb{E}[x^{(i)}])(x^{(j)}-\mathbb{E}[x^{(j)}])P(x^{(i)})P(x^{(j)})dx^{(i)}dx^{(j)}$$

Exam last year

Summary: Estimating Multivariate Gaussian

- Suppose $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$. Each data point $\boldsymbol{x}_i$ has d dimensions, and is drawn from a Guassian distribution with unknown $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$:

$$P(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- The unbiased estimators for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i$$

$$\tilde{\boldsymbol{\Sigma}} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})(\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})^T$$

Note: in this module, wherever you are asked to estimate the mean vector and the covariance matrix of data samples, use the unbiased estimators

Is MLE always the best choice?

- What if you only have a single data point $\boldsymbol{x}_1$?
- The MLE $\boldsymbol{\mu} = \boldsymbol{x}_1$
- Estimating the entire distribution based on a single data point seems unwise.

Summary: MLE for General Distributions

- Suppose $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$. Each data point $\boldsymbol{x}_i$ is of d dimensions, and drawn from a distribution with unknown parameter $\boldsymbol{\theta}$

$$\boldsymbol{x}_i \sim P(\boldsymbol{x}|\boldsymbol{\theta})$$

- Step 1: Compute the likelihood of $\boldsymbol{\theta}$ given sample $\mathcal{D}$:

$$l(\mathcal{D};\boldsymbol{\theta}) \triangleq P(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N} P(\boldsymbol{x}_i|\boldsymbol{\theta})$$

- Step 2: Compute the log-likelihood of $\boldsymbol{\theta}$ given $\mathcal{D}$: $\ln l(\mathcal{D};\boldsymbol{\theta})$
- Step 3: Compute the derivative of $\ln l(\boldsymbol{\theta}|\mathcal{D})$ w.r.t. $\boldsymbol{\theta}$ and set it to zero:

$$\nabla_{\boldsymbol{\theta}} \ln l(\boldsymbol{\theta}|\mathcal{D}) = \mathbf{0}$$

- Step 4: Solve the above system of equations to obtain the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$

Centered data matrix

$$\tilde{\boldsymbol{\Sigma}} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})(\boldsymbol{x}_i-\hat{\boldsymbol{\mu}})^T = \frac{1}{N-1}\sum_{i=1}^{N}\tilde{\boldsymbol{x}}_i\tilde{\boldsymbol{x}}_i^T = \frac{1}{N-1}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}$$

The Bayesian Approach

- We may introduce a prior distribution $P(\boldsymbol{\theta})$ that represents our belief about θ in the absence of any data.

$$P(\boldsymbol{\theta}|\mathcal{D}) = \frac{P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\mathcal{D})} = \frac{P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{\int P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}}$$

(Likelihood, Prior, Posterior, Normalization constant)

- If the likelihood is overly concentrated, the prior can provide some smoothing effect.

$$P(\boldsymbol{\theta}|\mathcal{D}) = \frac{P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\mathcal{D})} = \frac{P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{\int P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}}$$

(Likelihood, Prior, Posterior, Normalization constant)

Density Estimation Approaches

- Parametric density estimation
- Nonparametric density estimation
  - Without assuming any forms for the underlying density
  - Assume that similar inputs have similar outputs: if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar, then $P(\boldsymbol{x}_i)$ and $P(\boldsymbol{x}_j)$ are similar
  - Approaches
    - Histogram Estimator
    - Naïve Estimator / Parzen Windows / Kernel Estimator
    - K-NN Estimator

Non-Parametric Density Estimation

- Assume that $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$ are drawn from some unknown probability density $P(\boldsymbol{x})$
- To learn the estimator $\hat{P}(\boldsymbol{x})$ for $P(\boldsymbol{x})$
- We first focus on the univariate case, i.e., x_i is scalar
- Note that the introduced approaches can be generalized to the multivariate case easily

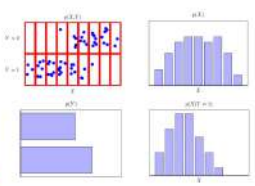
Look-up table for Volume of an n-ball

https://en.wikipedia.org/wiki/Volume_of_an_n-ball

Dimension	Volume of a ball of radius R	Radius of a ball of volume V
0	1	(all 0-balls have volume 1)
1	$2R$	$\frac{V}{2} \approx 0.5 \times V$
2	$\pi R^2 \approx 3.142 \times R^2$	$\frac{V^{\frac{1}{2}}}{\sqrt{\pi}} \approx 0.564 \times V^{\frac{1}{2}}$
3	$\frac{4\pi}{3} R^3 \approx 4.189 \times R^3$	$\left(\frac{3V}{4\pi}\right)^{\frac{1}{3}} \approx 0.620 \times V^{\frac{1}{3}}$
4	$\frac{\pi^2}{2} R^4 \approx 4.935 \times R^4$	$\frac{(2V)^{\frac{1}{4}}}{\sqrt{\pi}} \approx 0.671 \times V^{\frac{1}{4}}$
5	$\frac{8\pi^2}{15} R^5 \approx 5.264 \times R^5$	$\left(\frac{15V}{8\pi^2}\right)^{\frac{1}{5}} \approx 0.717 \times V^{\frac{1}{5}}$
6	$\frac{\pi^3}{6} R^6 \approx 5.168 \times R^6$	$\frac{(6V)^{\frac{1}{6}}}{\sqrt{\pi}} \approx 0.761 \times V^{\frac{1}{6}}$
7	$\frac{16\pi^3}{105} R^7 \approx 4.725 \times R^7$	$\left(\frac{105V}{16\pi^3}\right)^{\frac{1}{7}} \approx 0.801 \times V^{\frac{1}{7}}$
8	$\frac{\pi^4}{24} R^8 \approx 4.059 \times R^8$	$\frac{(24V)^{\frac{1}{8}}}{\sqrt{\pi}} \approx 0.833 \times V^{\frac{1}{8}}$
9	$\frac{32\pi^4}{945} R^9 \approx 3.299 \times R^9$	$\left(\frac{945V}{32\pi^4}\right)^{\frac{1}{9}} \approx 0.878 \times V^{\frac{1}{9}}$
10	$\frac{\pi^5}{120} R^{10} \approx 2.550 \times R^{10}$	$\frac{(120V)^{\frac{1}{10}}}{\sqrt{\pi}} \approx 0.911 \times V^{\frac{1}{10}}$
11	$\frac{64\pi^5}{10\,395} R^{11} \approx 1.884 \times R^{11}$	$\left(\frac{10\,395V}{64\pi^5}\right)^{\frac{1}{11}} \approx 0.944 \times V^{\frac{1}{11}}$
12	$\frac{\pi^6}{720} R^{12} \approx 1.335 \times R^{12}$	$\frac{(720V)^{\frac{1}{12}}}{\sqrt{\pi}} \approx 0.976 \times V^{\frac{1}{12}}$
13	$\frac{128\pi^6}{135\,135} R^{13} \approx 0.911 \times R^{13}$	$\left(\frac{135\,135V}{128\pi^6}\right)^{\frac{1}{13}} \approx 1.007 \times V^{\frac{1}{13}}$
14	$\frac{\pi^7}{5040} R^{14} \approx 0.599 \times R^{14}$	$\frac{(5040V)^{\frac{1}{14}}}{\sqrt{\pi}} \approx 1.037 \times V^{\frac{1}{14}}$
15	$\frac{256\pi^7}{2\,027\,025} R^{15} \approx 0.381 \times R^{15}$	$\left(\frac{2\,027\,025V}{256\pi^7}\right)^{\frac{1}{15}} \approx 1.066 \times V^{\frac{1}{15}}$
n	$v_n(R)$	$R_n(V)$

54

Histogram Estimator



- Simply partition x into distinct bins of a fixed width Δ
- Count the number N_t of data points falling into bin t
- Turn this count into a normalized probability density via dividing by the total number of observed data points N and by the width Δ of the bins:

$$P_t = \frac{N_t}{N\Delta}$$ Why divide by Δ?

- The model for the density $p(\boldsymbol{x})$ is constant over the width of each bin: find the bin where $\boldsymbol{x}$ is in (e.g., bin t), then

$$\hat{P}(\boldsymbol{x}) = \frac{\#\{\boldsymbol{x}_i \mid \boldsymbol{x}_i \text{ in the same bin as } \boldsymbol{x}\}}{N\Delta} = P_t$$

- For a bin t, given a density function, the probability that a data instance falling into the bin t is

$$P_t = \int_{\Delta} p(x)dx = p_t(x)\Delta$$

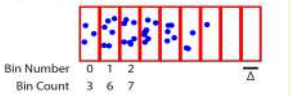
- On the other hand,

$$P_t = \frac{\#\{\boldsymbol{x}_i \mid \boldsymbol{x}_i \text{ in bin } t\}}{N} = \frac{N_t}{N}$$

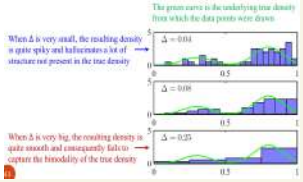
- Therefore:

$$P_t(x)\Delta = \frac{N_t}{N} \Rightarrow P_t(x) = \frac{N_t}{N\Delta}$$

- Note: different bins may have different widths Δ_t in general, but in practice, we use the same width Δ



- Histogram density as a function of bin width Δ



Analysis on Histogram Estimator

- Advantages:
  - Simple to evaluate and simple to use
  - One can throw away $\mathcal{D}$ once the histogram is computed
  - Can be updated incrementally
- Disadvantages:
  - The estimated density has discontinuities due to the bin edges rather than any property of the underlying density
  - Scales poorly to multivariate cases: we would have m^d *bins* (hypercubes) if we divided each feature (dimension) in a d-dimensional space into m bins

Naïve Estimator: An Alterative

- In Histogram Estimator, besides Δ, we have to choose an origin x_0 as well, the bins are the intervals defined as

$$[x_0 + m\Delta, x_0 + (m+1)\Delta)$$

m is an integer

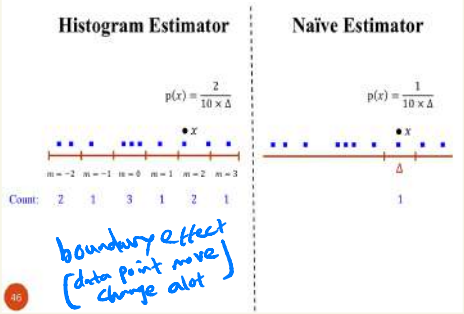
- The Naïve Estimator does not need to set an origin

$$\hat{p}(\boldsymbol{x}) = \frac{\#\{\boldsymbol{x}_i \mid \boldsymbol{x}_i \text{ in the same bin as } \boldsymbol{x}\}}{N\Delta}$$

Given $\boldsymbol{x}$, use $\boldsymbol{x}$ as a center to create a bin with a length of Δ

$$\hat{p}(\boldsymbol{x}) = \frac{\#\{\boldsymbol{x}_i \mid \boldsymbol{x} - \frac{\Delta}{2} \le \boldsymbol{x}_i < \boldsymbol{x} + \frac{\Delta}{2}\}}{N\Delta}$$

Histogram v.s. Naïve Estimator



46

$$\hat{P}(\boldsymbol{x}) = \frac{\#\{\boldsymbol{x}_i \mid \boldsymbol{x} - \frac{\Delta}{2} \le \boldsymbol{x}_i < \boldsymbol{x} + \frac{\Delta}{2}\}}{N\Delta}$$

The Naïve Estimator can also be written as

$$\hat{P}(\boldsymbol{x}) = \frac{1}{N\Delta}\sum_{i=1}^{N} w\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}}{\Delta}\right)$$

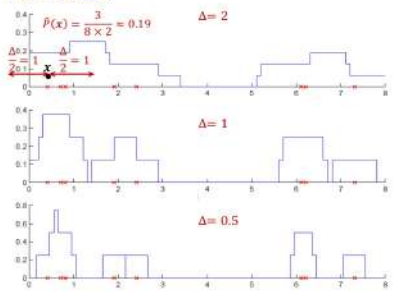
If x_i is in the bin with width Δ centered at x, then the count is increased by 1

$$u = \frac{\boldsymbol{x}_i - \boldsymbol{x}}{\Delta}$$

Windowing function $w(u) = \begin{cases} 1 & \text{if } -\frac{1}{2} \le u < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$

Parzen windows

Naïve Estimator



Generalization to Multivariate

- Suppose the observed data points $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ are d-dimensional
- In a d-dimensional space, we define $\mathcal{R}$ is a d-dimensional hypercube with h being the length of each edge. Then the volume of the hypercube is given by

$$V = h^d$$

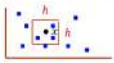
- The windowing function can be defined as

$$w(\boldsymbol{u}) = \begin{cases} 1 & \text{if } -\frac{1}{2} \le u_j < \frac{1}{2} \text{ for all } j \in \{1,2,\ldots,d\} \\ 0 & \text{otherwise} \end{cases}$$

$$\boldsymbol{u} = \frac{\boldsymbol{x}_i - \boldsymbol{x}}{h}$$

Defines a hypercube of length h centered at $\boldsymbol{x}$, if $\boldsymbol{x}_i$ falls in the cube, then the count is increased by 1

E.g., $d = 2$



- Hence, $w\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}}{h}\right)$ is equal to unity if $\boldsymbol{x}_i$ falls within the hypercube of volume V centered at $\boldsymbol{x}$, and is zero otherwise
- The density estimator can be written as

$$\hat{P}(\boldsymbol{x}) = \frac{\#\{\boldsymbol{x}_i \mid \boldsymbol{x}_i \text{ in the same hypercube as } \boldsymbol{x}\}}{NV}$$

$$\hat{P}(\boldsymbol{x}) = \frac{1}{NV}\sum_{i=1}^{N} w\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}}{h}\right)$$

Parzen window function can be a kernel function – Kernel estimator (Appendix, optional)

$$\hat{P}(\boldsymbol{x}) = \frac{3}{10h^2}$$

K-Nearest Neighbor Estimator

- Recall

$$P(\boldsymbol{x}) = \frac{\#\{\boldsymbol{x}_i \mid \boldsymbol{x}_i \text{ in the same hypercube as } \boldsymbol{x}\}}{VN} \quad (K_{\boldsymbol{x}})$$

- In the previous approaches, V (or Δ for univariate) is fixed for different queries $\boldsymbol{x}$'s
- The K-NN Estimator *adapts* the amount of smoothing to the *local* density of data, and the degree of smoothing is controlled by K, the number of neighbors

Consider K nearest neighbors of $\boldsymbol{x}$

$$P(\boldsymbol{x}) = \frac{K}{NV_{\boldsymbol{x}}}$$

The volume of the space centered at $\boldsymbol{x}$ that exactly contains K nearest neighbors of $\boldsymbol{x}$

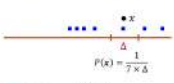
- Note that the number of nearest neighbors is typically much smaller than the number of training data points, i.e., $K \ll N$

Naïve Estimator v.s. K-NN Estimator

Univariate Case

Naïve Estimator

$$\hat{P}(\boldsymbol{x}) = \frac{\#\{\boldsymbol{x}_i \mid \boldsymbol{x} - \frac{\Delta}{2} \le \boldsymbol{x}_i < \boldsymbol{x} + \frac{\Delta}{2}\}}{N\Delta}$$

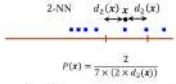


$$\hat{P}(\boldsymbol{x}) = \frac{1}{7 \times \Delta}$$

- Fix Δ, and check how many data points fall in the bin

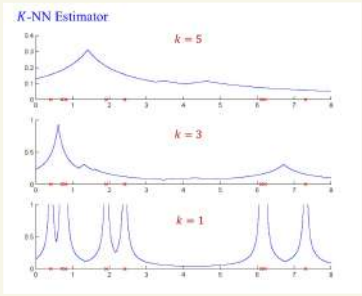
K-NN Estimator

$$\hat{P}(\boldsymbol{x}) = \frac{K}{N(2d_K(\boldsymbol{x}))}$$



$$P(\boldsymbol{x}) = \frac{2}{7 \times (2 \times d_2(\boldsymbol{x}))}$$

- Fix K, the number of observed data points to fall in the bin, and compute the bin size



- With a predefined K, for a particular data point $\boldsymbol{x}$,
  1. Compute distance between $\boldsymbol{x}$ and all the observed data, e.g., Euclidean distance $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2$
  2. Sort the observed data points based on the distances in ascending order:

$$d_1(\boldsymbol{x}) \le d_2(\boldsymbol{x}) \le \cdots d_j(\boldsymbol{x}) \le \cdots \le d_N(\boldsymbol{x})$$

$d_1(\boldsymbol{x})$ is the distance of $\boldsymbol{x}$ to the nearest observed instance

$d_j(\boldsymbol{x})$ is the distance of $\boldsymbol{x}$ to the j-th nearest observed instance

  3. The K-NN density estimate is

A ball in d-dimensional Euclidean space

$$\hat{P}(\boldsymbol{x}) = \frac{K}{NV_{d_K(\boldsymbol{x})}}$$

The volume of the d-ball of the radius $d_K(\boldsymbol{x})$ centered at $\boldsymbol{x}$. And $d_K(\boldsymbol{x})$ is the distance of $\boldsymbol{x}$ to the K-th nearest observed instance

K-NN Estimator

- The K-NN Estimator *adapts* the amount of smoothing to the *local* density of data, and the degree of smoothing is controlled by K, the number of neighbors

Consider K nearest neighbors of $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \frac{K}{NV_{\boldsymbol{x}}}$$

The volume of the space centered at $\boldsymbol{x}$ that exactly contains K nearest neighbors of $\boldsymbol{x}$

- For the multivariate case

$$\hat{p}(\boldsymbol{x}) = \frac{K}{NV_{d_K(\boldsymbol{x})}}$$

The volume of the d-ball of the radius $d_K(\boldsymbol{x})$ centered at $\boldsymbol{x}$. And $d_K(\boldsymbol{x})$ is the distance of $\boldsymbol{x}$ to the K-th nearest observed instance

- For the univariate case

$$\hat{p}(\boldsymbol{x}) = \frac{K}{N2d_K(\boldsymbol{x})}$$

Appendix: Kernel Estimator

Optional

- To get a smooth estimate, we use a smooth weight function, *kernel function*, e.g., the Gaussian kernel
- For the univariate case, i.e., each data point is 1-dimensional, the Gaussian kernel is defined as

$$k(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$$

- The Kernel Estimator is computed via

$$\hat{P}(x) = \frac{1}{N\Delta}\sum_{i=1}^{N} k\left(\frac{x_i - x}{\Delta}\right)$$

Kernel Estimator (cont.)

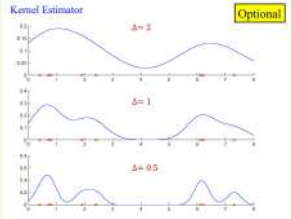
Optional

- For the multivariate case, i.e., each data point is d-dimensional, the Gaussian kernel is defined as

$$k(\boldsymbol{u}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{\|\boldsymbol{u}\|_2^2}{2}\right)$$

- The Kernel Estimator is computed via

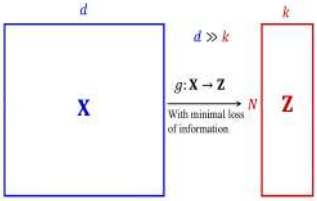
$$\hat{P}(\boldsymbol{x}) = \frac{1}{NV}\sum_{i=1}^{N} k\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}}{h}\right)$$



Lesson 12: Dimensionality Reduction

High-level Idea

- To summarize observed high-dimensional data points with low-dimensional vectors

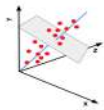


Why Dimensionality Reduction

- To avoid curse of dimensionality
  - Decision boundary in SVM: $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
  - Linear Regression: $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$
  - Perceptron: $f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x})$
  - One parameter to learn for every input dimension
  - Difficult to accurately estimate the best parameters when the number of data points is small
- To identify the features, or the transformations of features that capture the most important data characteristics
- All measurements contain error or noise. Mathematically we may write

$$\boldsymbol{x} = \bar{\boldsymbol{x}} + \boldsymbol{e}, \boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \sigma \boldsymbol{I})$$

- All measurements contain error or noise
  $\boldsymbol{x} = \bar{\boldsymbol{x}} + \boldsymbol{e}, \boldsymbol{e} \sim \mathcal{N}(\boldsymbol{0}, \sigma \boldsymbol{I})$
- Valid hypothesis: Data reside on a straight line, but the noise is isotropic (equal amount of variation) in all three dimensions.



What is noise?

- These photos depict the same person
- Different lighting, make-up, facial hair, expressions, etc., create visual differences
- The data variation that we care about is far smaller than the pixel-level differences.

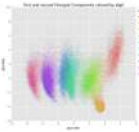
- The MNIST dataset
- Each column shows the same number.
- Some noise, which we do not have names for, change how they look.
- The data variation that we care about is far smaller than the pixel-level differences

Why Dimensionality Reduction

- Thus, we should identify important variations in the data and discard the noise.
- Benefits
  - Reduces storage requirements
  - Allows visualization in 2D or 3D
  - Reduces noise and improve the performance of machine learning

A Case Study: MNIST

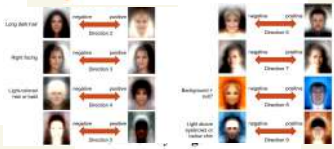
- We use PCA to reduce the number of dimensions to 2 and visualize the results.
- Some clustering structure



https://lookylook.github.io/2018/08/13/visualising-mnist-pca-tsne/

A Case Study: Eigenface

- We use PCA to identify 10 directions that capture most variations in the data.
- We visualize them by adding each direction to the average face, without which the faces are hard to interpret and look scary.
- The first direction seems to be mostly about lighting and background.



- PCA directions capture variations in data, which correspond to some semantic categories.
- But they are not always interpretable.
- PCA is linear. It would struggle to capture semantic categories that require non-linear variations, such as formal vs. casual dress.

Dimensionality Reduction Approaches

- Feature Selection
  - To select a subset of k features from the original d features to represent each data instance
    - Brute-force approach
    - Greedy search
- Feature Extraction
  - To learn k new features from the original d features to represent each data instance
    - Linear combination of original features
      - Principal component analysis
    - Nonlinear combination of original features

Principal Component Analysis

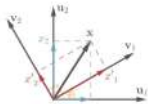
- One of the most widely-used (unsupervised) dimensionality reduction methods
- Takes a data matrix of N data points by d features, and summarizes it by principal components that are linear combinations of the original d variables
- The first k components display as much as possible of the variation among data instances

A Case Study: Eigenface

- PCA on images of human faces
- A technique initially created for face identification.

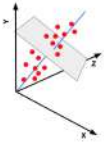
Linear Algebra: Change of Basis

- Vector $\boldsymbol{x} = (x_1, x_2) = x_1\boldsymbol{u}_1 + x_2\boldsymbol{u}_2$ where $\boldsymbol{u}_1 = (1, 0)$ and $\boldsymbol{u}_2 = (0, 1)$ are the basis vectors.
- Can we change the basis vectors to $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$?
- $\boldsymbol{x} = x'_1\boldsymbol{v}_1 + x'_2\boldsymbol{v}_2$. We just need to find x'_1 and x'_2 by projecting $\boldsymbol{x}$ onto $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$.

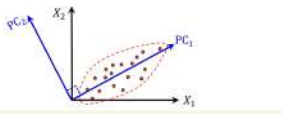


PCA: Geometric Rationale

- Data may only occupy a small subspace of the high-dimensional $\mathbb{R}^d$ space.
- We want to pick a few ($< d$) basis vectors that data are projected onto.
- Which basis vectors would you pick?



- Goal: to find a projection or rotation of the original d-dimensional coordinate system to capture the largest amount of variation in data
  - Ordered s.t. the 1st principal component has the highest variance, the 2nd component has the next highest variance, … , the d-th component has the lowest variance
  - Principal components are orthogonal to each other



PCA: Algorithm

Input: $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ a set of observed data

1. Centering the data points s.t. the mean is $\boldsymbol{0}$

$$\bar{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i \longrightarrow \boldsymbol{x}_i = \boldsymbol{x}_i - \bar{\boldsymbol{\mu}}$$

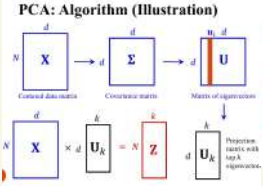
2. Compute sample covariance matrix

$$\boldsymbol{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N} \boldsymbol{x}_i\boldsymbol{x}_i^T$$

Each $\boldsymbol{u}_i$ is of d dimensions

3. Compute eigenvectors of $\boldsymbol{\Sigma}$, $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_d\}$, which are sorted based on their eigenvalues in non-increasing order, i.e., $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$
4. Select the first k eigenvectors to construct principal components

PCA: Algorithm (Illustration)



Derivation of PCA

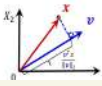
- The variance preservation view
  - The first k components display as much as possible of the variation among data instances
- The minimum reconstruction view
  - The first k components convey maximum useful information of original data instances

Appendix (optional)

- Consider a projection of a data point $\boldsymbol{x}$ onto a vector $\boldsymbol{v}$
- The projection of $\boldsymbol{x}$ onto $\boldsymbol{v}$ is

$$\frac{\boldsymbol{v}^T\boldsymbol{x}}{\|\boldsymbol{v}\|_2} = \frac{\|\boldsymbol{v}\|_2\|\boldsymbol{x}\|_2\cos(\theta)}{\|\boldsymbol{v}\|_2} = \|\boldsymbol{x}\|_2\cos(\theta)$$

- For simplicity, consider $\boldsymbol{u}$ with unit length, i.e., $\|\boldsymbol{v}\|_2 = 1$



Eigenvalues & Eigenvectors

- Given a d-by-d square matrix $\mathbf{A}$, if there exists a non-zero d-dimensional vector $\boldsymbol{u}$, s.t.,

$$\mathbf{A}\boldsymbol{u} = \lambda\boldsymbol{u} \quad \text{scalar}$$

then $\boldsymbol{u}$ is an eigenvector of $\mathbf{A}$, and λ is called the corresponding eigenvalue

- Notes:
  - There are d eigenvectors and eigenvalues
  - An eigenvalue can be positive, negative or zero
  - An eigenvector cannot be a zero vector
  - For symmetric matrices, eigenvectors are orthogonal to each other

Properties of Eigenvalues

Given a square matrix $\mathbf{A}$ (d-by-d)

- $\mathbf{A}$ is invertible ($\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ or $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$) if all the eigenvalues of $\mathbf{A}$ are non-zero (positive or negative)
- If all the eigenvalues of $\mathbf{A}$ are non-negative, then $\mathbf{A}$ is a positive semi-definite matrix:
  For any non-zero vector $\boldsymbol{x} \in \mathbb{R}^{d\times 1}$, we have $\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} \geq 0$
- If all the eigenvalues of $\mathbf{A}$ are positive, then $\mathbf{A}$ is a positive definite matrix:
  For any non-zero vector $\boldsymbol{x} \in \mathbb{R}^{d\times 1}$, we have $\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} > 0$
- Recall: when inducing a closed form solution of regularized linear regression model, we mentioned that if a matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}, \text{where } \mathbf{X} \in \mathbb{R}^{N\times d}, \mathbf{I} \in \mathbb{R}^{d\times d} \text{and } \lambda > 0$$

then $\mathbf{A}$ is always invertible:

$$\exists\, \mathbf{A}^{-1}, \text{s.t., } \mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \text{ and } \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$$

- We first prove $\mathbf{A}$ is positive definite
  - For any non-zero vector $\boldsymbol{x} \in \mathbb{R}^{d\times 1}$

$$\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} = \boldsymbol{x}^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\boldsymbol{x}$$
$$= \boldsymbol{x}^T(\mathbf{X}^T\mathbf{X})\boldsymbol{x} + \boldsymbol{x}^T(\lambda\mathbf{I})\boldsymbol{x}$$

Denote $\boldsymbol{z} = \mathbf{X}\boldsymbol{x}$

$$= \boldsymbol{z}^T\boldsymbol{z} + \lambda\boldsymbol{x}^T\boldsymbol{x}$$
$$= \|\boldsymbol{z}\|_2^2 + \lambda\|\boldsymbol{x}\|_2^2$$

$\|\boldsymbol{z}\|_2^2 \geq 0$ and $\|\boldsymbol{z}\|_2^2 = 0$ if and only if $\boldsymbol{z} = \boldsymbol{0}$

$\|\boldsymbol{x}\|_2^2 > 0$ because $\boldsymbol{x} \neq \boldsymbol{0} \Rightarrow \lambda\|\boldsymbol{x}\|_2^2 > 0$ as long as $\lambda > 0$

$$\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} > 0$$

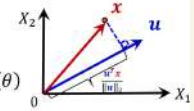
- As $\mathbf{A}$ is positive definite, all of its eigenvalues are positive, i.e., non-zero
- Recall: $\mathbf{A}$ is invertible if all the eigenvalues of $\mathbf{A}$ are non-zero (either positive or negative)
- Therefore, if a matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}, \text{where } \mathbf{X} \in \mathbb{R}^{N\times d}, \mathbf{I} \in \mathbb{R}^{d\times d} \text{and } \lambda > 0$$

then $\mathbf{A}$ is invertible!

PCA: Variance Preservation

- The first k components display as much as possible of the variation among data instances
- Consider a projection of a data point $\boldsymbol{x}$ onto a vector going through the origin, represented by $\boldsymbol{u}$
- The projection of $\boldsymbol{x}$ onto $\boldsymbol{u}$ is



$$\frac{\boldsymbol{u}^T\boldsymbol{x}}{\|\boldsymbol{u}\|_2} = \frac{\|\boldsymbol{u}\|_2\|\boldsymbol{x}\|_2\cos(\theta)}{\|\boldsymbol{u}\|_2} = \|\boldsymbol{x}\|_2\cos(\theta)$$

- For simplicity, consider $\boldsymbol{u}$ with unit length, i.e., $\|\boldsymbol{u}\|_2 = 1$
- The projected instances $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ onto $\boldsymbol{u}$ are

$$\{\boldsymbol{u}^T\boldsymbol{x}_1, \boldsymbol{u}^T\boldsymbol{x}_2, \dots, \boldsymbol{u}^T\boldsymbol{x}_N\}$$

- In PCA, data points are centered at the beginning

$$\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i = 0$$

- After projection onto $\boldsymbol{u}$, the mean of data points is still 0

$$\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{u}^T\boldsymbol{x}_i = \boldsymbol{u}^T\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i = 0$$

- The variance of the data points projected onto $\boldsymbol{u}$ is

$$\frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{u}^T\boldsymbol{x}_i - 0)^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{u}^T\boldsymbol{x}_i)^2$$

Each row is a data instance

$$= \boldsymbol{u}^T\underbrace{\frac{1}{N-1}\sum_{i=1}^{N}\boldsymbol{x}_i\boldsymbol{x}_i^T}_{\boldsymbol{\Sigma}} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}$$

- The goal of PCA (for simplicity, projected on 1 principal component only) is to find $\boldsymbol{u}$ that maximizes the variance, expecting to maximally preserve distinction among data
- The resultant optimization problem is

$$\max_{\boldsymbol{u}} \ \boldsymbol{u}^T\boldsymbol{\Sigma}\boldsymbol{u} \quad \text{s.t.} \ \ \|\boldsymbol{u}\|_2^2 = 1$$

- It can be solved by forming the Lagrangian

$$\boldsymbol{u}^T\boldsymbol{\Sigma}\boldsymbol{u} + \lambda(1 - \boldsymbol{u}^T\boldsymbol{u})$$

- By setting the gradient w.r.t. $\boldsymbol{u}$ to zero, we have

$$2\boldsymbol{\Sigma}\boldsymbol{u} - 2\lambda\boldsymbol{u} = 0 \longrightarrow \boldsymbol{\Sigma}\boldsymbol{u} = \lambda\boldsymbol{u}$$

The desired direction $\boldsymbol{u}$ is an eigenvector of $\boldsymbol{\Sigma}$

$\boldsymbol{\Sigma}$ has d eigenvectors, which one?

- Recall that the variance of the projected dataset $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$ is $\boldsymbol{u}^T\boldsymbol{\Sigma}\,\boldsymbol{u}$
- By substituting $\boldsymbol{\Sigma}\boldsymbol{u} = \lambda\boldsymbol{u}$ into the above formula, the projected variance becomes

$$\boldsymbol{u}^T\boldsymbol{\Sigma}\,\boldsymbol{u} = \boldsymbol{u}^T\lambda\boldsymbol{u} = \lambda\boldsymbol{u}^T\boldsymbol{u} = \lambda \qquad (\|\boldsymbol{u}\|_2^2 = 1)$$

- To find a direction that maximizes the projected variance is to find the eigenvector $\boldsymbol{u}$ of $\boldsymbol{\Sigma}$ with the largest eigenvalue
- Generalized to multiple components case: let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ be the eigenvalues of $\boldsymbol{\Sigma}$, and $\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_d$ be the corresponding eigenvectors, and choose the top k eigenvectors as the principal components

Determine Value of k

- Wrapper approaches
  - Dimensionality reduction is usually an intermediate step for some downstream tasks, such as classification, regression, clustering
  - Use cross-validation based on the performance of the final task to tune the value of k
- Based on the percentage of variance preserved

$$p_{\text{var}} = \frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{d}\lambda_i} \times 100$$

  - All the λ_i's are nonnegative
  - Predefine a value for the percentage of variance to determine the value of k

Compute Eigenvalues and Eigenvectors

- How to compute eigenvalues and eigenvectors of $\bar{\boldsymbol{\Sigma}} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}$?
- In a general case, if a d-by-d square matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \mathbf{X}^T\mathbf{X}, \text{ where } \mathbf{X} \in \mathbb{R}^{N\times d}$$

then eigenvectors and eigenvalues of $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ can be computed by performing Singular Value Decomposition (SVD) on $\mathbf{X}$

Orthogonal Vectors

- Two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are said to be orthogonal if they are perpendicular to each other, i.e., the inner or dot product of two vectors is 0
  - $\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = 0$
- A set of vectors $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_d\}$ are mutually orthogonal if every pair of vectors are orthogonal
  - $\boldsymbol{v}_i \cdot \boldsymbol{v}_j = 0$, for any $i \neq j$

$$\boldsymbol{v}_1 = \begin{pmatrix}1\\0\\-1\end{pmatrix} \quad \boldsymbol{v}_2 = \begin{pmatrix}1\\\sqrt{2}\\1\end{pmatrix} \quad \boldsymbol{v}_3 = \begin{pmatrix}1\\-\sqrt{2}\\1\end{pmatrix}$$

$$\boldsymbol{v}_1 \cdot \boldsymbol{v}_2 = \boldsymbol{v}_1 \cdot \boldsymbol{v}_3 = \boldsymbol{v}_2 \cdot \boldsymbol{v}_3 = 0$$

- A set of vectors $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_d\}$ are mutually orthonormal if every pair of vectors are orthogonal, and the L_2 norm of each vector is 1
  - $\boldsymbol{v}_i \cdot \boldsymbol{v}_j = 0$, for any $i \neq j$
  - $\|\boldsymbol{v}_i\|_2 = \sqrt{\boldsymbol{v}_i \cdot \boldsymbol{v}_i} = 1$
- A set of orthogonal vectors $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_d\}$ can be normalized to orthonormal via $\left\{\frac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|_2}, \dots, \frac{\boldsymbol{v}_d}{\|\boldsymbol{v}_d\|_2}\right\}$

$$\boldsymbol{v}_1 = \begin{pmatrix}1\\0\\-1\end{pmatrix} \ \|\boldsymbol{v}_1\|_2 = \sqrt{2} \qquad \boldsymbol{v}_2 = \begin{pmatrix}\frac{1}{\sqrt{2}}\\1\\\frac{1}{\sqrt{2}}\end{pmatrix} \ \|\boldsymbol{v}_2\|_2 = 2 \qquad \boldsymbol{v}_3 = \begin{pmatrix}\frac{1}{\sqrt{2}}\\-1\\\frac{1}{\sqrt{2}}\end{pmatrix} \ \|\boldsymbol{v}_3\|_2 = 2$$

$$\boldsymbol{v}_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\-1\end{pmatrix} \ \|\boldsymbol{v}_1\|_2 = 1 \qquad \boldsymbol{v}_2 = \frac{1}{2}\begin{pmatrix}1\\\sqrt{2}\\1\end{pmatrix} \ \|\boldsymbol{v}_2\|_2 = 1 \qquad \boldsymbol{v}_3 = \frac{1}{2}\begin{pmatrix}1\\-\sqrt{2}\\1\end{pmatrix} \ \|\boldsymbol{v}_3\|_2 = 1$$

- Given a matrix $\mathbf{V} = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_d)$, where $\boldsymbol{v}_i$ is an N-dimensional column vector, and $N \geq d$
- If the columns of $\mathbf{V}$ are mutually orthonormal, then we have

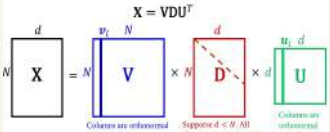
$$\mathbf{V}^T\mathbf{V} = \mathbf{I}_d$$

$$\mathbf{I}_d = \begin{pmatrix}1 & 0 & \cdots & 0\\0 & 1 & \cdots & 0\\\cdots & \cdots & \cdots & \cdots\\0 & 0 & \cdots & 1\end{pmatrix} \Big\} d \quad (d \text{ columns})$$

Singular Value Decomposition (SVD)

- The SVD of $\mathbf{X}$ (N-by-d) has the following form

$$\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{U}^T$$



Obtain Eigenvectors via SVD

- Perform SVD on $\mathbf{X}$ to get $\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{U}^T$
- Then $\mathbf{A}$ can be rewritten as

$$\mathbf{A} = \mathbf{X}^T\mathbf{X} = (\mathbf{V}\mathbf{D}\mathbf{U}^T)^T\mathbf{V}\mathbf{D}\mathbf{U}^T = \mathbf{U}\mathbf{D}^T\underbrace{\mathbf{V}^T\mathbf{V}}_{\mathbf{V}^T\mathbf{V}=\mathbf{I}_N}\mathbf{D}\mathbf{U}^T$$

$$\mathbf{A} = \mathbf{U}\mathbf{D}^T\mathbf{D}\mathbf{U}^T$$

Denote $\bar{\mathbf{D}} = \mathbf{D}^T\mathbf{D}$

$$= \mathbf{U}\bar{\mathbf{D}}\mathbf{U}^T$$

$\bar{\mathbf{D}}$ (d-by-d): d-by-d diagonal matrix with diagonal elements $\lambda_1^2 \geq \lambda_2^2 \geq \cdots \geq \lambda_d^2 \geq 0$

Suppose $d < N$. All elements are zero expect for the first d-by-d submatrix diagonal being $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$

$$\mathbf{A} = \mathbf{U}\bar{\mathbf{D}}\mathbf{U}^T \qquad \bar{\mathbf{D}}: \text{Diagonal matrix with diagonal elements } \lambda_1^2 \geq \lambda_2^2 \geq \cdots \geq \lambda_d^2 \geq 0$$

$$\mathbf{U}^T\mathbf{U} = \mathbf{I}_d$$

$$\mathbf{A}\mathbf{U} = \mathbf{U}\bar{\mathbf{D}}\underbrace{\mathbf{U}^T\mathbf{U}}_{\mathbf{I}_d} = \mathbf{U}\bar{\mathbf{D}}$$

$$(\mathbf{A}\times\boldsymbol{u}_1, \mathbf{A}\times\boldsymbol{u}_2, \dots, \mathbf{A}\times\boldsymbol{u}_d) = [\lambda_1^2\times\boldsymbol{u}_1, \lambda_2^2\times\boldsymbol{u}_2, \dots, \lambda_d^2\times\boldsymbol{u}_d]$$

$$\mathbf{A}\boldsymbol{u}_i = \lambda_i^2\boldsymbol{u}_i, i = 1, \dots, d$$

Each column $\boldsymbol{u}_i$ of $\mathbf{U}$ is an eigenvector of $\mathbf{A}$ with the eigenvalue λ_i^2

Reference (Optional)

- For feature subset selection:
  - An Introduction to Variable and Feature Selection, Isabelle Guyon, Andre Elisseeff, in JMLR 2003
- For dimensionality reduction:
  - Dimensionality Reduction: A Comparative Review, L.J.P. van der Maaten and E. O. Postma and H. J. van den Herik, Technical Report, 2008
  - https://lvdmaaten.github.io/drtoolbox/

Appendix (optional)

Derivation of PCA

- The variance preservation view
  - The first k components display as much as possible of the variation among data instances
- The minimum reconstruction view
  - The first k components convey maximum useful information of original data instances

Appendix (optional)

Minimum Reconstruction Error

- Given any orthonormal basis $\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_d$, a data point $\boldsymbol{x}_i$ (has been centered) can be written as

$$\boldsymbol{x}_i = \sum_{j=1}^{d}\underbrace{\boldsymbol{v}_j^T\boldsymbol{x}_i}_{\alpha_{ij}}\boldsymbol{v}_j \qquad \boldsymbol{x}_i \to \boldsymbol{x}_i' = \sum_{j=1}^{k}\alpha_{ij}\boldsymbol{v}_j$$

- Consider the k-term approximation of $\boldsymbol{x}_i$:

$$\hat{\boldsymbol{x}}_i = \sum_{j=1}^{k}\alpha_{ij}\boldsymbol{v}_j$$

- The error of the approximate over all data points is

$$E = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 = \frac{1}{N}\sum_{i=1}^{N}\left\|\sum_{j=1}^{d}\alpha_{ij}\boldsymbol{v}_j - \sum_{j=1}^{k}\alpha_{ij}\boldsymbol{v}_j\right\|_2^2 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=k+1}^{d}\alpha_{ij}^2$$

Appendix (optional)

Minimum Reconstruction Error (cont.)

- The error of the approximate over all data points

$$E = \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=k+1}^{d}\alpha_{ij}^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{j=k+1}^{d}\boldsymbol{v}_j^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{v}_j = \sum_{j=k+1}^{d}\boldsymbol{v}_j^T\boldsymbol{\Sigma}\boldsymbol{v}_j$$

- Suppose $k = d - 1$, i.e., we aim to remove a single dimension, then resultant optimization problem is

$$\min_{\boldsymbol{v}_d} \ \boldsymbol{v}_d^T\boldsymbol{\Sigma}\boldsymbol{v}_d \quad \text{s.t.} \ \ \|\boldsymbol{v}_d\|_2^2 = 1$$

Appendix (optional)

Minimum Reconstruction Error (cont.)

- By setting the gradient of the Lagrangian w.r.t. $\boldsymbol{v}$ to zero, we have

$$\frac{\partial\boldsymbol{v}_d}{\partial}: 2\boldsymbol{\Sigma}\boldsymbol{v}_d - 2\lambda\boldsymbol{v}_d = 0 \longrightarrow \boxed{\boldsymbol{\Sigma}\boldsymbol{v}_d = \lambda\boldsymbol{v}_d}$$ The desired direction $\boldsymbol{v}_d$ is an eigenvector of $\boldsymbol{\Sigma}$

$\boldsymbol{\Sigma}$ has d eigenvectors, which one?

- Our goal is to minimize the reconstruction error $\boldsymbol{v}_d^T\boldsymbol{\Sigma}\,\boldsymbol{v}_d$

$$\boldsymbol{v}_d^T\boldsymbol{\Sigma}\,\boldsymbol{v}_d = \boldsymbol{v}_d^T\lambda\boldsymbol{v}_d = \lambda\boldsymbol{v}_d^T\boldsymbol{v}_d = \lambda$$

- Therefore, $\boldsymbol{v}_d$ should be the eigenvector $\boldsymbol{u}_d$ of $\boldsymbol{\Sigma}$ with the smallest eigenvalue because $\boldsymbol{u}_d^T\boldsymbol{\Sigma}\boldsymbol{u}_d = \lambda_d$
- Similarly, the other dimensions to remove are subsequently the eigenvectors corresponding to the least eigenvalues

$\frac{d}{dx} e^u = u' e^u$

$x^2 = 2x$