University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

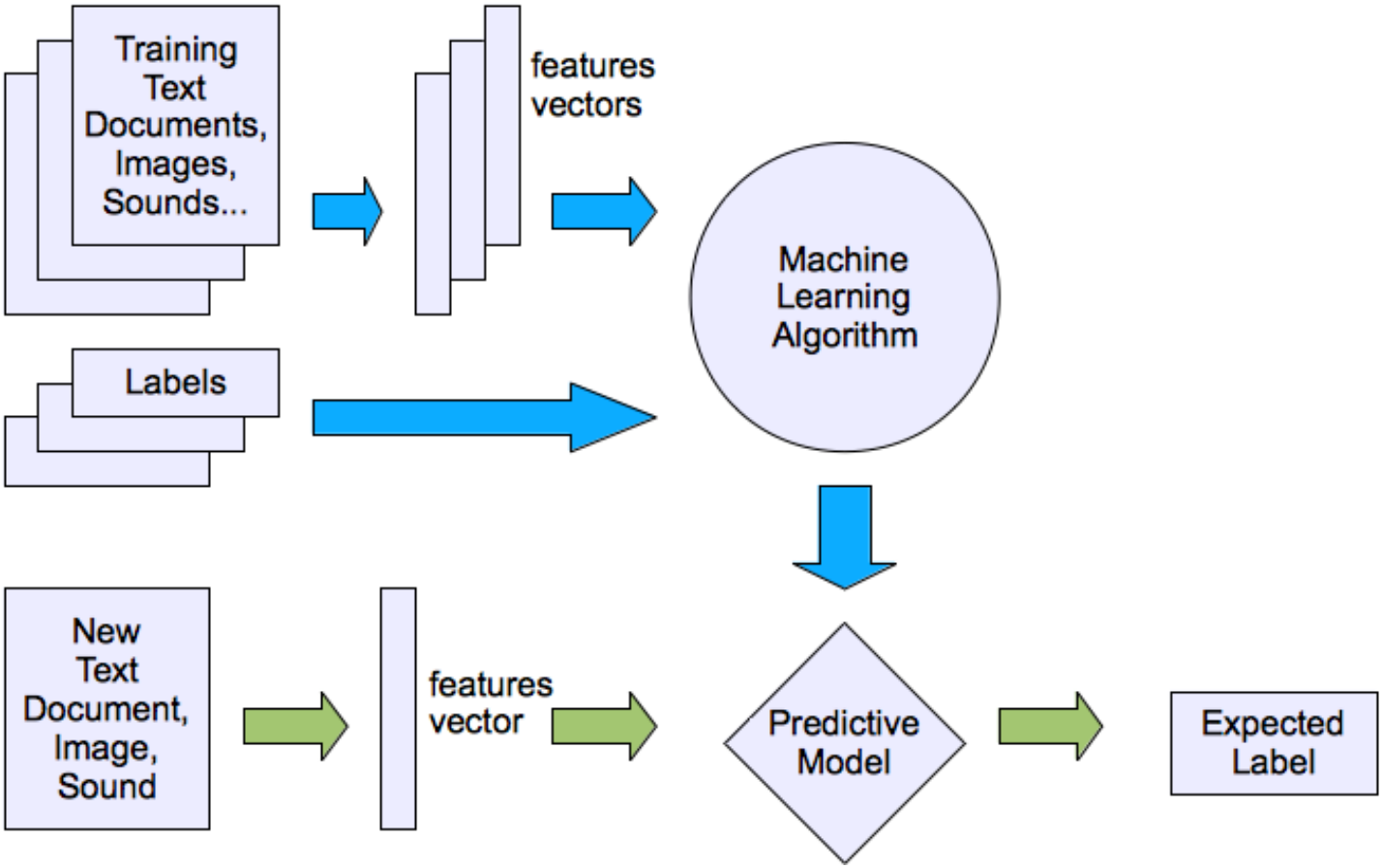
Topic 1 – Introduction

Ying Weng

2024 Autumn

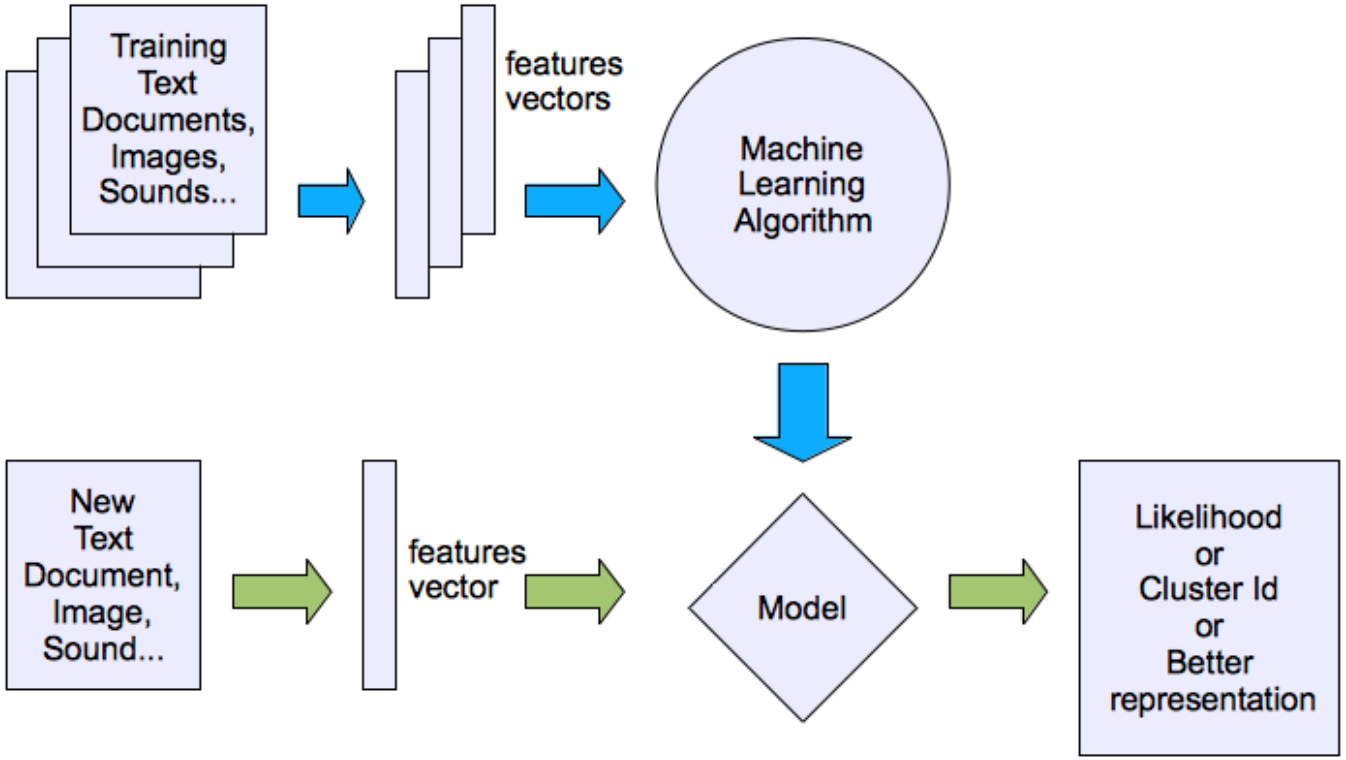
Types of Learning

- Supervised learning



Types of Learning

- Unsupervised learning



Types of Learning

- **Supervised (inductive) learning**
  - Training data includes desired outputs
- **Unsupervised learning**
  - Training data does not include desired outputs
  - This is the new frontier of machine learning because most big datasets do not come with labels
- **Semi-supervised learning**
  - Training data includes a few desired outputs

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055
Machine Learning

Topic 2 – Design a Learning System

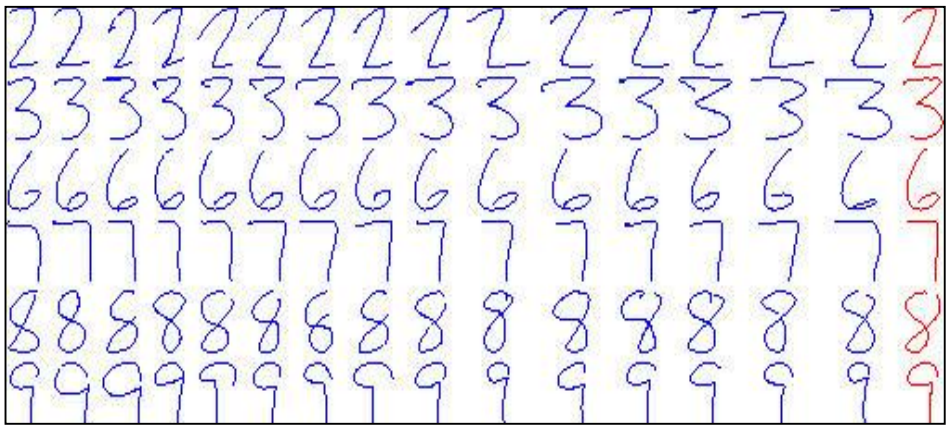
Ying Weng
2024 Autumn

Design a Learning System

Step 1: Collect Training Examples (Experience).

- Without examples, our system will not learn
  - so-called learning from examples

Identify or Class or Label

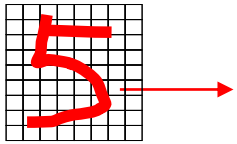


2
3
6
7
8
9

Design a Learning System

Step 2: Representing Experience

- Choose a representation scheme for the experience / examples



X = (1,1,0,1,1,1,1,1,1,1,0,0,0,0,1,1, …., 1); 64-d Vector

The sensor input represented by an n-d vector,

called the **feature vector**, $\mathbf{X} = (x_1, x_2, x_3, \ldots, x_n)$

Design a Learning System

Step 2: Representing Experience

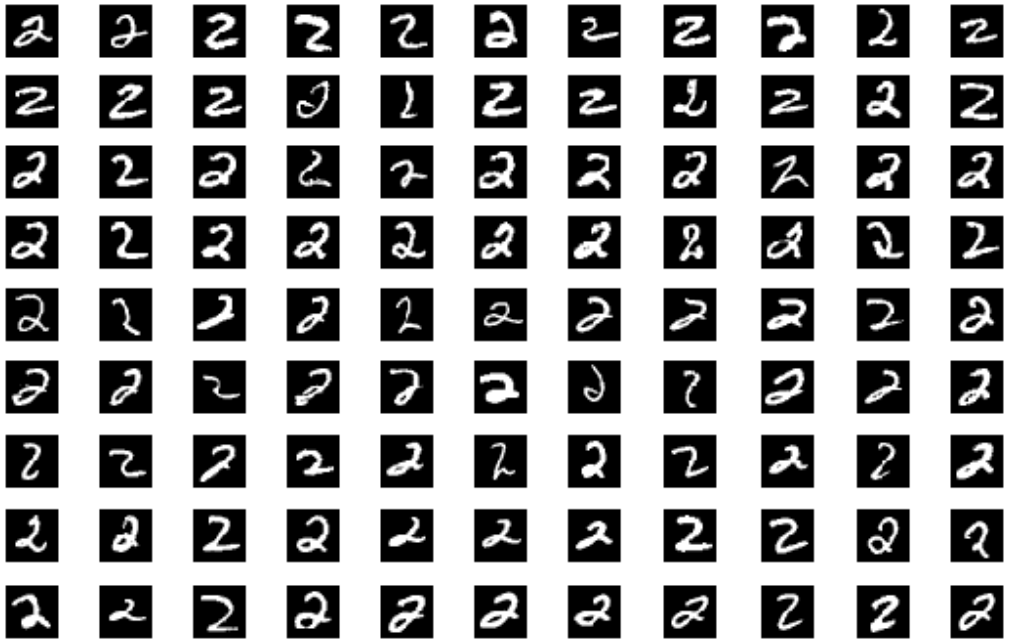
- THE MNIST DATABASE
  http://yann.lecun.com/exdb/mnist/

- The original black and white (bi-level) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

Design a Learning System

Step 2: Representing Experience

- THE MNIST DATABASE
  http://yann.lecun.com/exdb/mnist/



The feature vector of input data is a 784 dimensional vector

Design a Learning System

Step 2: Representing Experience

- Choose a representation scheme for the experience/examples

  - The sensor input represented by an n-d vector, called the feature vector, **X** = (x1, x2, x3, …, xn)

  - To represent the experience, we need to know what **X** is.

  - So we need a corresponding vector **D**, which will record our knowledge (experience) about **X**.

  - The experience **E** is a pair of vectors **E** = (**X**, **D**).

Design a Learning System

Step 2: Representing Experience

- Choose a representation scheme for the experience/examples.
  - The experience **E** is a pair of vectors **E** = (**X**, **D**).
- So, what would **D** be like? There are many possibilities.

Design a Learning System

Step 2: Representing Experience

- So, what would **D** be like? There are many possibilities.
- Assuming our system is to recognise 10 digits only, then D can be a 10-d binary vector; each correspond to one of the digits.

D = (d0, d1, d2, d3, d4, d5, d6, d7, d8, d9)

e.g,

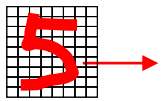
if X is digit 5, then d5=1; all others =0

Design a Learning System

Step 2: Representing Experience

- So, what would **D** be like? There are many possibilities.
- Assuming our system is to recognise 10 digits only, then D can be a 10-d binary vector; each correspond to one of the digits.

D = (d0, d1, d2, d3, d4, d5, d6, d7, d8, d9)



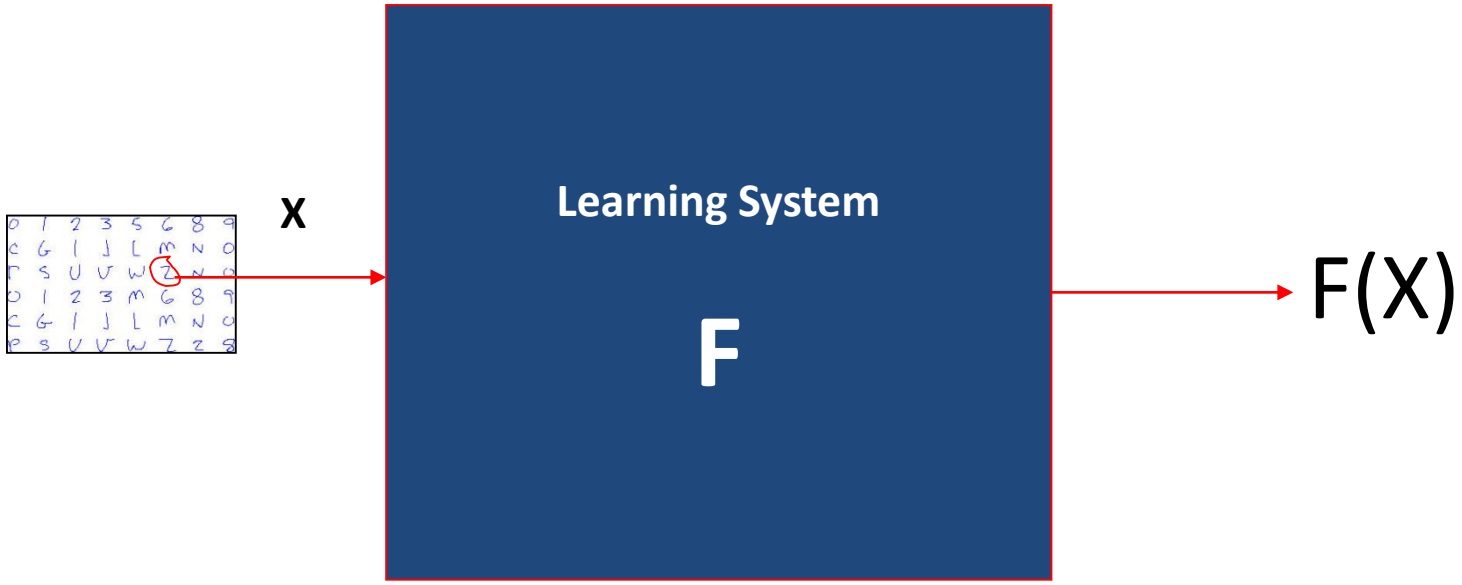
X = (1,1,0,1,1,1,1,1,1,1,0,0,0,0,1,1, …., 1); 64-d Vector

D= (0,0,0,0,0,1,0,0,0,0)

Design a Learning System

Step 3: Choose a Representation for the Black Box

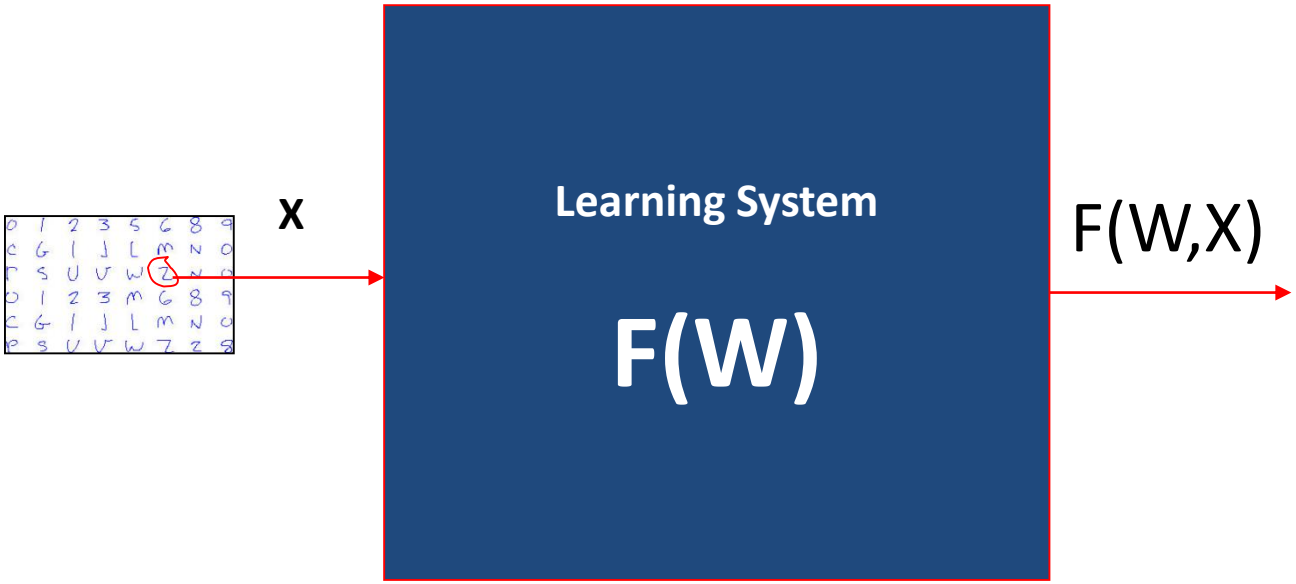
- We need to choose a function F to approximate the block box. For a given X, the value of F will give the classification of X. There are considerable flexibilities in choosing F.



Design a Learning System

Step 3: Choose a Representation for the Black Box

- F will be a function of some adjustable parameters, or weights, W = (w1, w2, w3, …w_N), which the learning algorithm can modify or learn



Design a Learning System

Step 4: Learning/Adjusting the Weights

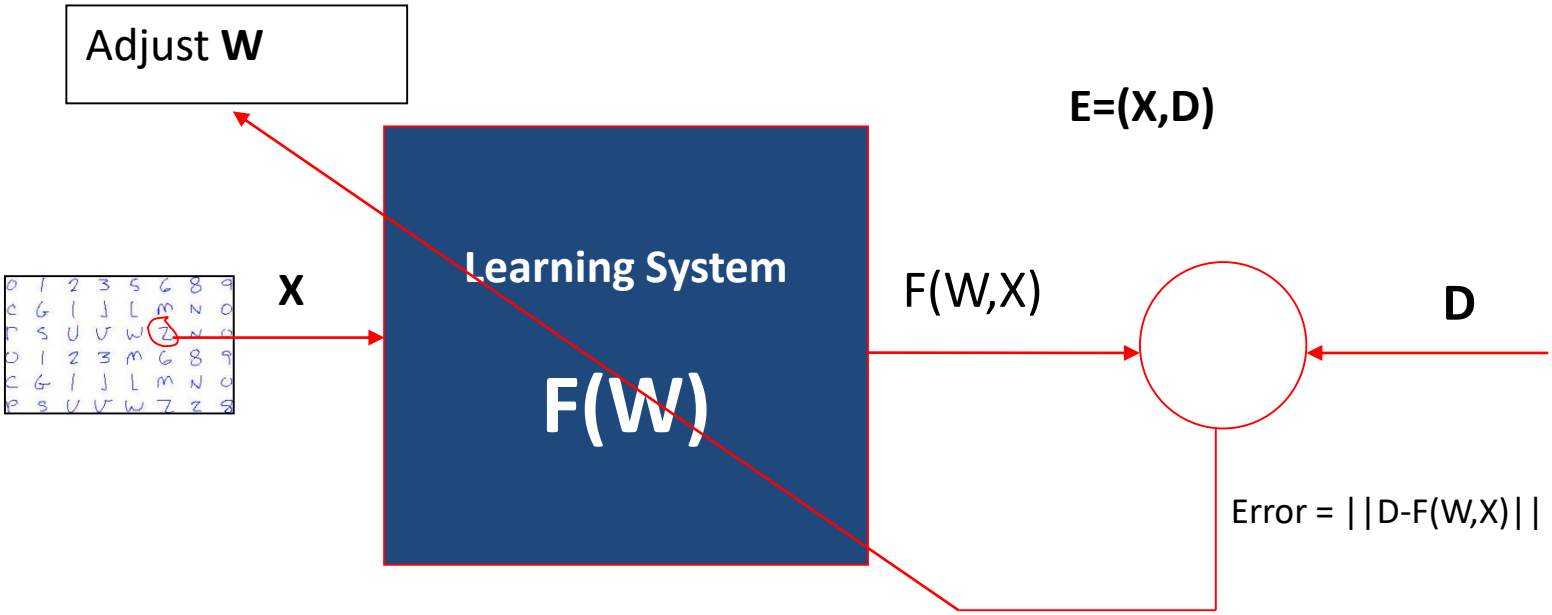
- We need a learning algorithm to adjust the weights such that the experience/prior knowledge from the training data can be learned into the system:

E=(X,D)

F(W,X) = D

Design a Learning System

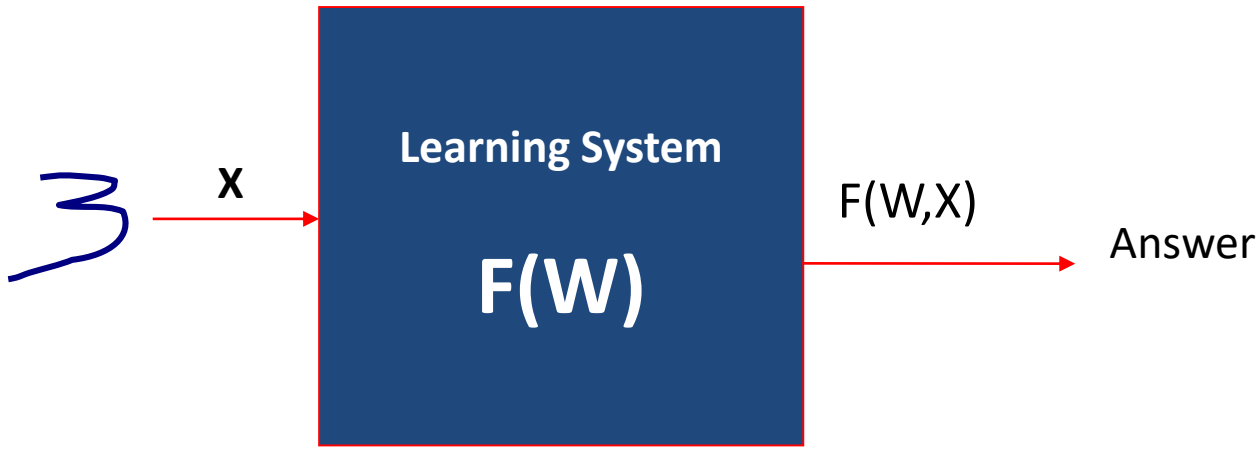
Step 4: Learning/Adjusting the Weights



Design a Learning System

Step 5: Use/Test the System

– Once learning is completed, all parameters are fixed. An unknown input **X** is presented to the system, the system computes its answer according to F(**W**,**X**)



University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

Topic 3 – Data Collection

Ying Weng

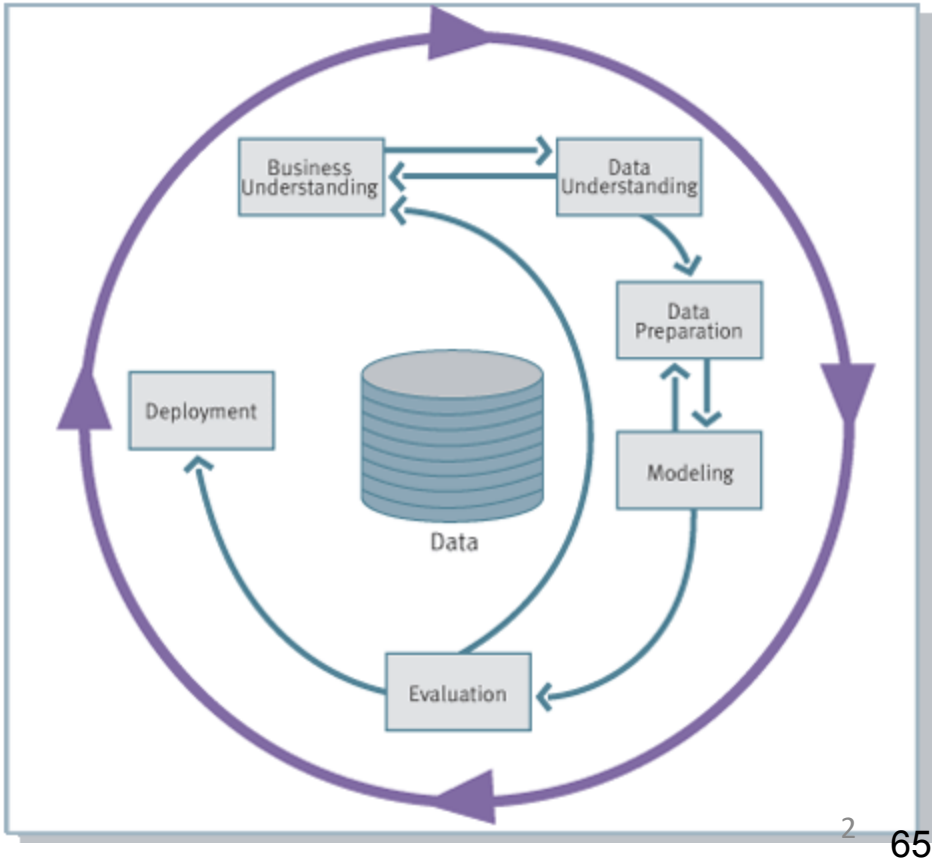
2024 Autumn

Data Mining Process Model

CRoss Industry Standard Process for Data Mining (**CRISP-DM**)

Industry Standard

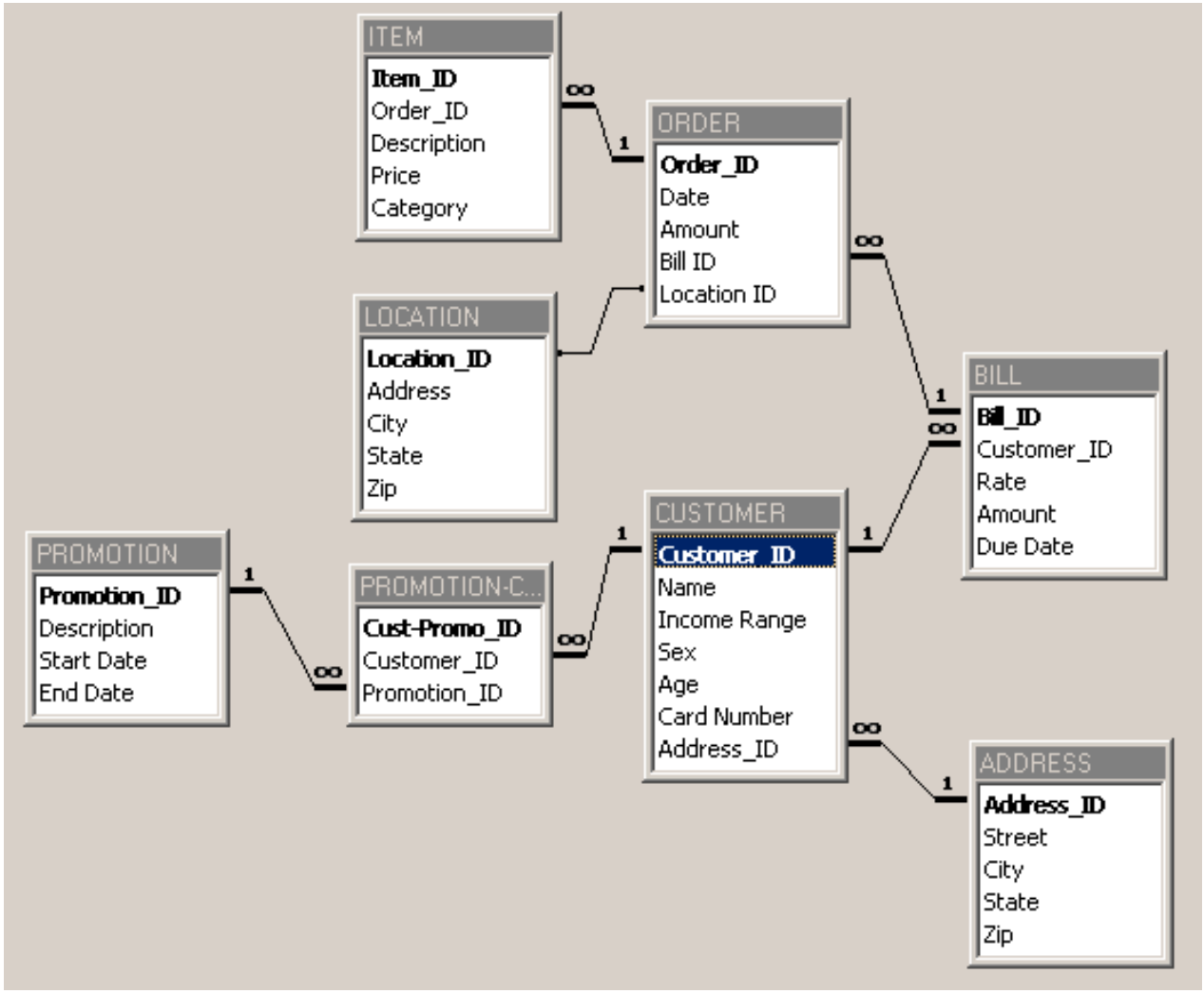
The most popular data mining methodology model according to KDNuggets.com



Step 1: Business Understanding

- Define the problem
- Choose a machine learning model(s)
- Estimate project cost
- Estimate project completion time
- Address legal issues
- Develop a maintenance plan

Step 2: Data Understanding



Step 3: Data Preprocessing

- Noisy data
  - Locate duplicate records
  - Locate incorrect attribute values
  - Smooth data
- Missing data
  - Discard records with missing values
  - Replace missing real-valued items with the class mean
  - Replace missing values with values found within *highly similar* instances
- Data transformation
  - Data normalization
  - Data type conversion
  - Attribute and instance selection

Step 4: Modeling

- Choose training and test data
- Designate a set of input attributes
- If learning is supervised, choose one or more output attributes
- Select learning parameter values
- Train the model

Step 5: Evaluation

- Statistical analysis
- Heuristic analysis
- Experimental analysis
- Human analysis

Measures of Effectiveness of the Model

- Accuracy

  Percentage of total predictions that were correct
- Return on investment

  Cost-benefit ratios
- Explanation

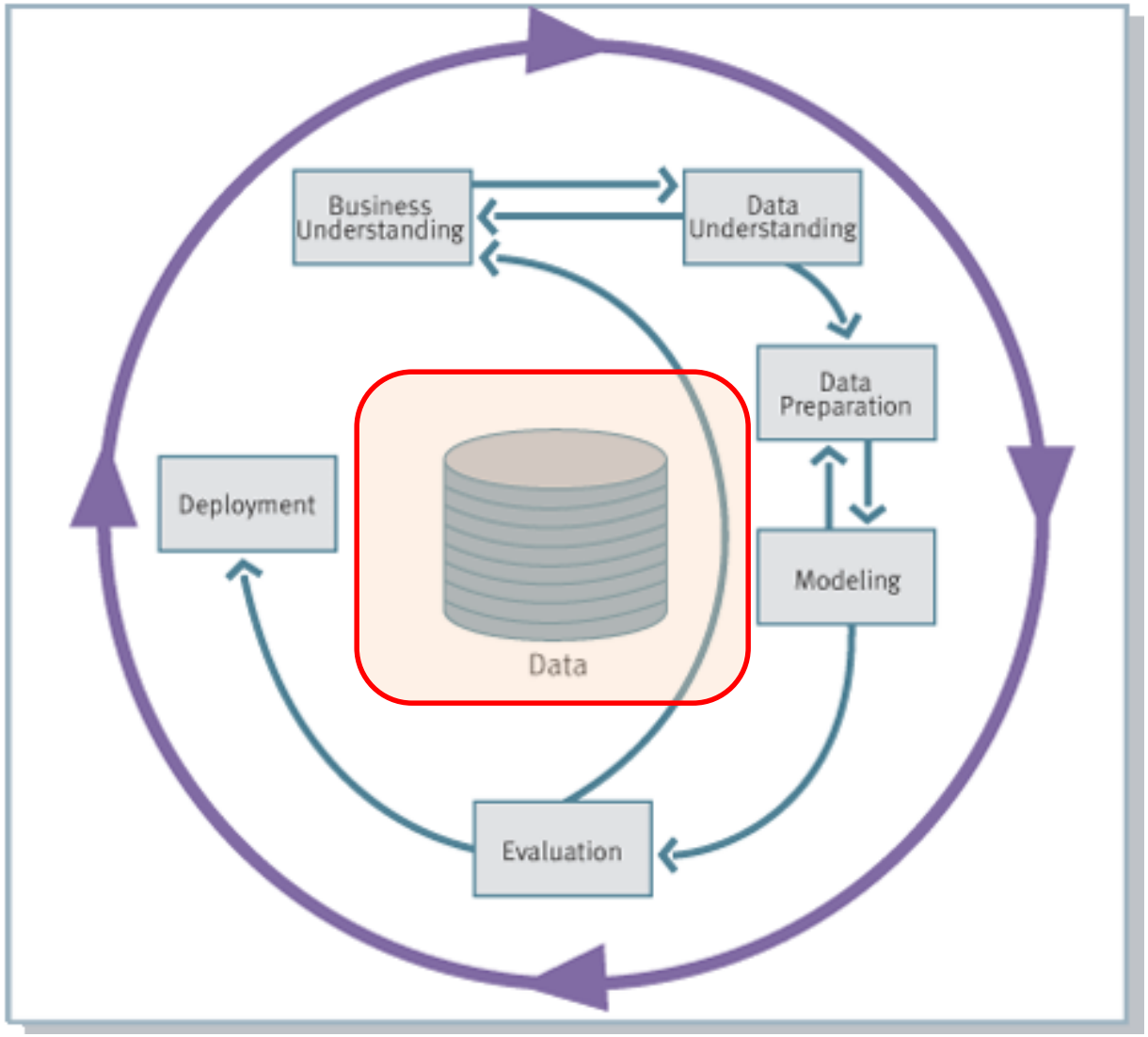
  Able to justify intuition
- Validation

  Automated checking of correctness, indexes

Step 6: Deployment

- Apply the model to real world usage
- Apps, API, etc.
- Regularly update the model with new data
- …

You Need Collect Data before Learning Starts!



What is Data?

Collection of data objects and their associated attributes

An **attribute** is a property or characteristic of an object

- Examples: eye color of a person, temperature, etc.
- Attribute is also known as variable, field, characteristic, or feature

A **collection of attributes** describe an object

- Object is also known as record, point, case, sample, entity, or instance

Attributes

Objects

Tid	Refund	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

Types of Attributes

There are different types of attributes

NO NEED TO EXPLAIN

- Nominal
  - Examples: ID numbers, eye color, zip codes
- Ordinal
  - Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}
- Interval
  - Examples: calendar dates, temperatures in Celsius or Fahrenheit.
- Ratio
  - Examples: temperature in Kelvin, length, time, counts

Attribute Type	Description	Examples	Operations
Nominal	The values of a nominal attribute are just different names, i.e., nominal attributes provide only enough information to distinguish one object from another.	zip codes, employee ID numbers, eye color, sex: {*male*, *female*}	mode, entropy, contingency correlation, χ^2 test
Ordinal	The values of an ordinal attribute provide enough information to order objects.	hardness of minerals, {*good*, *better*, *best*}, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists.	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, *t* and *F* tests
Ratio	For ratio variables, both differences and ratios are meaningful.	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

Topic 5 – Machine Learning Theory and Practice

Ying Weng

2024 Autumn

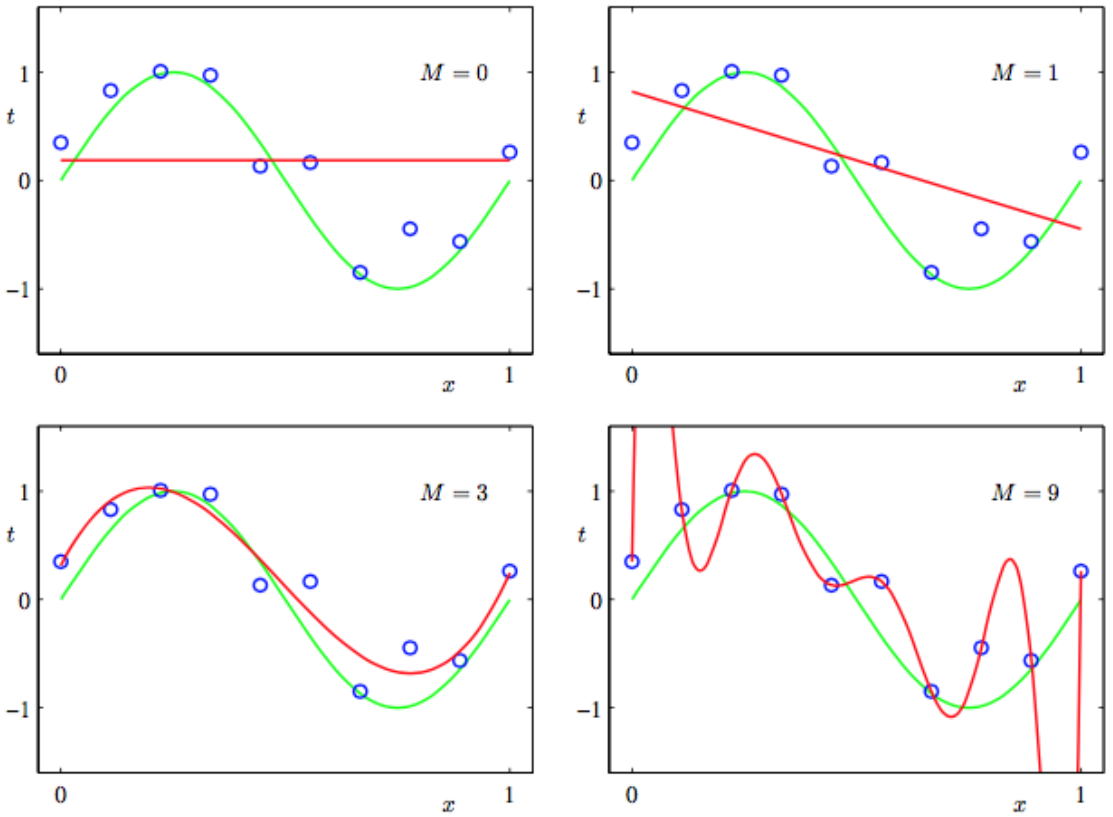
Classification VS Regression

Classification
predict a label (discrete value)

Regression
predict a response (continuous value)

Model Selection

Choosing order M of the polynomial



Model Selection

- **Goal**: achieve good *generalization* by making accurate predictions for new data.

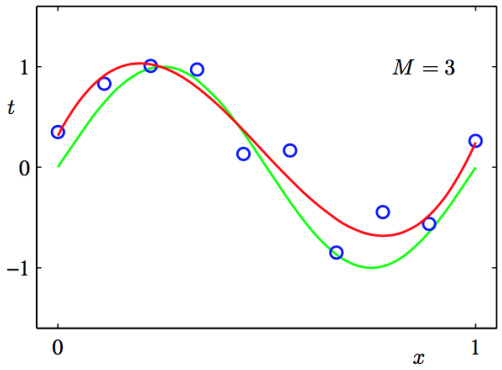
- We use Root-mean-square (RMS) error on data

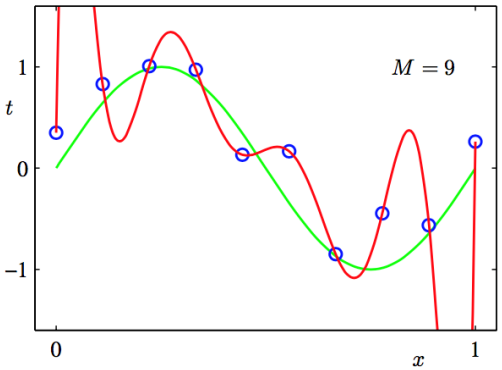
$$E_{RMS} = \sqrt{2E(w^*)/N}$$

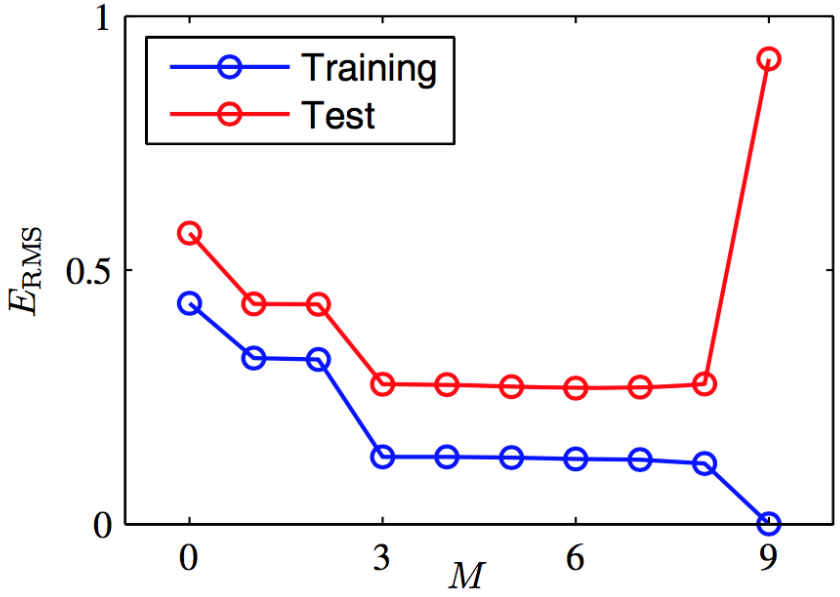
- Where N allows us to compare different sizes of data set, and w^* is the solution of minimizing $E(w)$ (*hypothesis*).

- It measures how well the model w^* doing in predicting the values of *t* for new data observations of *x*.

Overfitting







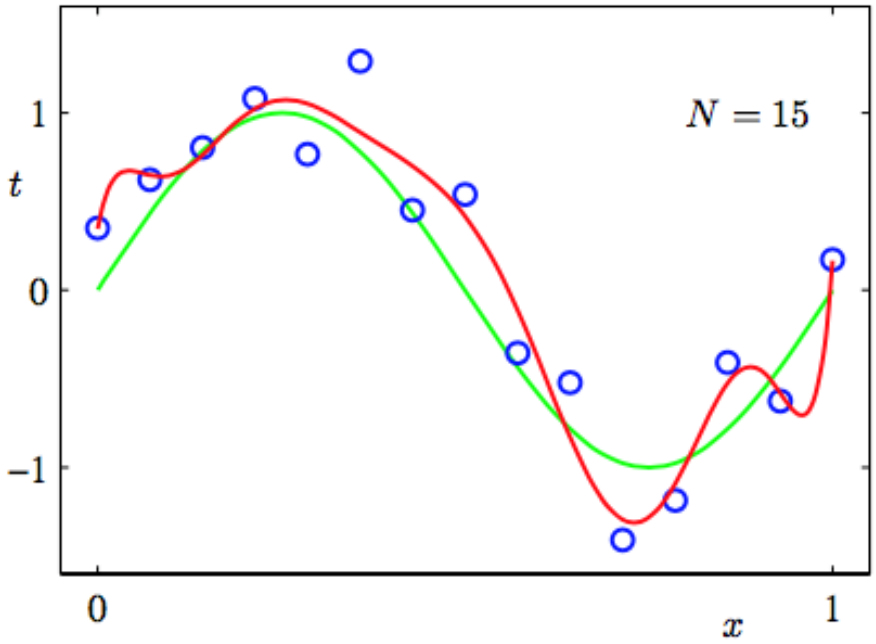
	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

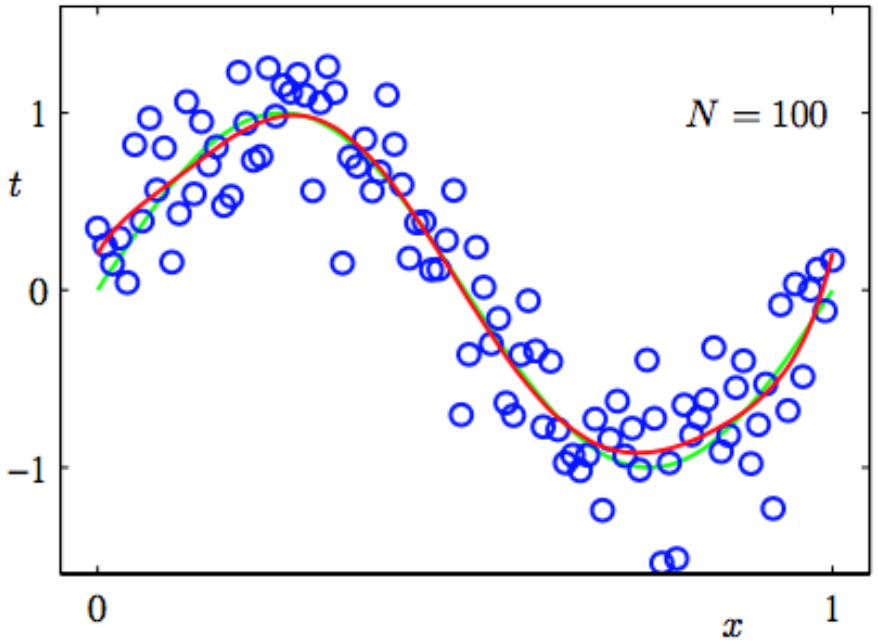
Overfitting

Overfitting can occur when:

- Learning is performed for too long (e.g. in Neural Networks).
- The examples in the training set are not representative of all possible situations (is usually the case!).
- Model parameters are adjusted to uninformative features in the training set that have no causal relation to the true underlying target function!

Overfitting

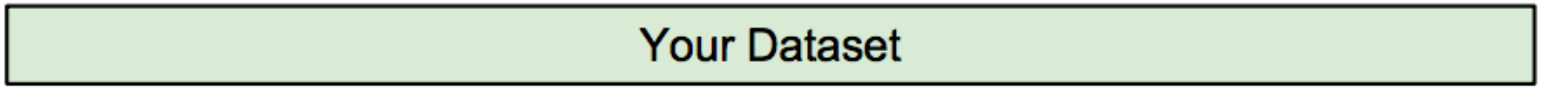




Increasing the size of the data set reduces the overfitting problem.

Cross Validation

- **Idea #1**: Choose hyperparameters that work best on the data

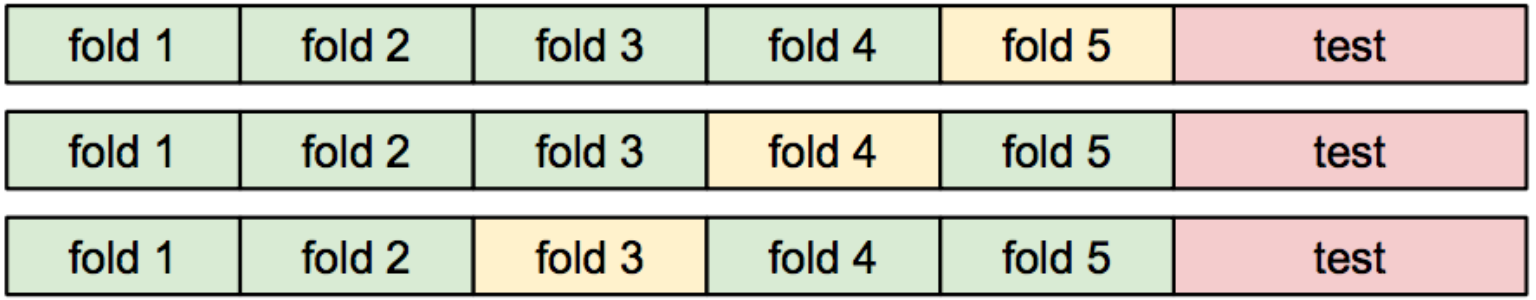


- **Idea #2**: Split data into **train** and **test**, choose hyperparameters that work best on test data

- **Idea #3**: Split data into **train**, **val**, and **test**; choose hyperparameters on **val** and evaluate on **test**

Cross Validation

- **Idea #4**: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results



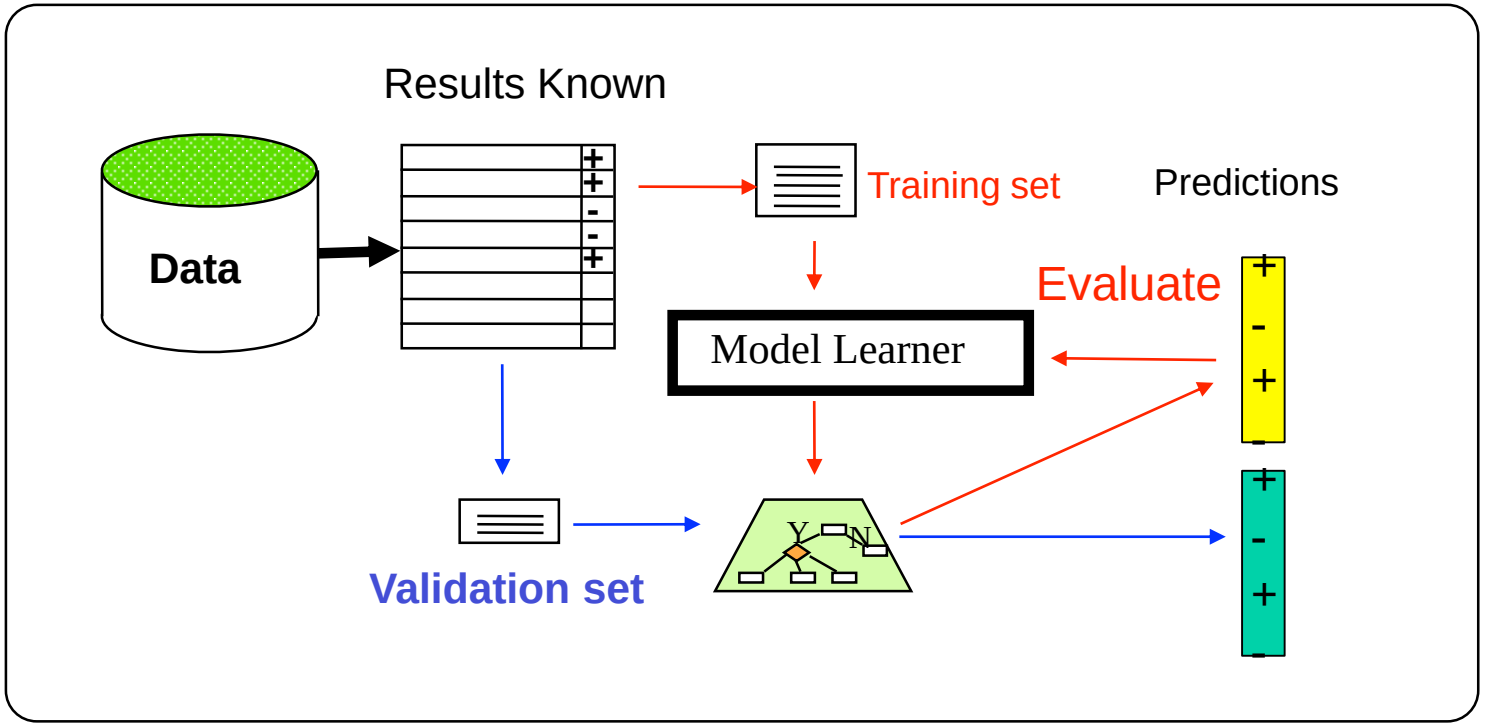
Cross Validation

- **Cross Validation** is often used to counter overfitting.

- Partition the dataset into S groups, with $(S\text{-}2)$ training sets, a validation set and a testing set.
  - The training set is used to determine the coefficients $\boldsymbol{w}$
  - The validation set is used to optimize the model complexity (hyperparameters, either M or λ in the previous example)
  - The testing set is used to evaluate the final selected mode

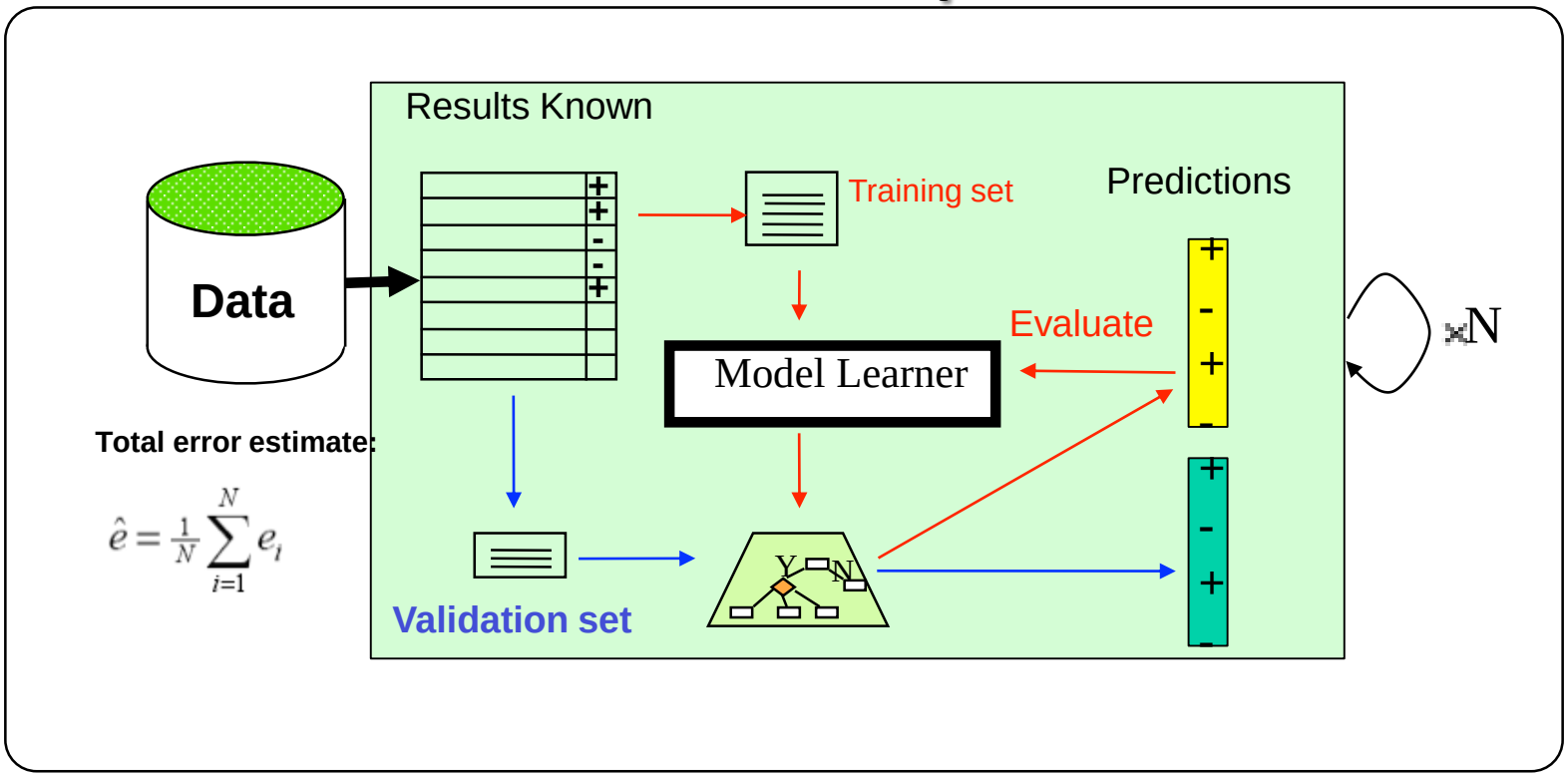
- The procedure is then repeated for all S possible choices, the performance scores from the S runs are then averaged.

Evaluation procedure



- For large datasets, a single split is usually sufficient.
- For smaller datasets, rely on cross validation

Cross validation procedure



- Split the data into training, validation and test sets in a repeated fashion.
- Estimate the total error as the average of each fold error.

Classification Measures - Error Rate

- Common performance measure for classification problems
  - Success: instance's class is predicted correctly (True Positives (**TP**) / Negatives (**TN**))
  - Error: instance's class is predicted incorrectly (False Positives (**FP**) /Negatives (**FN**))
  - False positives - **Type I error**. False Negative - **Type II error**

- Classification **error rate**: proportion of instances misclassified over the whole set of instances

  Error Rate = (**FP**+**FN**)/(**P**+**N**)

  a.k.a, **accuracy**

  Accuracy = 1 - **Error Rate**
- Classification Error Rate on the Training Set can be too optimistic!

Unbalanced data

- Balanced set: (roughly) equal number of positive / negative examples:

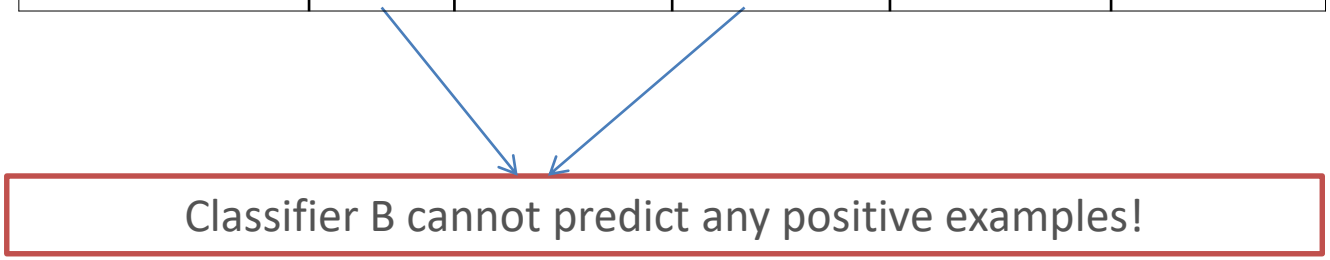
Classifier	TP	TN	FP	FN	Recall Rate
A	25	25	25	25	50%
B	37	37	13	13	74%

Recall = TP/(TP+FN)

Unbalanced data

- Unbalanced set: unequal number of positive / negative examples

Classifier	TP	TN	FP	FN	Recall Rate
A	25	75	75	25	50%
B	0	150	0	50	0%

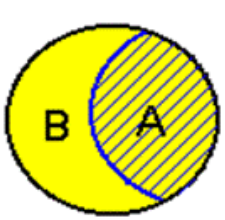


Recall = TP/(TP+FN)

Classification Measures

- Error Rate (Accuracy)
- Precision/Recall
- F-measure
- ROC curve
- Confusion matrix
- …

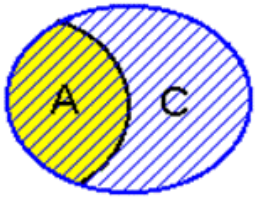
Recall/Precision



B: Number of positive examples not retrieved

A: Number of positive examples retrieved

$$\text{RECALL: } \frac{A}{A+B} \times 100\%$$



C: Number of negative examples retrieved

A: Number of positive examples retrieved

$$\text{PRECISION: } \frac{A}{A+C} \times 100\%$$

More insight over a classifier's behavior

For the positive class:

Classifier A: Recall = 50%, Precision = 25%

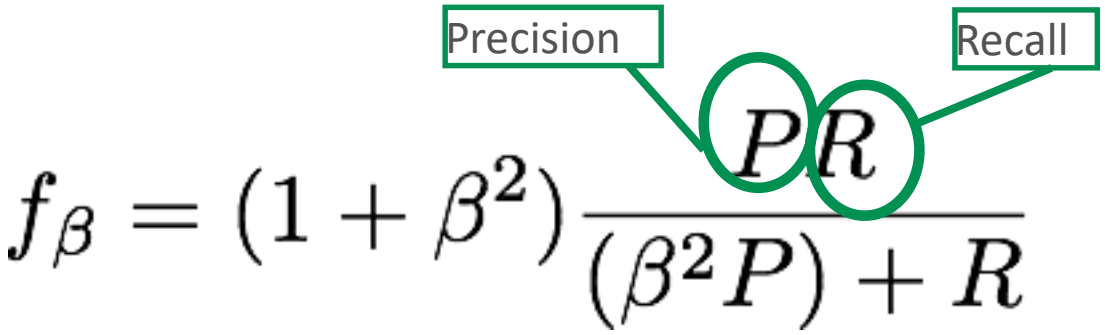
Classifier B: Recall = 0%, Precision = 0%

Classifier B is useless!

F-measure

- Comparing different approaches is difficult when using multiple evaluation measures (e.g. Recall and Precision)

- F-measure combines recall and precision into a single measure:

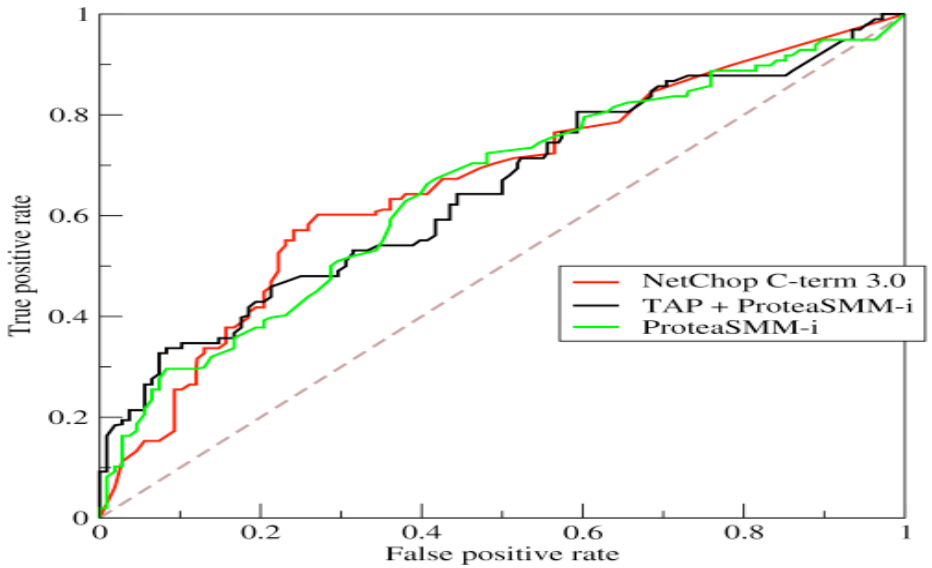


β is a non-negative real values

- We often use f1 measure.

ROC curves

- Receiver Operator Characteristic (ROC) curves plot TP vs FP rates



- Can be achieved by e.g. varying decision threshold of a classifier
- Area under the curve is often used as measure of goodness

Confusion matrix

- A visualization tool used to present the results attained by a learner.
- Easy to see if the system is commonly mislabeling one class as another.

Predicted True	**A**	**B**	**C**
A	5	3	0
B	2	3	1
C	0	2	11

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

Topic 6 – Instance Based Learning

Ying Weng

2024 Autumn

Instance Based Learning

- Directly compare new problem instances with instances seen in training
- No explicit modeling of the training data
- Complexity grows with the training data
- Classical instance based learning technique
  - **K Nearest Neighbour**

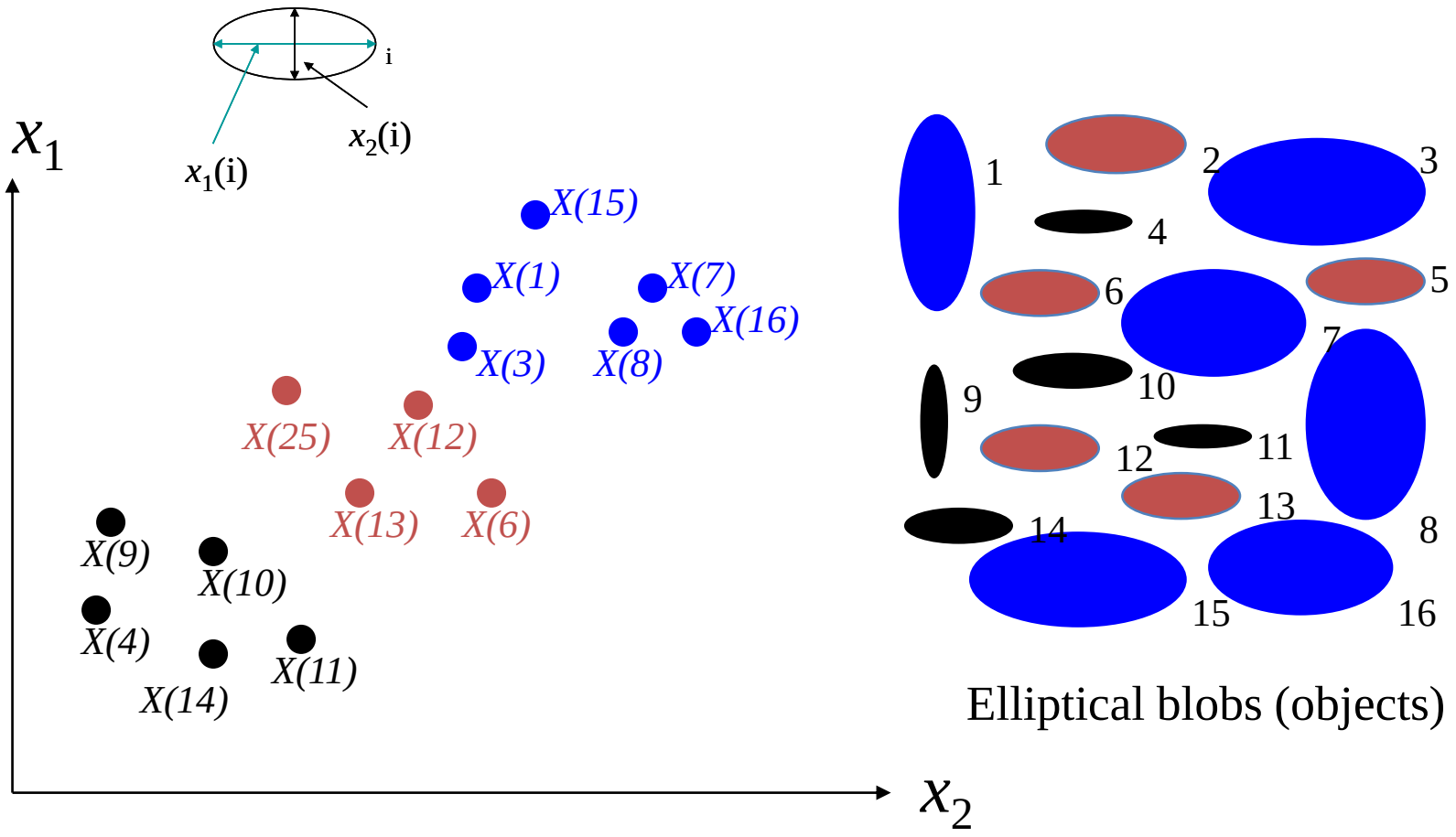
K-Nearest Neighbour Model

Example

Classify whether a customer will respond to a survey question using a 3-Nearest Neighbour classifier.

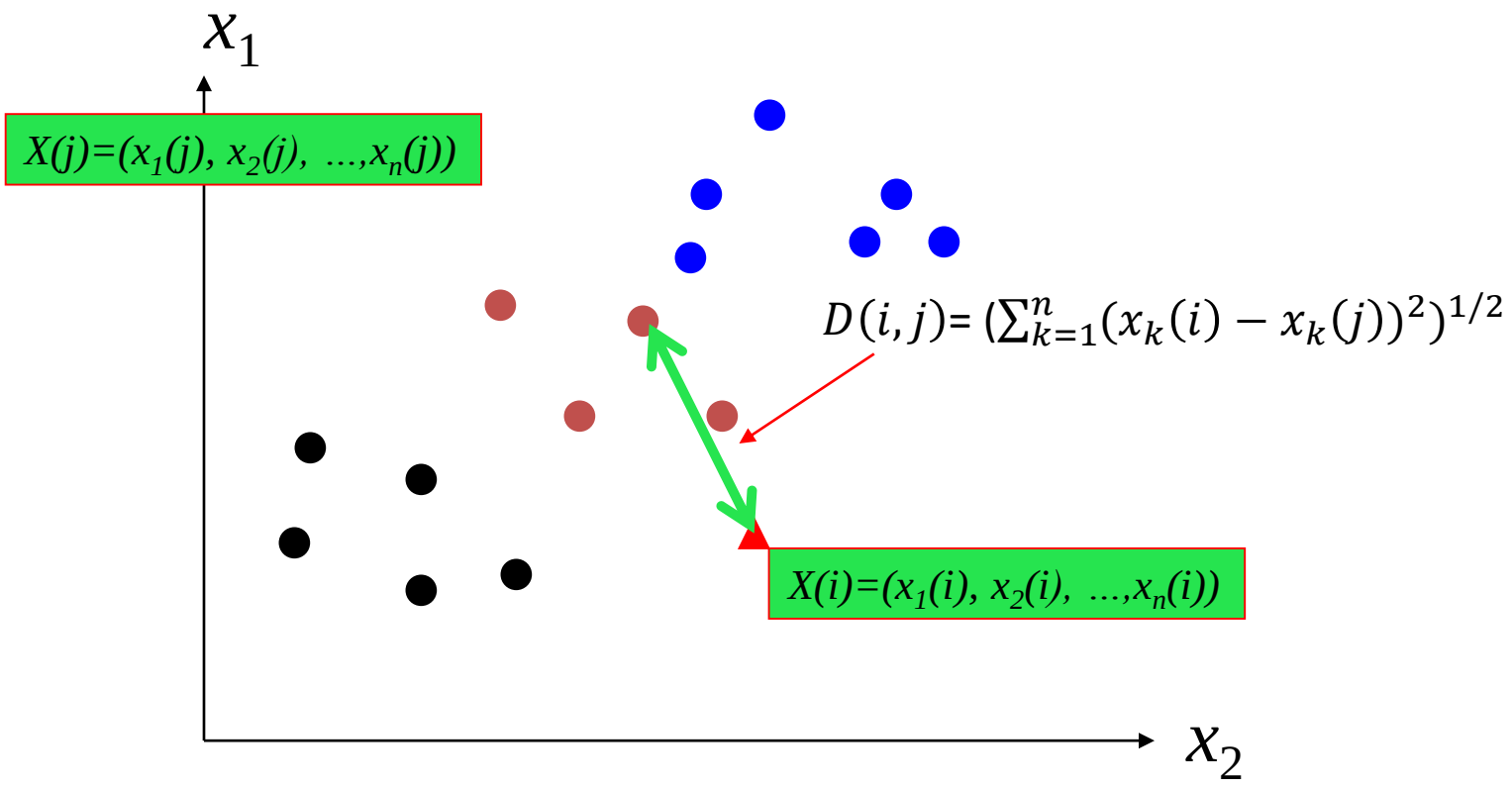
Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

Objects, Feature Vectors, Points



Elliptical blobs (objects)

Nearest Neighbours



Nearest Neighbour Algorithm

Given training data *(X(1),D(1)), (X(2),D(2)), …, (X(N),D(N)),*

Define a distance metric between points in inputs space.
Common measures are:

Euclidean Distance

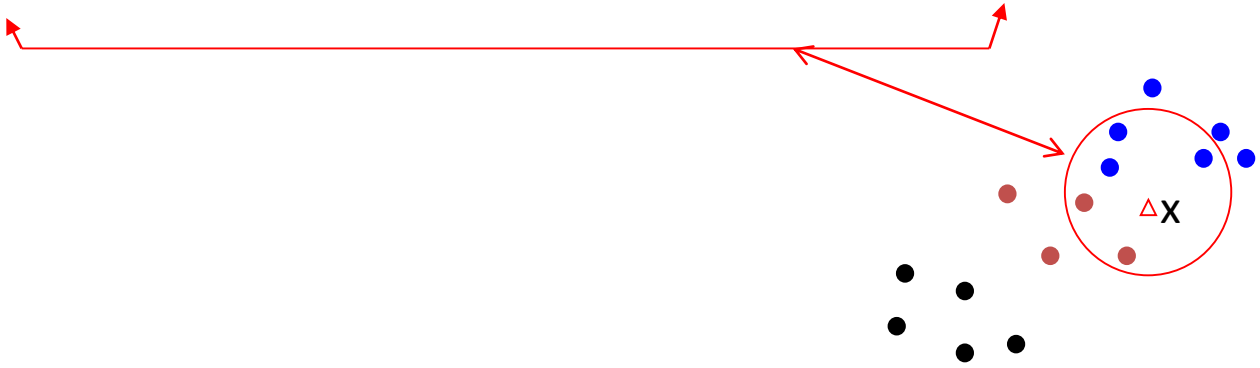
$$D(i,j) = \left(\sum_{k=1}^{n}(x_k(i) - x_k(j))^2\right)^{1/2}$$

K-Nearest Neighbour Model

Given test point X

- Find the K nearest training inputs to X
- Denote these points as

(X(1),D(1)), (X(2), D(2)), …, (X(k), D(k))



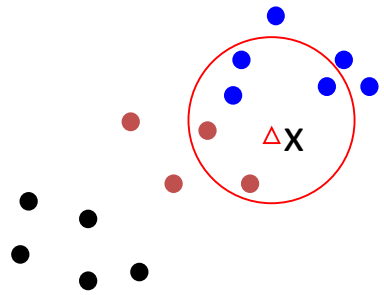
K-Nearest Neighbour Model

Instance based learning

The class identification of X

Y = most common class in set $\{D(1), D(2), \ldots, D(k)\}$

Majority rule



K-Nearest Neighbour Model

Example

Classify whether a customer will respond to a survey question using a 3-Nearest Neighbour classifier.

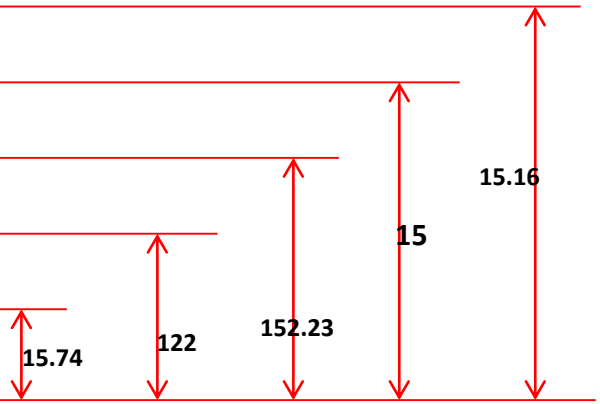
Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

K-Nearest Neighbour Model

Example

3-Nearest Neighbours

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

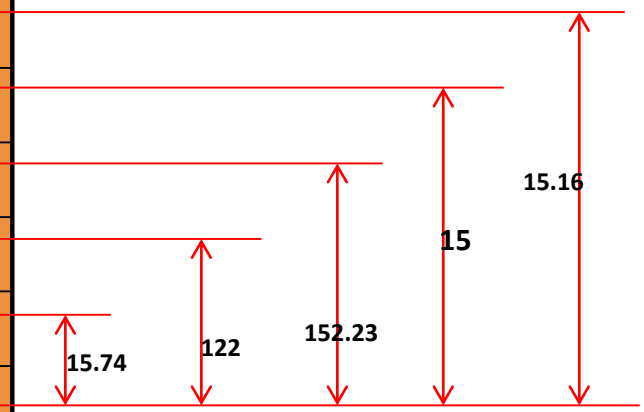


K-Nearest Neighbour Model

Example

3-Nearest Neighbours

Customer	Age	Income	No. credit cards	Response
John				No
Rachel				Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie				Yes
David	37	50K	2	?



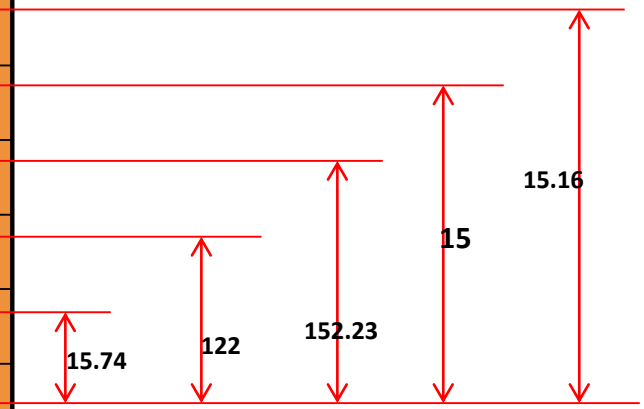
Three nearest ones to David are: No, Yes, Yes

K-Nearest Neighbour Model

Example

3-Nearest Neighbors

Customer	Age	Income	No. credit cards	Response
John				No
Rachel				Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie				Yes
David	37	50K	2	Yes



Three nearest ones to David are: No, Yes, Yes

K-Nearest Neighbour Model

Picking K

- Use *N fold cross validation* – Pick K to minimize the cross validation error
- For each of N training example
  - Find its K nearest neighbours
  - Make a classification based on these K neighbours
  - Calculate classification error
  - Output average error over all examples
- Use the K that gives lowest average error over the N training examples

K-Nearest Neighbour Model

Q&A

For the example we saw earlier, pick the best K from the set {1, 2, 3} to build a K-NN classifier.

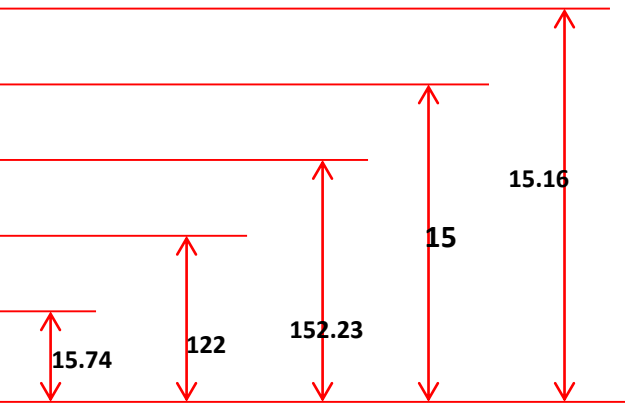
Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?

K-Nearest Neighbour Model

Q&A

For the example we saw earlier, pick the best K from the set {1, 2, 3} to build a K-NN classifier.

Customer	Age	Income	No. credit cards	Response
John	35	35K	3	No
Rachel	22	50K	2	Yes
Hannah	63	200K	1	No
Tom	59	170K	1	No
Nellie	25	40K	4	Yes
David	37	50K	2	?



University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

Topic 6 – Bayesian Learning

Ying Weng

2024 Autumn

Marginalization

$$P(A = v_1 \cup A = v_2 \cup \cdots \cup A = v_i) = \sum_{j=1}^{i} P(A = v_j)$$

$$P(B \cap [A = v_1 \cup A = v_2 \cup \cdots \cup A = v_i]) = \sum_{j=1}^{i} P(B \cap A = v_j)$$

$$P(B) = \sum_{j=1}^{k} P(B \cap A = v_j)$$

Marginalization: We can recover probability distribution of any variable in a joint distribution by summing over the other variables

Bayes Rule

- **Bayes, Thomas (1763)** An essay towards solving a problem in the doctrine of chances. Philosophical Transactions of the Royal Society of London, **53:370-418**

- **Bayes Rule**

$$p(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B \mid A)P(A)}{P(B)}$$

Bayesian Learning

$$p(h \mid x) = \frac{P(x \mid h)P(h)}{P(x)}$$

Understanding Bayes' rule

x = data

h = hypothesis (model)

- rearranging

$p(h \mid x)P(x) = P(x \mid h)P(h)$

$P(x, h) = P(x, h)$

the same joint probability on both sides

$P(h)$: prior belief (probability of hypothesis *h* before seeing any data)

$P(x \mid h)$: likelihood (probability of the data if the hypothesis *h* is true)

$P(x) = \sum_h P(x \mid h)P(h)$: data evidence (marginal probability of the data)

$P(h \mid x)$: posterior (probability of hypothesis *h* after having seen the data *d*)

Choosing Hypotheses

- Generally, we want the most probable hypothesis(class label) given the observed data
  - Maximum a posteriori (**MAP**) hypothesis
  - Maximum likelihood (**ML**) hypothesis

Maximum A Posteriori (MAP)

- Maximum a posteriori (**MAP**) hypothesis

$$p(h \mid x) = \frac{P(x \mid h)P(h)}{P(x)}$$

$$h_{MAP} = \arg\max_{h \in H} p(h \mid x) = \arg\max_{h \in H} \frac{P(x \mid h)P(h)}{P(x)} = \arg\max_{h \in H} P(x \mid h)P(h)$$

Note $P(x)$ is independent of h, hence can be ignored.

Maximum Likelihood (ML)

$$h_{MAP} = \arg\max_{h \in H} P(x \mid h)P(h)$$

- Assuming that each hypothesis in H is equally probable, i.e., $P(h_i) = P(h_j)$, for all i and j, then we can drop $P(h)$ in MAP. $P(x|h)$ is often called the likelihood of data x given h. Any hypothesis that maximizes $P(x|h)$ is called the maximum likelihood hypothesis

$$h_{ML} = \arg\max_{h \in H} P(x \mid h)$$

An Illustrating Example

Classifying days according to whether someone will play tennis.

Each day is described by the attributes, Outlook, Temperature, Humidity and Wind.

Based on the training data in the table, classify the following instance

Outlook = sunny
Temperature = cool
Humidity = high
Wind = strong

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

An Illustrating Example

Training sample pairs (X, D)

$X = (x_1, x_2, \ldots, x_n)$ is the feature vector representing the instance.

$D = (d_1, d_2, \ldots d_m)$ is the desired (target) output of the classifier

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

An Illustrating Example

Training sample pairs (X, D)

$X = (x_1, x_2, \ldots, x_n)$ is the feature vector representing the instance.

n = 4

x_1= *outlook* = {*sunny, overcast, rain*}

x_2= *temperature* = {*hot, mild, cool*}

x_3= *humidity* = {*high, normal*}

x_4= *wind* = {*weak, strong*}

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

An Illustrating Example

Training sample pairs (X, D)

$D = (d_1, d_2, \ldots d_m)$ is the desired (target) output of the classifier

$m = 1$

$d = Play\ Tennis = \{yes, no\}$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Bayesian Classifier

- The Bayesian approach to classifying a new instance X is to assign it to the most probable target value Y (MAP classifier)

$$
\begin{aligned}
Y &= \arg\max_{d_i \in d} p(d_i \mid X) \\
&= \arg\max_{d_i \in d} p(d_i \mid x_1, x_2, x_3, x_4) \\
&= \arg\max_{d_i \in d} \frac{p(x_1, x_2, x_3, x_4 \mid d_i) P(d_i)}{p(x_1, x_2, x_3, x_4)} \\
&= \arg\max_{d_i \in d} p(x_1, x_2, x_3, x_4 \mid d_i) P(d_i)
\end{aligned}
$$

Bayesian Classifier

$$Y = \arg\max_{d_i \in d} p(x_1, x_2, x_3, x_4 \mid d_i) P(d_i)$$

$P(d_i)$ is easy to calculate: simply counting how many times each target value d_i occurs in the training set.

$P(d = yes) = 9/14$

$P(d = no) = 5/14$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Bayesian Classifier

$$Y = \arg\max_{d_i \in d} p(x_1, x_2, x_3, x_4 \mid d_i) P(d_i)$$

$P(x_1, x_2, x_3, x_4|d_i)$ is much more difficult to estimate.

In this simple example, there are 3x3x2x2x2 = 72 possible terms.

To obtain a reliable estimate, we need to see each terms many times.

Hence, we need a very, very large training set! (which in most cases is impossible to get).

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Naïve Bayes Classifier

Naïve Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value.

This means, we have

$$P(x_1, x_2, \cdots, x_n \mid d_i) = \prod_{i}^{n} P(x_i \mid d_i)$$

Naïve Bayes Classifier

$$Y = \arg\max_{d_i \in d} P(d_i) \prod_{k=1}^{4} P(x_k \mid d_i)$$

Back to the Example

Naïve Bayes Classifier

$$Y = \arg\max_{d_i \in d} \prod_{k=1}^{4} P(x_k \mid d_i) P(d_i)$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

$$Y = \arg\max_{d_i \in \{yes, no\}} P(d_i) P(x_1 = suny \mid d_i) P(x_2 = cool \mid d_i) P(x_3 = high \mid d_i) P(x_4 = strong \mid d_i)$$

Back to the Example

$$Y = \arg\max_{d_i \in \{yes,no\}} P(d_i)P(x_1 = suny \mid d_i)P(x_2 = cool \mid d_i)P(x_3 = high \mid d_i)P(x_4 = strong \mid d_i)$$

$P(d=yes)=9/14 = 0.64$

$P(d=no)=5/14 = 0.36$

$P(x_1=sunny|yes)=$ **?**
$P(x_1=sunny|no)=$ **?**

$P(x_2=cool|yes)=$ **?**
$P(x_2=cool|no)=$ **?**

$P(x_3=high|yes)=$ **?**
$P(x_3=high|no)=$ **?**

$P(x_4=strong|yes)=$ **?**
$P(x_4=strong|no)=$ **?**

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Back to the Example

$$Y = \arg\max_{d_i \in \{yes,no\}} P(d_i)P(x_1 = suny \mid d_i)P(x_2 = cool \mid d_i)P(x_3 = high \mid d_i)P(x_4 = strong \mid d_i)$$

$P(d=yes)=9/14 = 0.64$

$P(d=no)=5/14 = 0.36$

$P(x_1=sunny|yes)=2/9$
$P(x_1=sunny|no)=3/5$

$P(x_2=cool|yes)=3/9$
$P(x_2=cool|no)=1/5$

$P(x_3=high|yes)=3/9$
$P(x_3=high|no)=4/5$

$P(x_4=strong|yes)=3/9$
$P(x_4=strong|no)=3/5$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Back to the Example

$$Y = \arg\max_{d_i \in \{yes,no\}} P(d_i)P(x_1 = suny \mid d_i)P(x_2 = cool \mid d_i)P(x_3 = high \mid d_i)P(x_4 = strong \mid d_i)$$

$$P(yes)P(x_1 = suny \mid yes)P(x_2 = cool \mid yes)P(x_3 = high \mid yes)P(x_4 = strong \mid yes) = ?$$

$$P(no)P(x_1 = suny \mid no)P(x_2 = cool \mid no)P(x_3 = high \mid no)P(x_4 = strong \mid no) = ?$$

Y = *Play Tennis* = *yes or no?*

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Back to the Example

$$Y = \arg\max_{d_i \in \{yes,no\}} P(d_i)P(x_1 = suny \mid d_i)P(x_2 = cool \mid d_i)P(x_3 = high \mid d_i)P(x_4 = strong \mid d_i)$$

$$P(yes)P(x_1 = suny \mid yes)P(x_2 = cool \mid yes)P(x_3 = high \mid yes)P(x_4 = strong \mid yes) = 0.0053$$

$$P(no)P(x_1 = suny \mid no)P(x_2 = cool \mid no)P(x_3 = high \mid no)P(x_4 = strong \mid no) = 0.0206$$

$Y = Play\ Tennis = no$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Estimating Probabilities

- So far, we estimate the probabilities by the fraction of times the event is observed to occur over the entire opportunities

- In the above example, we estimated

  $P(wind=strong|play\ tennis=no)=Nc/N$,
  where $N = 5$ is the total number of training samples for which $play\ tennis = no$, and Nc is the number of these for which wind=strong

- What happens if $Nc = 0$?

Estimating Probabilities

- When Nc is small, however, such approach provides poor estimation. To avoid this difficulty, we can adopt the **m-estimate** of probability

$$\frac{N_C + mP}{N + m}$$

where P is the prior estimate of the probability we wish to estimate, m is a constant called the equivalent sample size, which determines how heavily to weight P relative to the observed data.

A typical method for choosing P in the absence of other information is to assume uniform priors: If an attribute has k possible values we set $P=1/K$.

Back to the Example

Estimating Probabilities

- For example: *P(wind =strong | play tennis=no)*, we note *wind* has two possible values: *weak*, *strong*; so the uniform priors of *wind P =1/2*

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Back to the Example

Estimating Probabilities

- Question: *P(outlook =sunny | play tennis=yes)*, what is the uniform priors of *outlook*?

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Back to the Example

Estimating Probabilities

- Solution: $P(outlook = sunny \mid play\ tennis = yes)$, we note *outlook* has three possible values: *sunny*, *overcast*, *rain*; so the uniform priors of *wind* $P = 1/3$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Naïve Bayesian Classifier

- What about using Naïve Bayesian Classifier for our handwritten digit recognition problem?
  - Each pixel is an x_i. There will be 784 x's.
  - Digit label is d_k. Note there will be 10 possible d's.
  - $P(d_k)$ can be calculated by counting number of training images for the digit, divided to total number of training images.
  - $P(x_i|d_k)$ can be calculated by counting number of images for a given digit, given pixel position, and given an intensity value, divided by number of training images with that digit.

Naïve Bayesian Classifier

$P(x_i|d_k)$ can be calculated by counting number of images for a given digit, given pixel position, and given an intensity value, divided by number of training images with that digit.

For example,

$$P(x_1 = 255|d = 0)$$
$$= \frac{Number\ of\ images\ whose\ first\ pixel\ value\ is\ 255\ and\ contain\ digit\ 0}{Total\ number\ of\ images\ contain\ digit\ 0}$$

Naïve Bayesian Classifier

- For a given input image X and given digit label d_k, calculate $P(d_k)$ and all $P(x_i|d_k)$
- For each digit label d_k, calculate
  $P(d_k|X) = P(d_k)P(x_1 = 0|d_k)P(x_2 = 255|d_k) \dots P(x_{784} = 0|d_k)$
- Choose the digit label k that give the max value according to above calculation.

0	255	0	0	0	0	0	0	0	0
0	0	0	0	255	255	0	0	0	0
0	0	0	255	0	0	255	0	0	0
0	0	0	0	0	0	255	0	0	0
0	0	0	0	0	255	0	0	0	0
0	0	0	0	255	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0
0	0	0	255	255	255	255	255	255	0

Input image X

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

Topic 8 – Data Clustering

Ying Weng

2024 Autumn

Supervised VS Unsupervised Learning

- Supervised learning
  - Learns a function that maps an input to an output based on example input-output pairs.
  - Training data is labeled.
- Unsupervised learning
  - Learns from test data that has not been labeled.
  - Learns relationships between elements in a data set and classify the raw data without "help."
  - Typical application includes data clustering.

K-Means

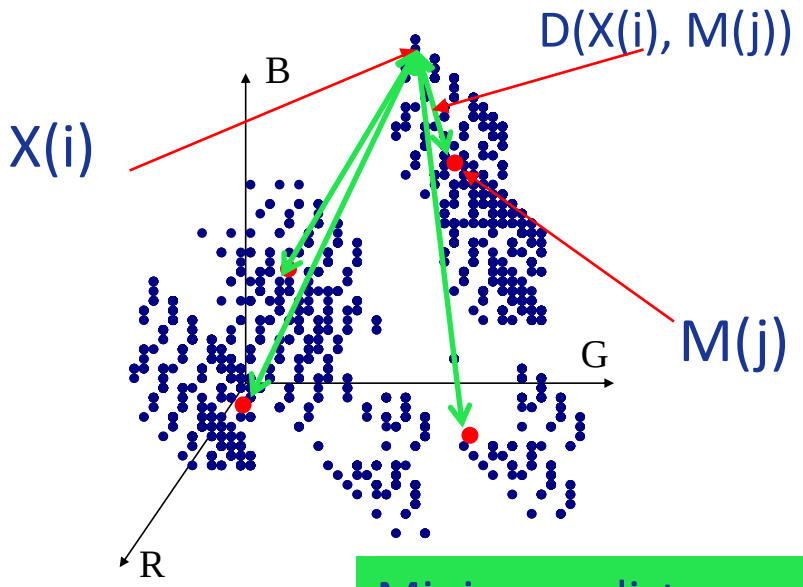
- An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing N_j data points
  - Define, $X(i) = [x_1(i), x_2(i), \ldots, x_n(i)]$, $i = 1, 2, \ldots N$, as N data points
  - We want to cluster these N points into K subsets, or K clusters, where K is pre-set
  - For each cluster, we define $M(j) = [m_1(j), m_2(j), \ldots, m_n(j)]$, $j=1, 2, \ldots K$, as its prototype or cluster centroids
  - Define the distance between data point X(i) and cluster prototype M(j) as

$$D\big(X(i), M(j)\big) = \big\|X(i) - M(j)\big\| = \sqrt{\sum_{l=1}^{n} \big(x_l(i) - m_l(j)\big)^2}$$

K-Means

- A data point X(i) is assigned to the jth cluster, C(j), X(i) $\in$ C(j), if following condition holds

$$D\big(X(i), M(j)\big) \leq D\big(X(i), M(l)\big) \quad \text{for all } l = 1, 2, ..., k$$



Minimum distance classifier

K-Means Algorithm

Step 1

- ✪ Arbitrarily choose from the given sample set k initial cluster centres,

$M^{(0)}(j) = [m^{(0)}_1(j), m^{(0)}_2(j), ,..., m^{(0)}_n(j)]$ $j = 1, 2, ..., K$,

e.g., the first K samples of the sample set
or can also be generated randomly

Set $t = 0$ (t is the iteration index)

K-Means Algorithm

Step 2

- Assign each of the samples X(i) = [x_1(i), x_2(i), …, x_n(i)], i = 1, 2, ….N, to one of the clusters according to the distance between the sample and the centre of the cluster:

$$X(i) \in C^{(t)}(j)$$
$$if \quad D\left(X(i), M^{(t)}(j)\right) \leq D\left(X(i), M^{(t)}(l)\right)$$
$$\text{for all } l = 1, 2, ..., k$$

K-Means Algorithm

Step 3

Update the cluster centres to get

$M^{(t+1)}(j) = [m^{(t+1)}_1(j), m^{(t+1)}_2(j), ,..., m^{(t+1)}_n(j)]$; j = 1, 2, ..., K

according to

$$M^{(t+1)}(j) = \frac{1}{N_j^{(t)}} \sum_{X(i) \in C^{(t)}(j)} X(i)$$

$N^{(t)}_j$ is the number of samples in $C^{(t)}_j$

K-Means Algorithm

Step 4

– Calculate the error of approximation

$$E(t) = \sum_{j=1}^{K} \sum_{X(i) \in C^{(t)}(j)} \left\| X(i) - M^{(t)}(j) \right\|$$

K-Means Algorithm

Step 5

– If the terminating criterion is met, then stop, otherwise

Set t = t+1

Go to Step 2.

K-Means Algorithm

Stopping criterions

- The K-means algorithm can be stopped based on following criterions

1. The errors do not change significantly in two consecutive epochs

   $|E(t)-E(t-1)|<\varepsilon$, where ε is some preset small value

2. No further change in the assignment of the data points to clusters in two consecutive epochs.

3. It can also stop after a fixed number of epochs regardless of the error

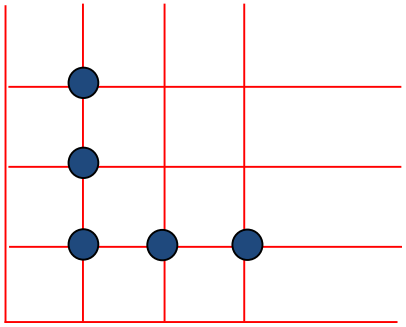
K-Means Algorithm

A worked example to see how it works exactly

Five 2-dimensional data points:

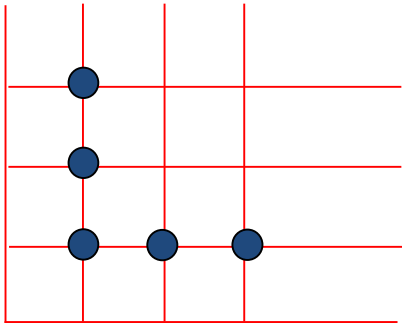
(1, 1), (2, 1), (3, 1), (1, 2), (1, 3)

Cluster them into two clusters and find the cluster centres



K-Means Algorithm

A worked example to see how it works exactly



(1) Euclidean distance to $m_1^0(1,2)$

$\sqrt{(1-1)^2+(1-2)^2} = 1$

$\sqrt{(2-1)^2+(1-2)^2} = \sqrt{2} = 1.41$

$\sqrt{(3-1)^2+(1-2)^2} = \sqrt{5} = 2.24$

$\sqrt{(1-1)^2+(2-2)^2} = 0$

$\sqrt{(1-1)^2+(3-2)^2} = 1$

Euclidean distance to $m_2^0(3,1)$

$\sqrt{(1-3)^2+(1-1)^2} = 2$ $\therefore C_1$

$\sqrt{(2-3)^2+(1-1)^2} = 1$ $\therefore C_2$

$\sqrt{(3-3)^2+(1-1)^2} = 0$ $\therefore C_2$

$\sqrt{(1-3)^2+(2-1)^2} = \sqrt{5} = 2.24$ $\therefore C_1$

$\sqrt{(1-3)^2+(3-1)^2} = \sqrt{8} = 2.83$ $\therefore C_1$

C_1 class: (1,1), (1,2), (1,3) $\Rightarrow m_1^{(0+1)}$:

$$\frac{1}{3}\sum_{i=1}^{3} x_i = \frac{1+1+1}{3} = 1$$

$$\frac{1}{3}\sum_{i=1}^{3} y_i = \frac{1+2+3}{3} = 2$$

C_2 class: (2,1), (3,1) $\Rightarrow m_2^{(0+1)}$:

$$\frac{1}{2}\sum_{i=1}^{2} x_i = \frac{2+3}{2} = 2.5$$

$$\frac{1}{2}\sum_{i=1}^{2} y_i = \frac{1+1}{2} = 1$$

(2) Euclidean distance to $m_1'(1,2)$

1

$\sqrt{2} = 1.41$

$\sqrt{5} = 2.24$

0

1

Euclidean distance to $m_2'(2.5,1)$

$\sqrt{(1-2.5)^2+(1-1)^2} = 1.5$ $\therefore C_1$

$\sqrt{(2-2.5)^2+(1-1)^2} = 0.5$ $\therefore C_2$

$\sqrt{(3-2.5)^2+(1-1)^2} = 0.5$ $\therefore C_2$

$\sqrt{(1-2.5)^2+(2-1)^2} = \sqrt{3.25} = 1.80$ $\therefore C_1$

$\sqrt{(1-2.5)^2+(3-1)^2} = \sqrt{6.25} = 2.5$ $\therefore C_1$

$\therefore C_1$ class: (1,1), (1,2), (1,3)

C_2 class: (2,1), (3,1)

K-Means Algorithm

- What is the algorithm doing exactly?
  - It tries to find the centre vectors M(j)'s that optimize the following cost function

$$E = \sum_{j=1}^{K} \sum_{X(i) \in C(j)} \| X(i) - M(j) \|$$

K-Means Algorithm

- Some remarks
  - Is a gradient descent algorithm, trying to minimize a cost function *E*
  - In general, the algorithm does not achieve a global minimum of E over the assignments
  - Sensitive to initial choice of cluster centers. Different starting cluster centroids may lead to different solution
  - Is a popular method, many more advanced methods derived from this simple algorithm

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

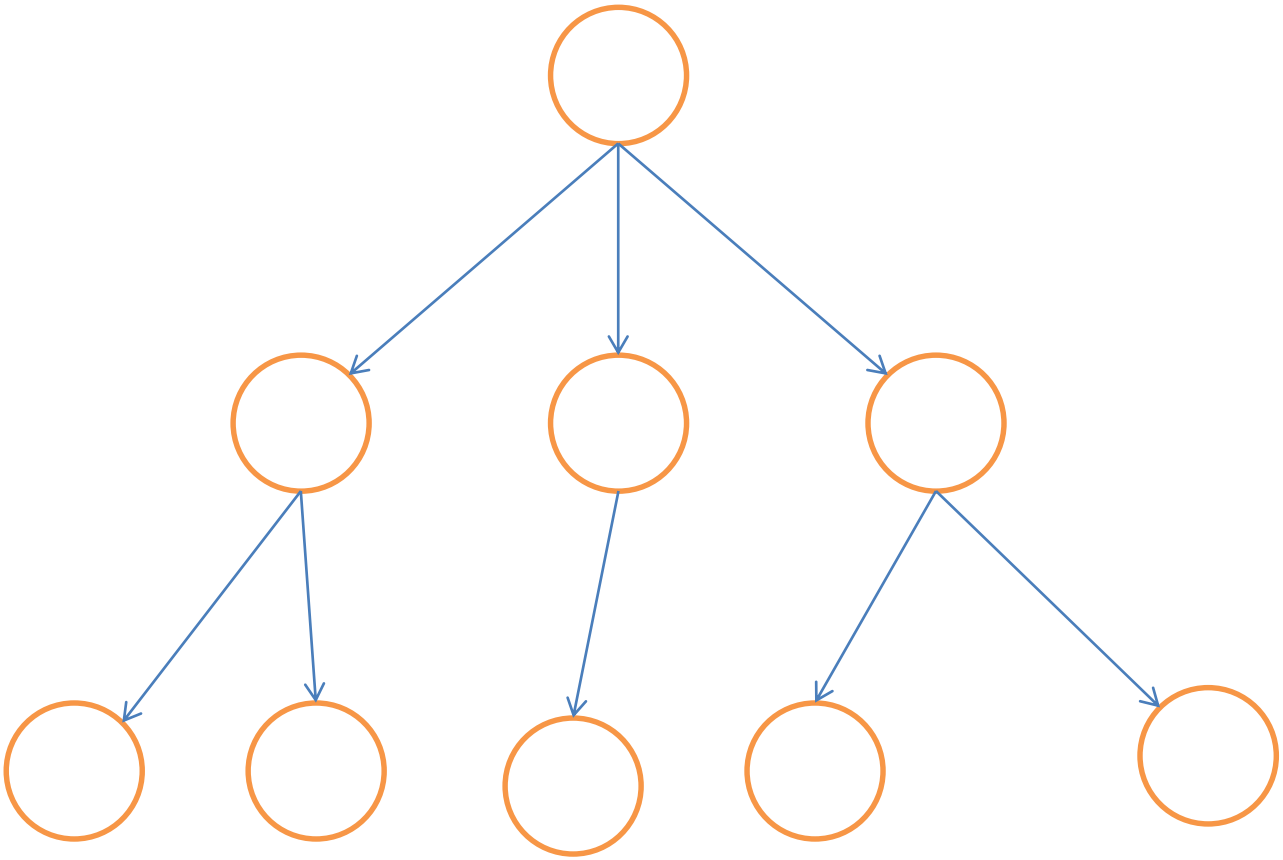
Topic 9 – Decision Tree

Ying Weng

2024 Autumn

Trees

- Node
- Root
- Leaf
- Branch
- Path
- Depth



Decision Trees

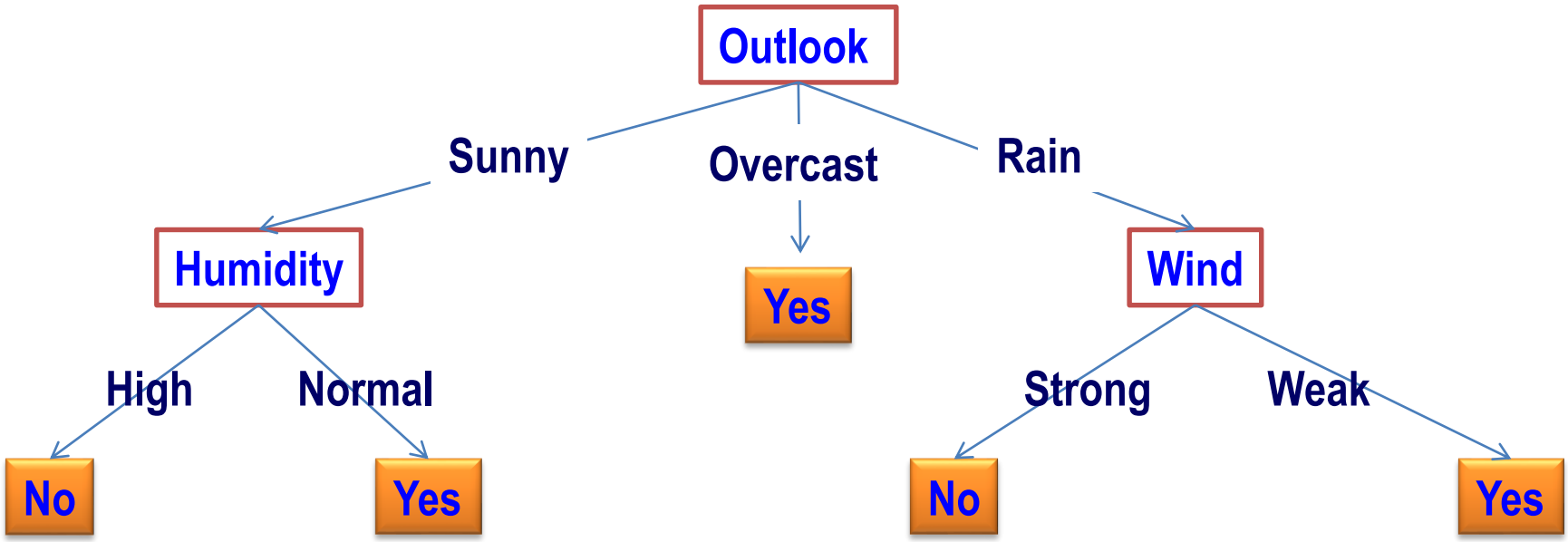
- A hierarchical data structure that represents data by implementing a divide and conquer strategy
- Can be used as a non-parametric classification method
- Given a collection of examples, learn a decision tree that represents it
- Use this representation to classify new examples

Decision Trees

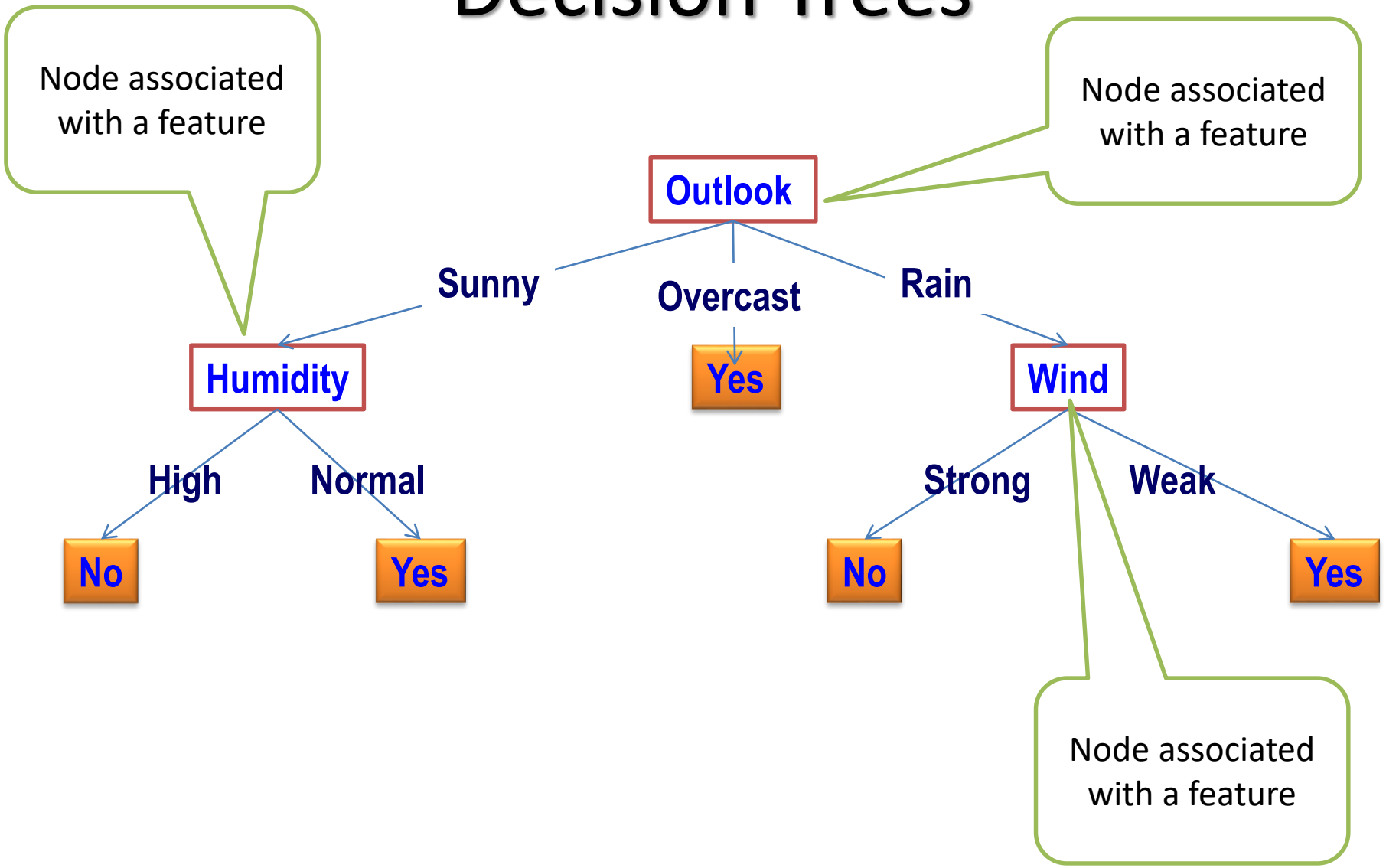
- Each node is associated with a feature (one of the elements of a feature vector that represent an object)
- Each node test the value of its associated feature
- There is one branch for each value of the feature
- Leaves specify the categories (classes)
- Can categorize instances into multiple disjoint categories – multi-class

Decision Trees

- Play Tennis Example
- Feature Vector = (Outlook, Temperature, Humidity, Wind)



Decision Trees

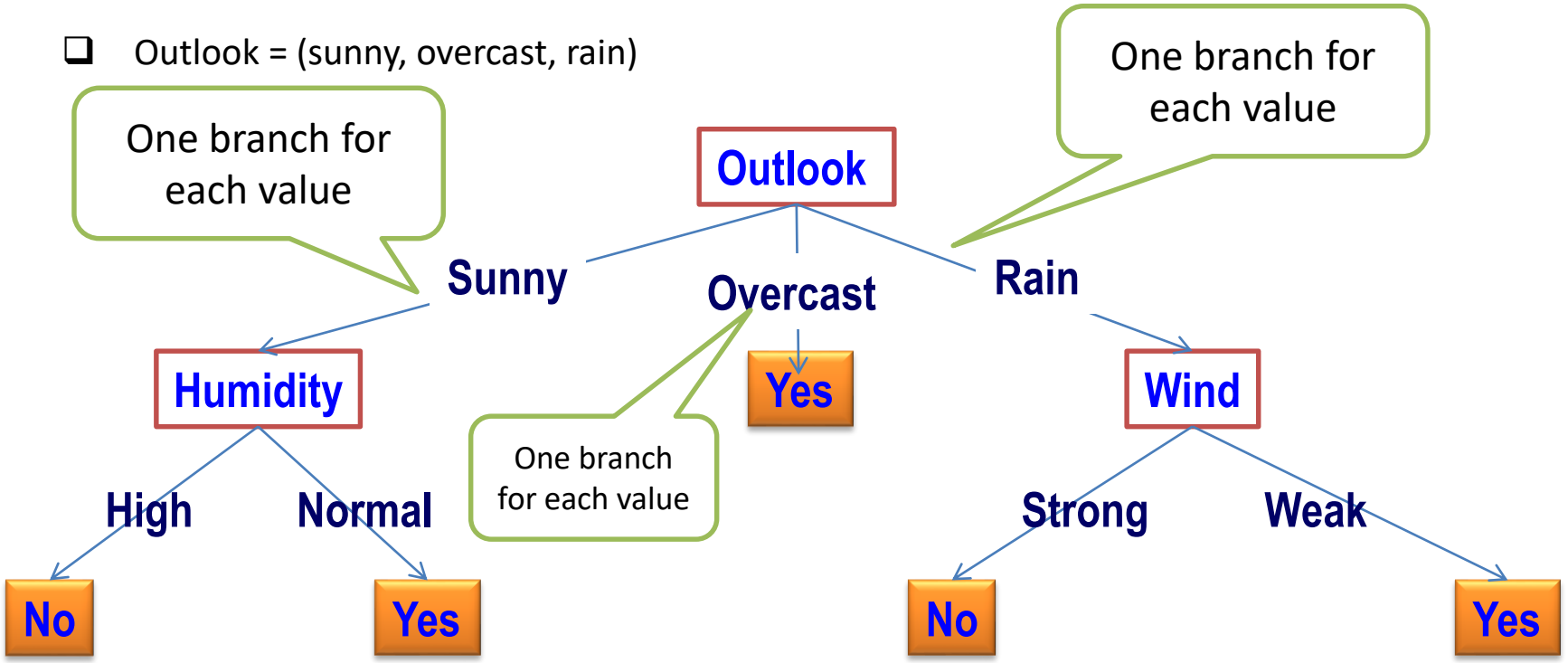


Decision Trees

- Play Tennis Example
- Feature values:
  - Outlook = (sunny, overcast, rain)
  - Temperature =(hot, mild, cool)
  - Humidity = (high, normal)
  - Wind =(strong, weak)

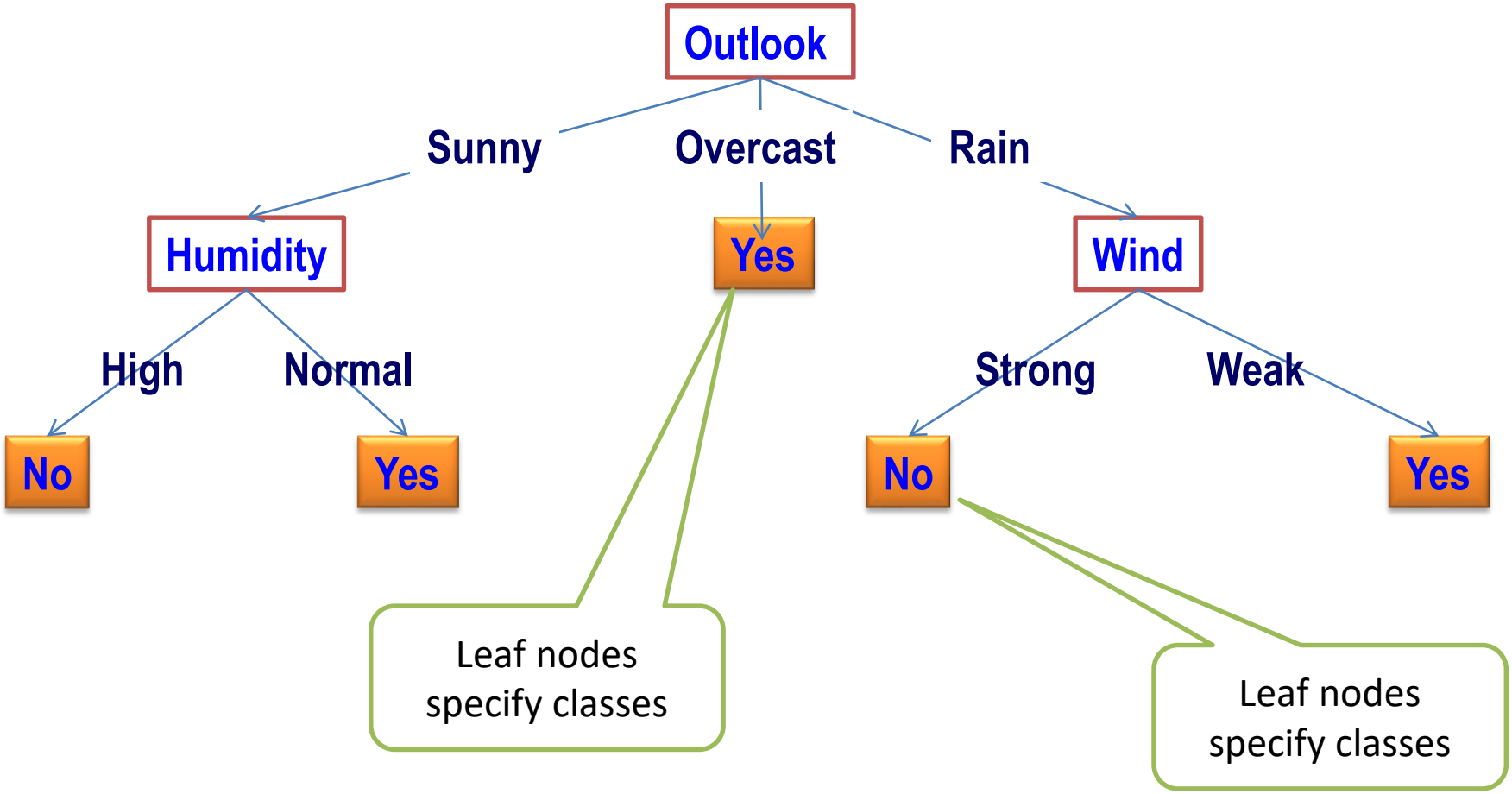
Decision Trees

- Outlook = (sunny, overcast, rain)



Decision Trees

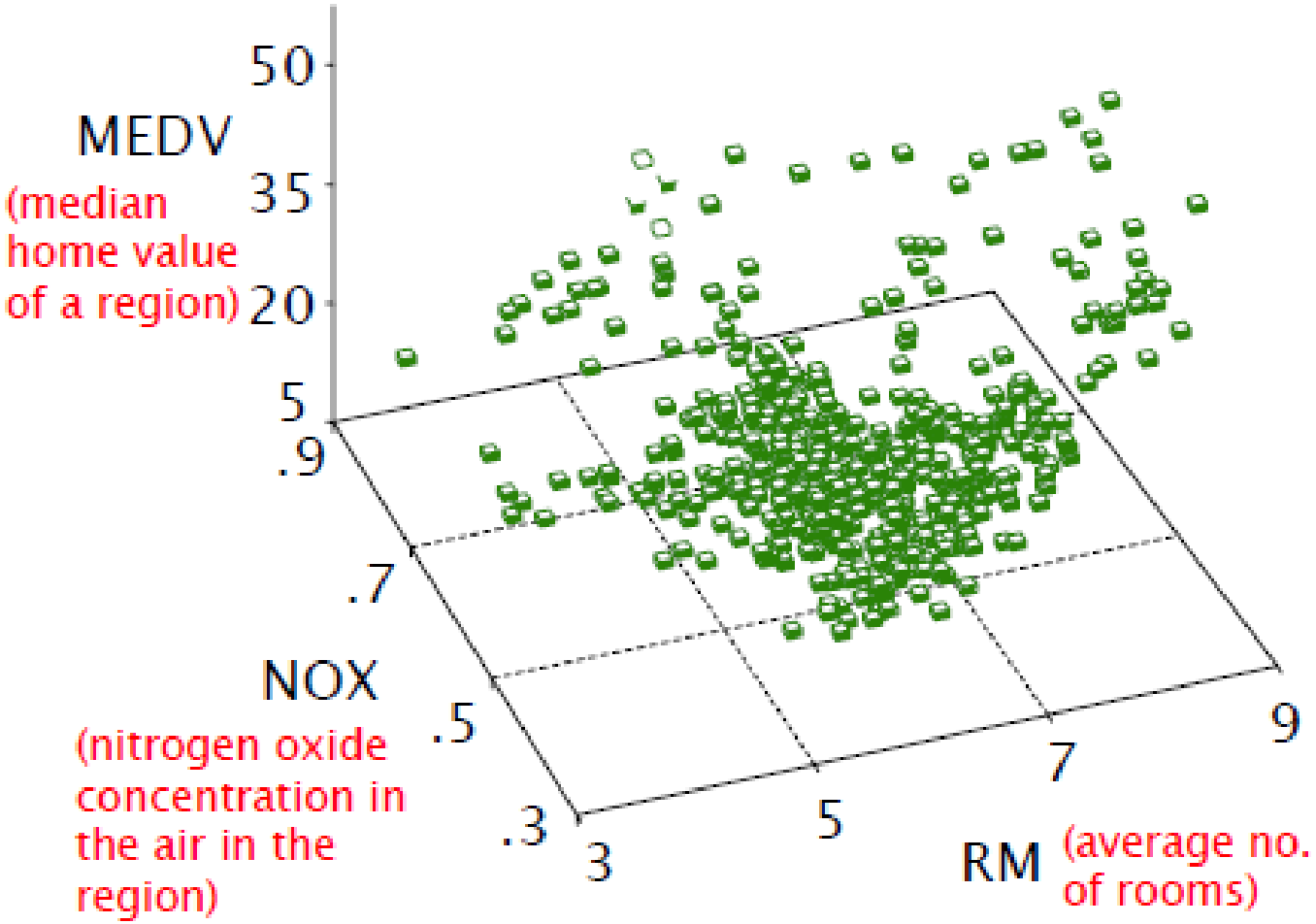
- Class = (Yes, No)



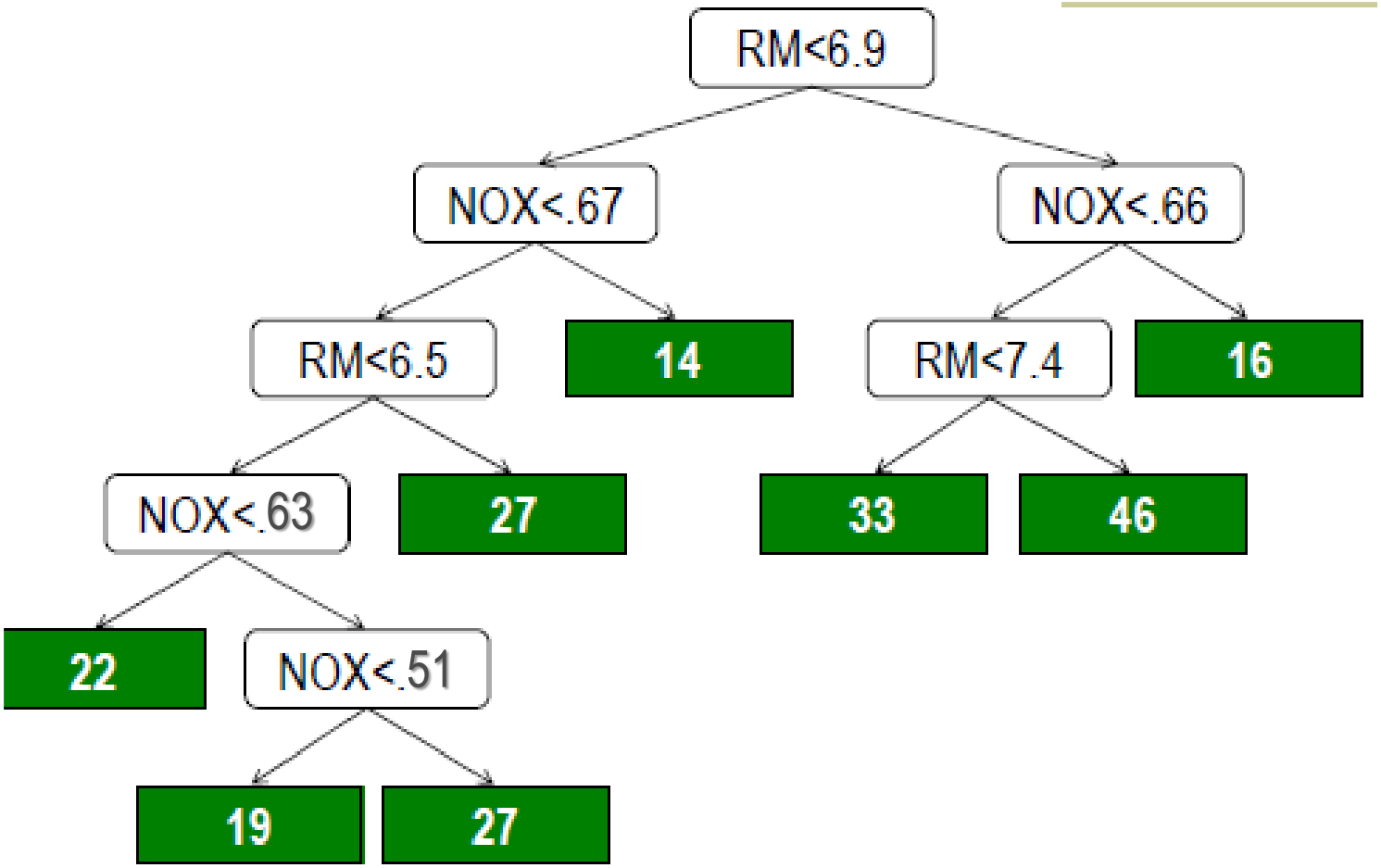
Variation of Decision Trees

- Classification tree
  - The output is discrete (label).
  - The leaves give the predicted class as well as the probability of class membership.
- Regression tree
  - The output is continuous.
  - The leaves give the predicted value of the output.
- Tree with binary splits
- Tree with multiway splits

Boston Housing Data



Regression Tree



Decision Trees

- Design Decision Tree Classifier
  - Picking the root node
  - Recursively branching

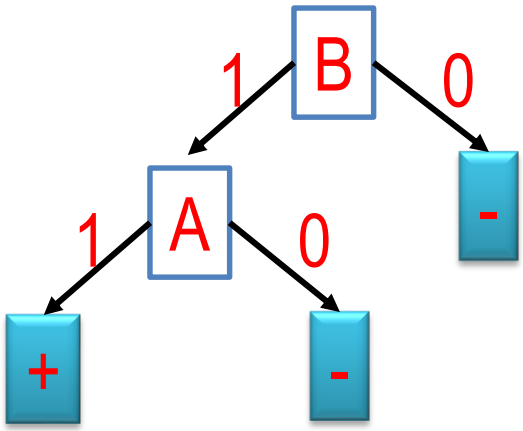
Decision Trees

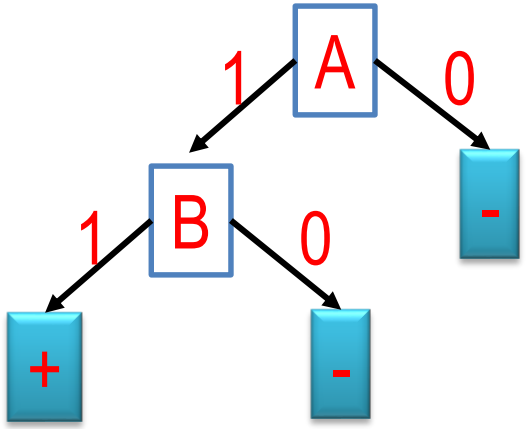
- Picking the root node
  - Consider data with two Boolean attributes (A,B) and two classes + and –

{ (A=0,B=0), - }: 50 examples
{ (A=0,B=1), - }: 50 examples
{ (A=1,B=0), - }: 3 examples
{ (A=1,B=1), + }: 100 examples

Decision Trees

- Picking the root node
  - Trees looks structurally similar; which attribute should we choose?





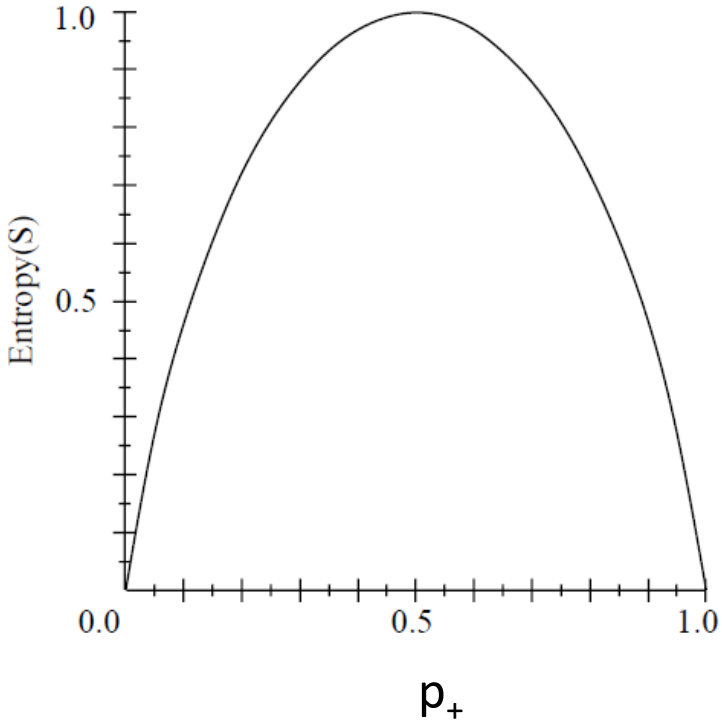
Decision Trees

- Picking the root node

  - The goal is to have the resulting decision tree as small as possible (Occam's Razor)
  - The main decision in the algorithm is the selection of the next attribute to condition on (start from the root node).
  - We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node.
  - The most popular heuristics is based on information gain, originated with the ID3 system of Quinlan.

Entropy

- S is a sample of training examples
- p_+ is the proportion of positive examples in S
- p_- is the proportion of negative examples in S
- Entropy measures the impurity of S



$$Entropy(S) = -p_+ \log(p_+) - p_- \log(p_-)$$

Entropy(IIP)

- The idea: associate information with probability
- A random event *E* with probability *P*(*E*) contains:

$$I(E) = log(\frac{1}{P(E)}) = -log(P(E))$$

units of information

- Suppose that grey level values are generated by a random variable, then r_k contains:

$$I(r_k) = -log(P(r_k))$$

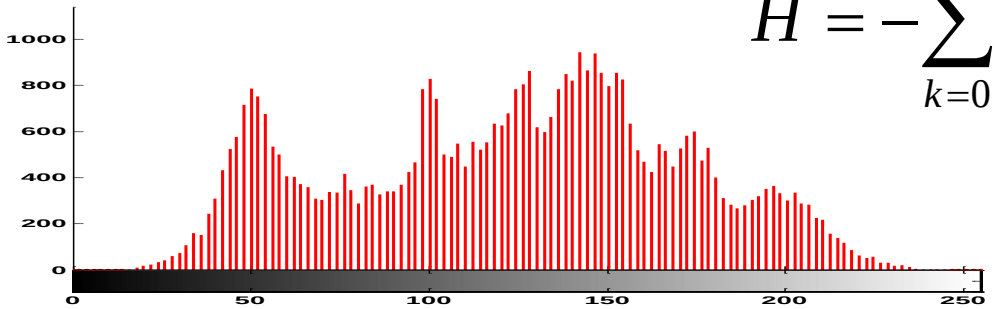
Note: I(E)=0 when P(E)=1

units of information

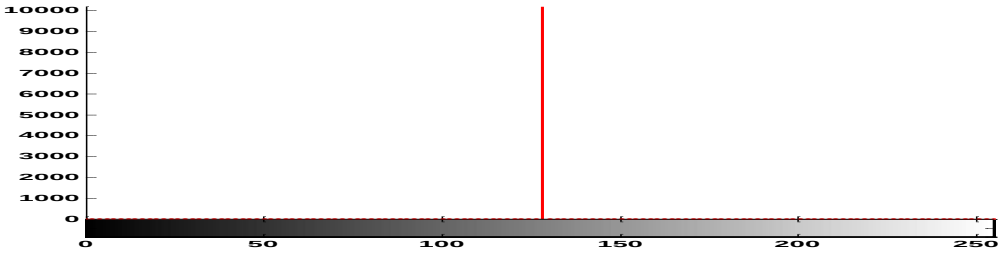
Entropy(IIP)

- Entropy is the average information content of an image, a measure of histogram dispersion

$$H = -\sum_{k=0}^{L-1} p(r_k)\log_2 p(r_k)$$



entropy=7.4635



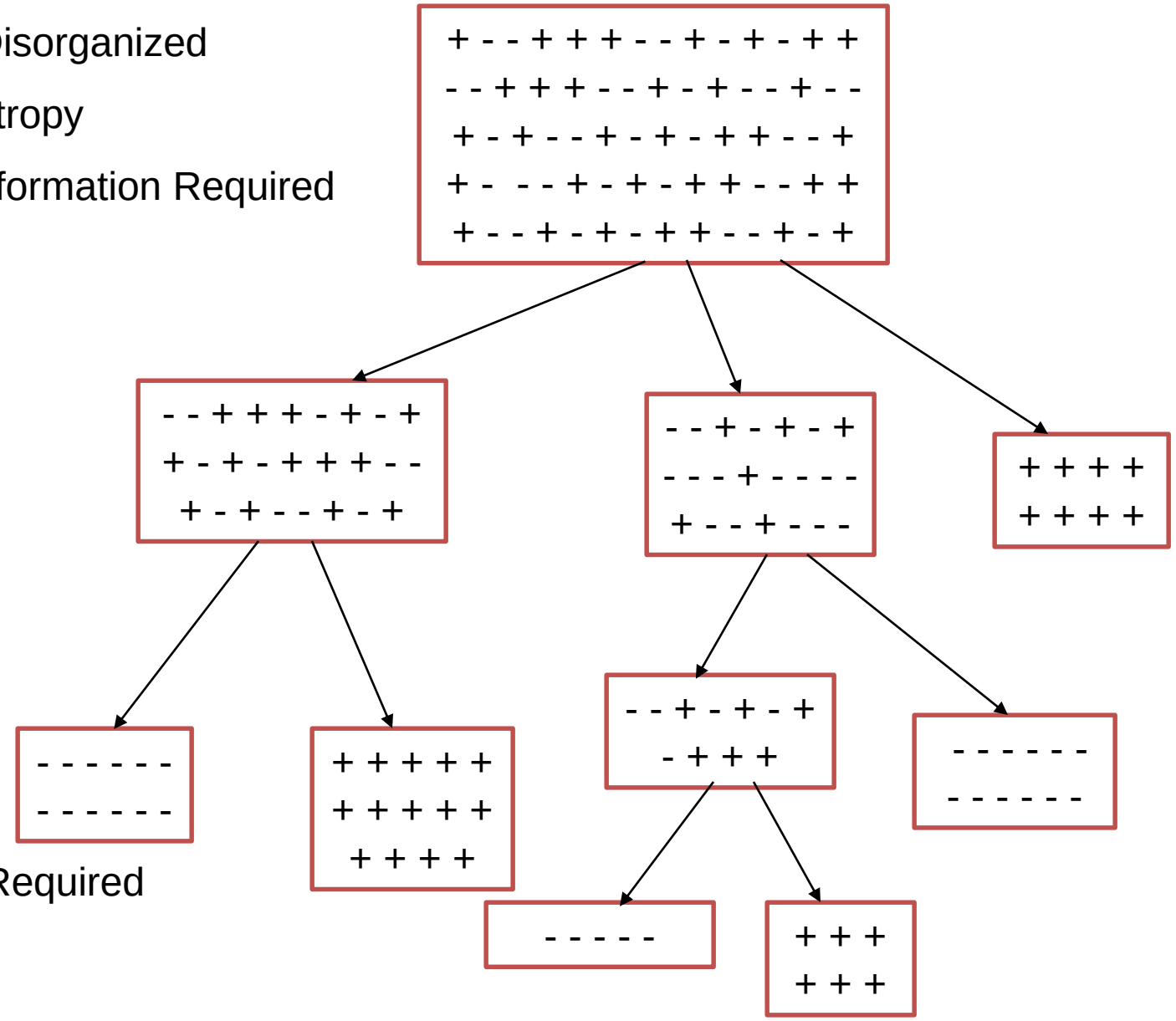
entropy=0

- Can't compress to less than H bits/pixel without losing information

Highly Disorganized

High Entropy

Much Information Required



Highly Organized

Low Entropy

Little Information Required

Information Gain

- Gain (S, A) = expected reduction in entropy due to sorting on A

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

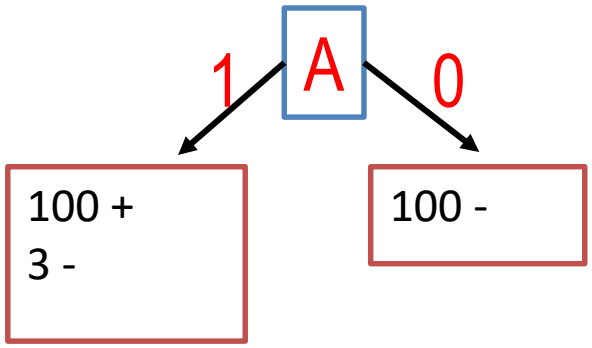
- Values (A) is the set of all possible values for attribute A, S_v is the subset of S which attribute A has value v, |S| and | S_v | represent the number of samples in set S and set S_v respectively

- Gain(S,A) is the expected reduction in entropy caused by knowing the value of attribute A.

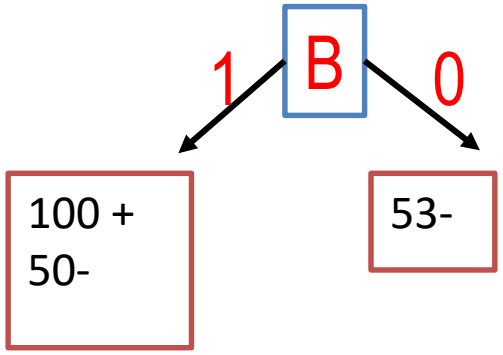
Information Gain

Example: Choose A or B ?

Split on A



Split on B



Example

Play Tennis Example

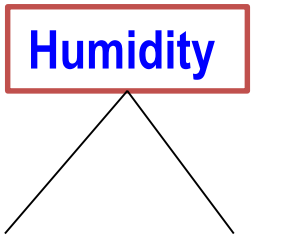
$$\text{Entropy}(S) = -\frac{9}{14}\log\left(\frac{9}{14}\right) - \frac{5}{14}\log\left(\frac{5}{14}\right) = 0.94$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Example

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No



High 3+,4- *E*=0.985

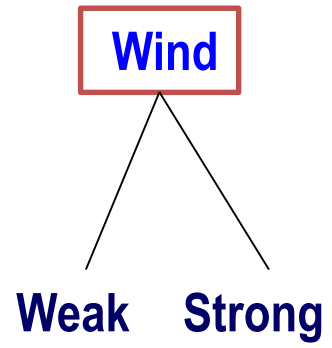
Normal 6+,1- *E*=0.592

***Gain(S,* Humidity*)* = .94 - 7/14 * 0.985 - 7/14 *.592 = 0.151**

Example

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No



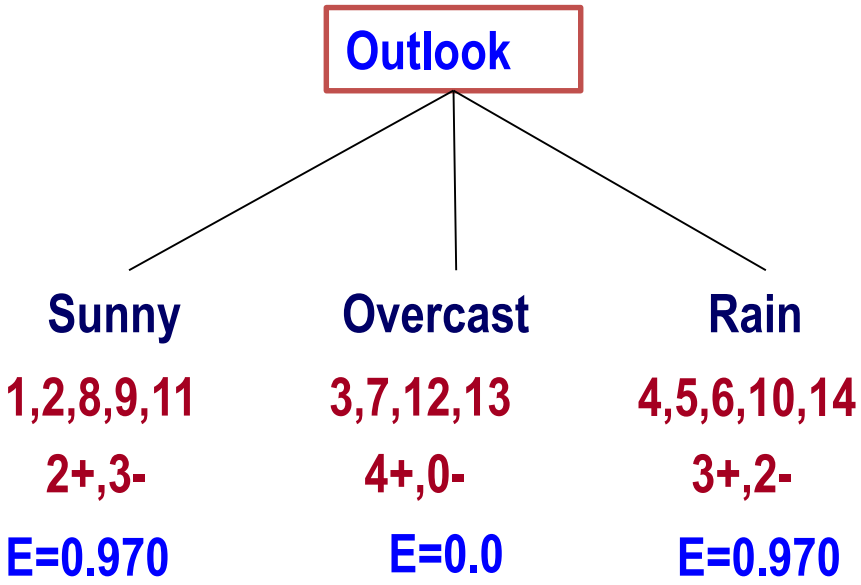
6+,2- 3+,3-

E=0.811 *E*=1.0

Gain(S, Wind*)* = .94 - 8/14 * 0.811 - 6/14 * 1.0 = 0.048

Example

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

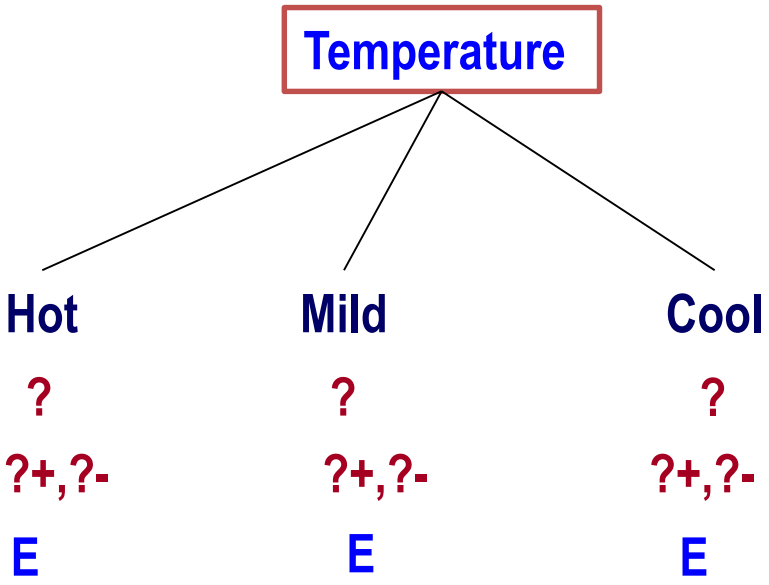


Gain(S, Outlook*)* = 0.246

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Example

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

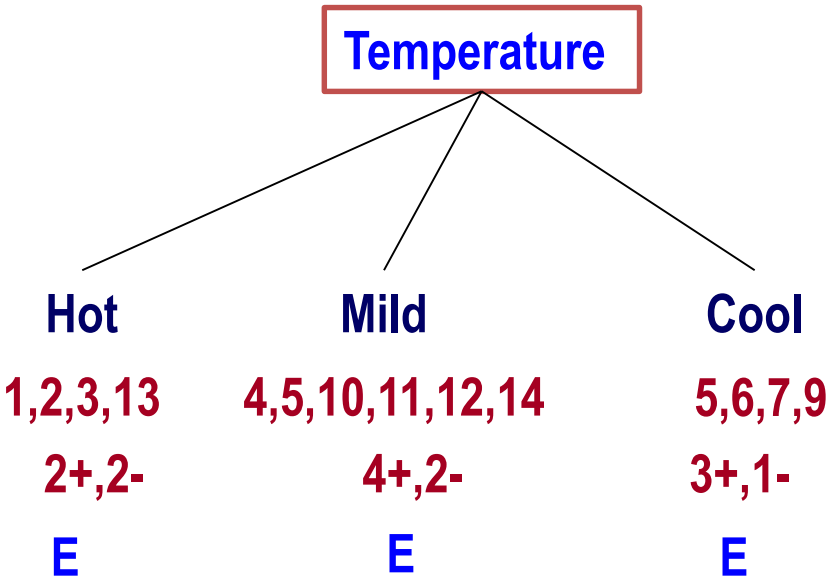


***Gain(S,* Temperature*)* = ?**

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Example

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

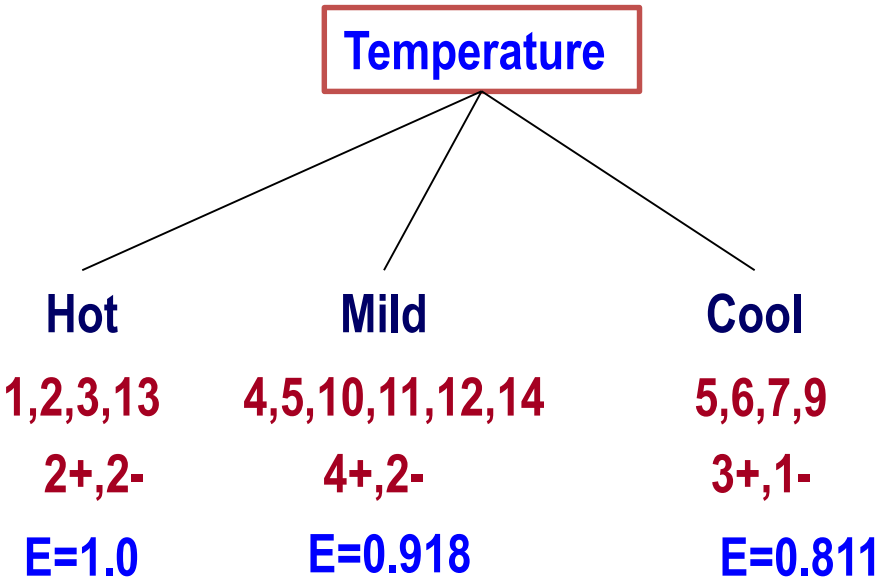


Gain(S, Temperature*)* = ?

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Example

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



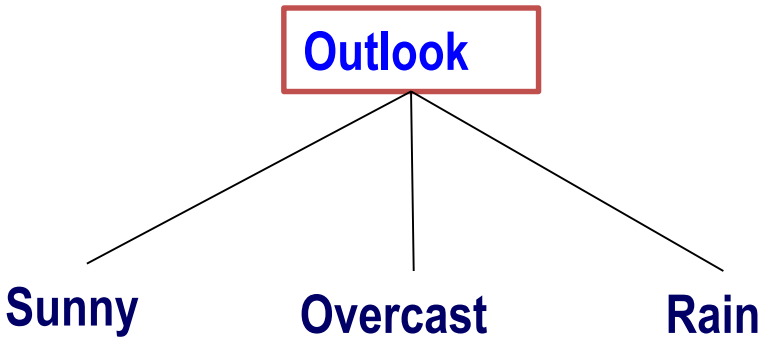
Gain(S, Temperature*)* = 0.029

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	*No*
Day2	Sunny	Hot	High	Strong	*No*
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	*No*
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	*No*
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	*No*

Example

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Pick Outlook as the root



Gain(S, Outlook) = 0.246

Gain(S, Temperature) = 0.029

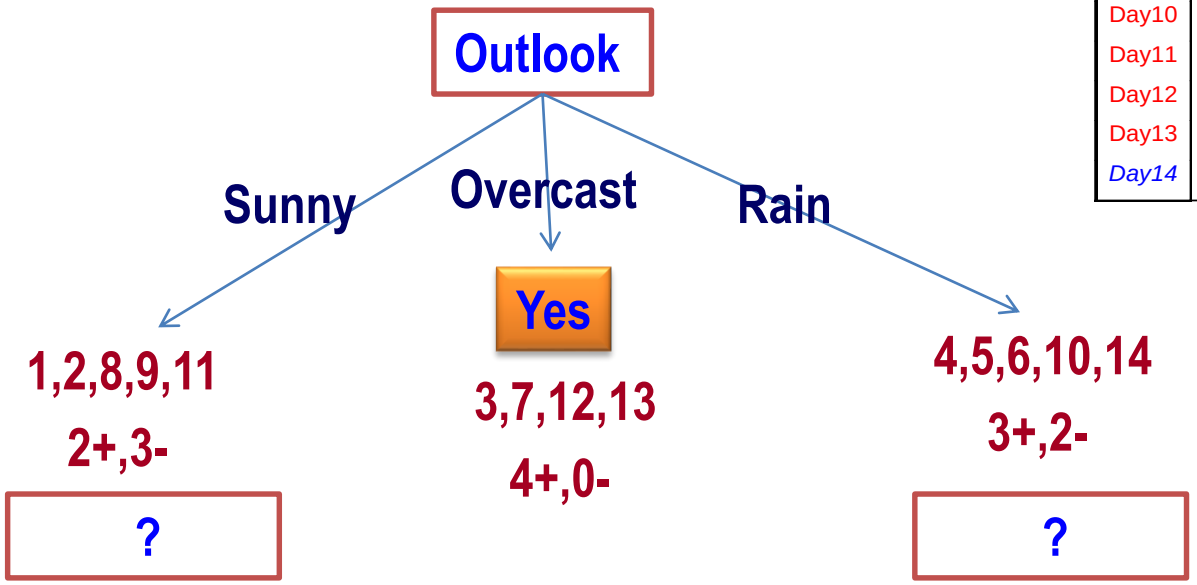
Gain(S, Humidity) = 0.151

Gain(S, Wind) = 0.048

Example

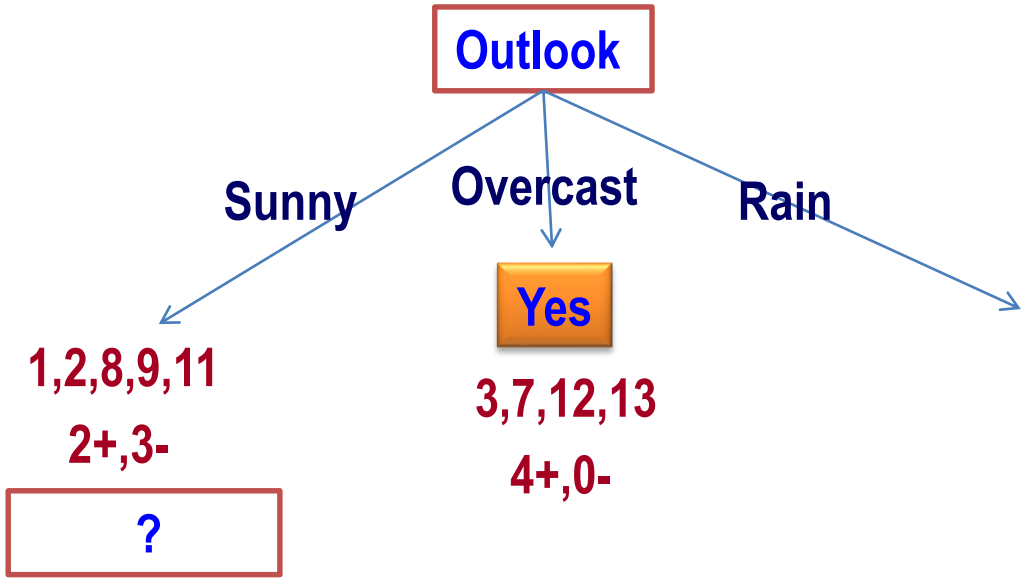
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

Pick Outlook as the root



Continue until: Every attribute is included in path, or, all examples in the leaf have same label

Example



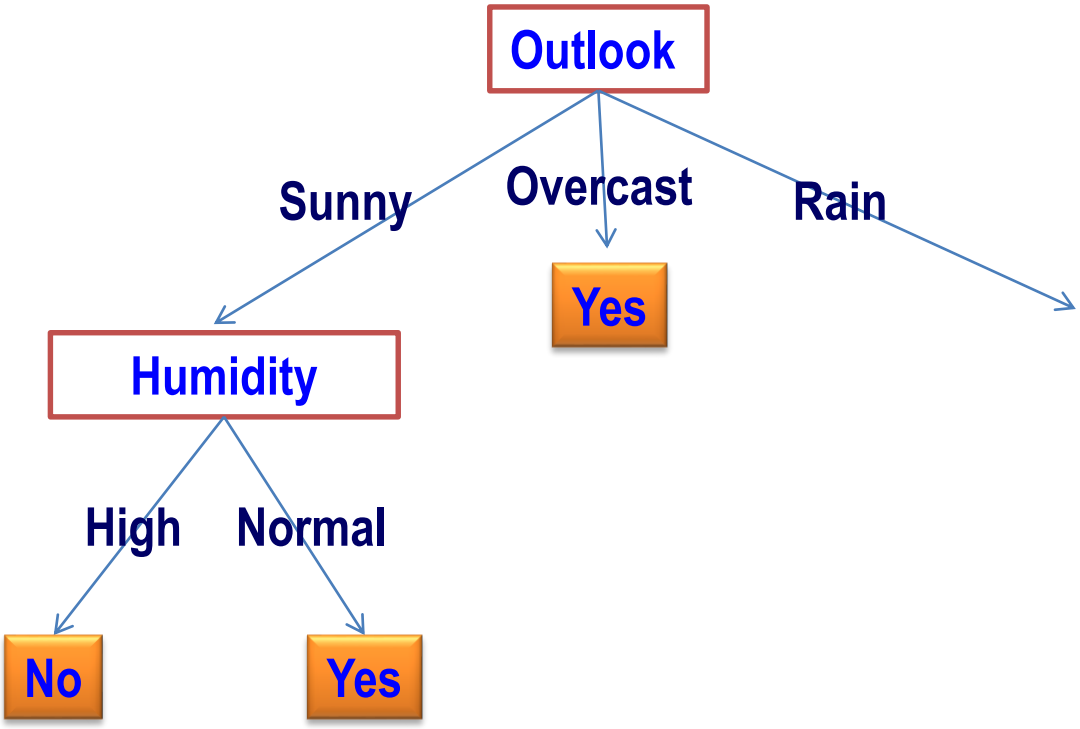
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No

1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Gain (S_{sunny}, Humidity) = .97-(3/5) * 0-(2/5) * 0 = .97
Gain (S_{sunny}, Temp) = .97- 0-(2/5) *1 = .57
Gain (S_{sunny}, Wind) = .97-(2/5) *1 - (3/5) *.92 = .02

Example

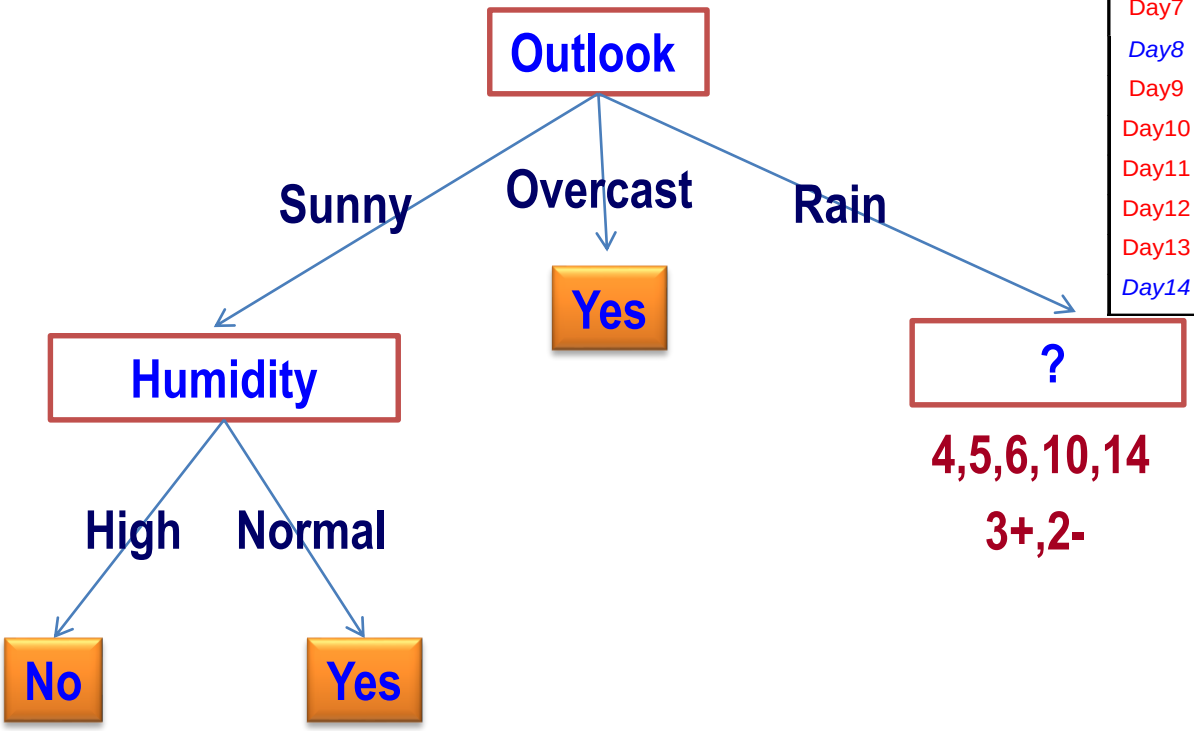
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No



Gain (S_{sunny}, Humidity) = .97-(3/5) * 0-(2/5) * 0 = .97
Gain (S_{sunny}, Temp) = .97- 0-(2/5) *1 = .57
Gain (S_{sunny}, Wind) = .97-(2/5) *1 - (3/5) *.92 = .02

Example

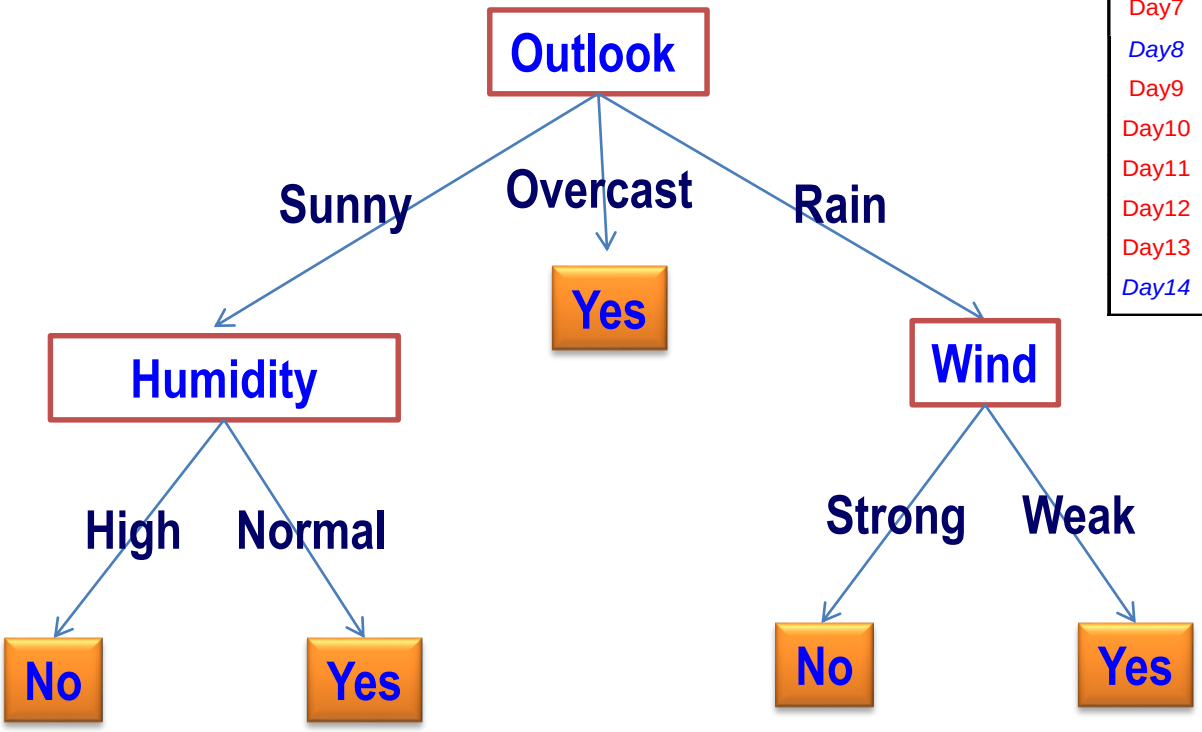
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No



Gain (S_{rain}, Humidity) =
Gain (S_{rain}, Temp) =
Gain (S_{rain}, Wind) =

Example

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
Day1	Sunny	Hot	High	Weak	No
Day2	Sunny	Hot	High	Strong	No
Day3	Overcast	Hot	High	Weak	Yes
Day4	Rain	Mild	High	Weak	Yes
Day5	Rain	Cool	Normal	Weak	Yes
Day6	Rain	Cool	Normal	Strong	No
Day7	Overcast	Cool	Normal	Strong	Yes
Day8	Sunny	Mild	High	Weak	No
Day9	Sunny	Cool	Normal	Weak	Yes
Day10	Rain	Mild	Normal	Weak	Yes
Day11	Sunny	Mild	Normal	Strong	Yes
Day12	Overcast	Mild	High	Strong	Yes
Day13	Overcast	Hot	Normal	Weak	Yes
Day14	Rain	Mild	High	Strong	No



University of Nottingham
UK | CHINA | MALAYSIA

COMP3055
Machine Learning

Topic 10 – Data Processing and Representation

Ying Weng
2024 Autumn

PCA

- Principal Component Analysis (PCA) is one of the most often used dimensionality reduction technique.

PCA Goals

- We wish to explain/summarize the underlying variance-covariance structure of a large set of variables through a few linear combinations of these variables.

PCA Procedures

- Get data (example)
- Step 1
  - Subtract the mean (example)
- Step 2
  - Calculate the covariance matrix
- Step 3
  - Calculate the eigenvectors and eigenvalues of the covariance matrix

A 2D Numerical Example

PCA Example – Data

- Original data

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

STEP 1

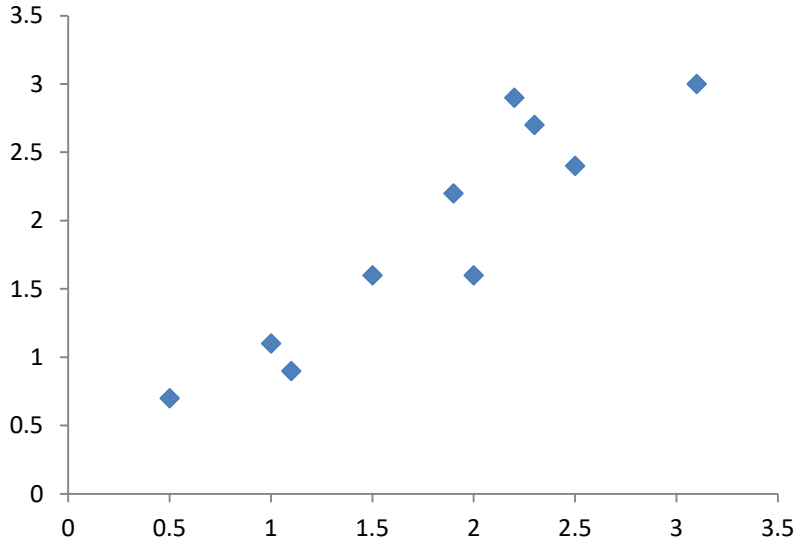
- Subtract the mean
- from each of the data dimensions. All the x values have average (x) subtracted and y values have average (y) subtracted from them. This produces a data set whose mean is zero.
- Subtracting the mean makes variance and covariance calculation easier by simplifying their equations. The variance and co-variance values are not affected by the mean value.

STEP 1

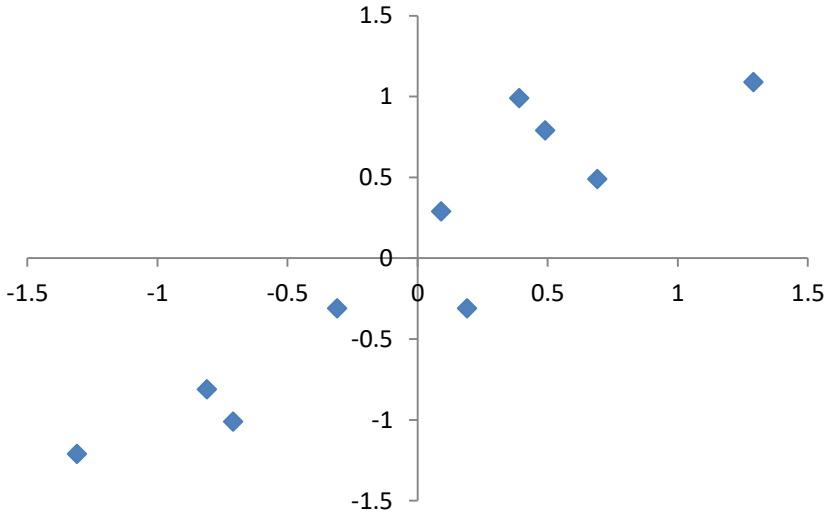
- Zero-mean data

0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

STEP 1



Original



Zero-mean

STEP 2

- Calculate the covariance matrix

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

- since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

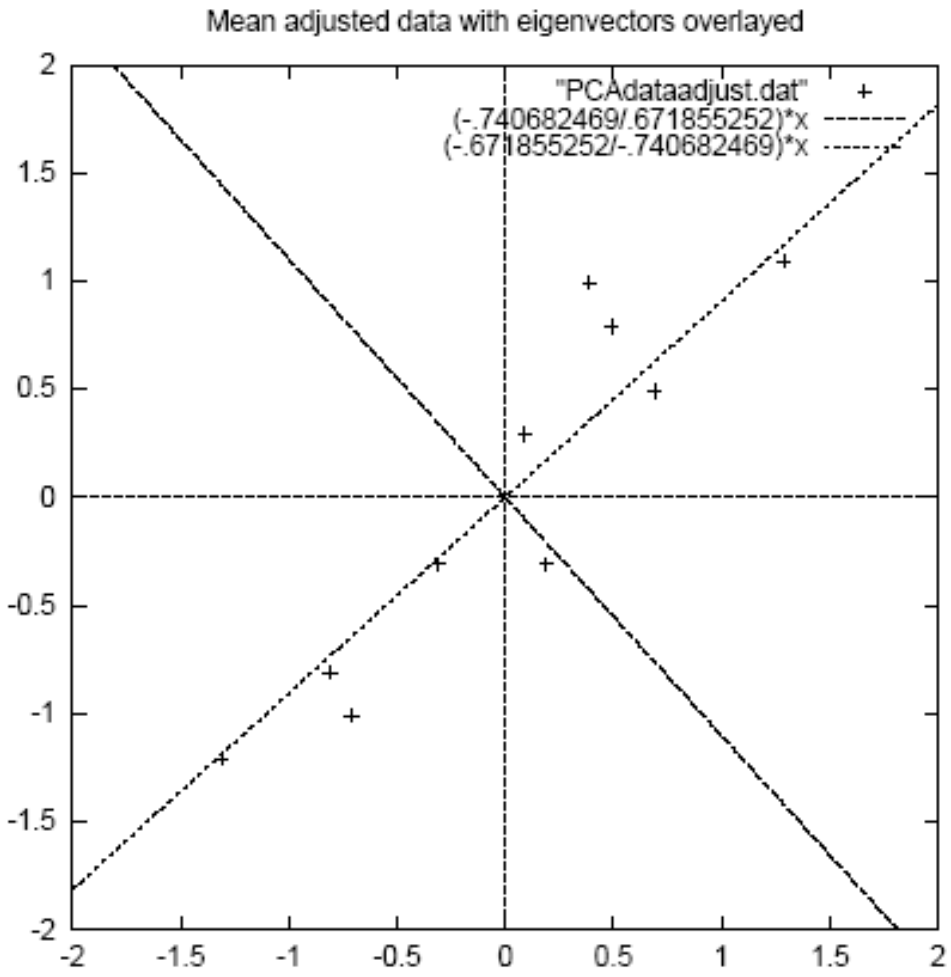
STEP 3

- Calculate the eigenvectors and eigenvalues of the covariance matrix

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

STEP 3



- eigenvectors are plotted as diagonal dotted lines on the plot.
- Note they are perpendicular to each other.
- Note one of the eigenvectors goes through the middle of the points, like drawing a line of best fit.
- The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlayed on top.

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

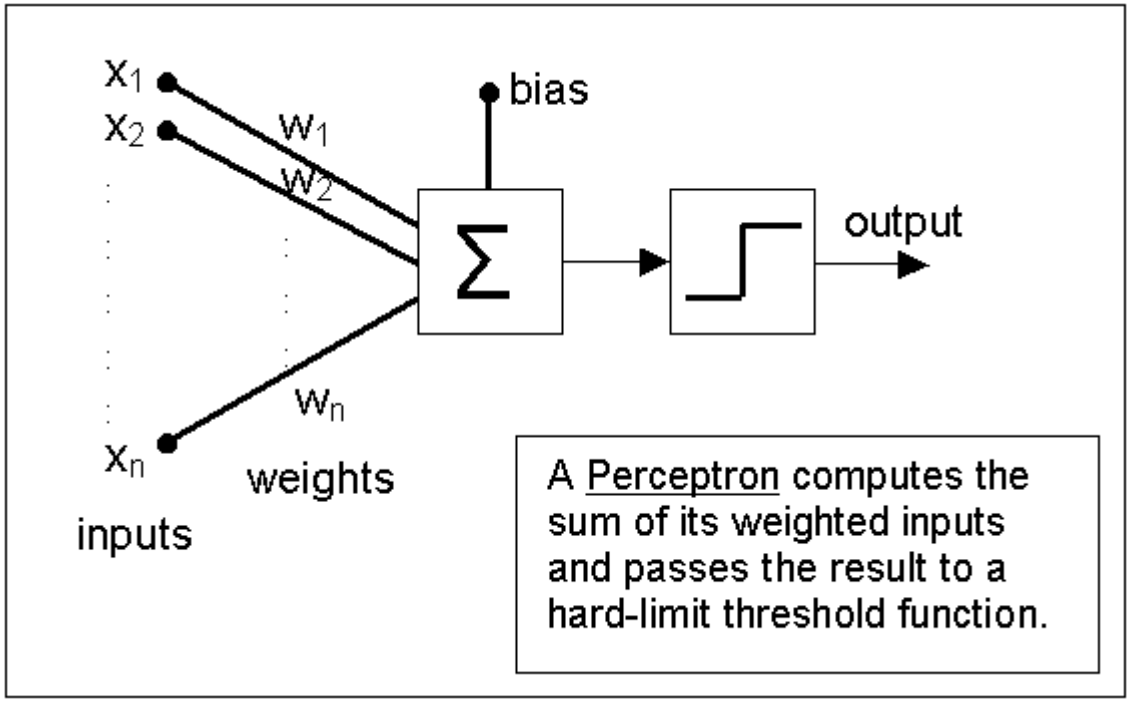
Topic 11 – Perceptron, ADLINE, and Delta Rule

Zheng Lu

2024 Autumn

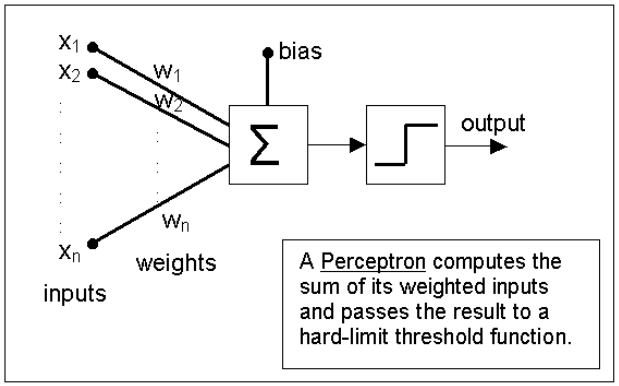
Perceptron - Basic

Perceptron is a type of Artificial Neural Network (ANN)



Perceptron - Operation

Perceptron takes a vector of real-valued inputs, calculates a *linear combination* of these inputs. Then it outputs 1 if the result is greater than some threshold and -1 otherwise.



$$R = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = w_0 + \sum_{i=1}^{n} w_i x_i$$

$$output = o = sign(R) = \begin{cases} +1, & if\ R > 0 \\ -1, & otherwise \end{cases}$$

Perceptron – Decision Surface

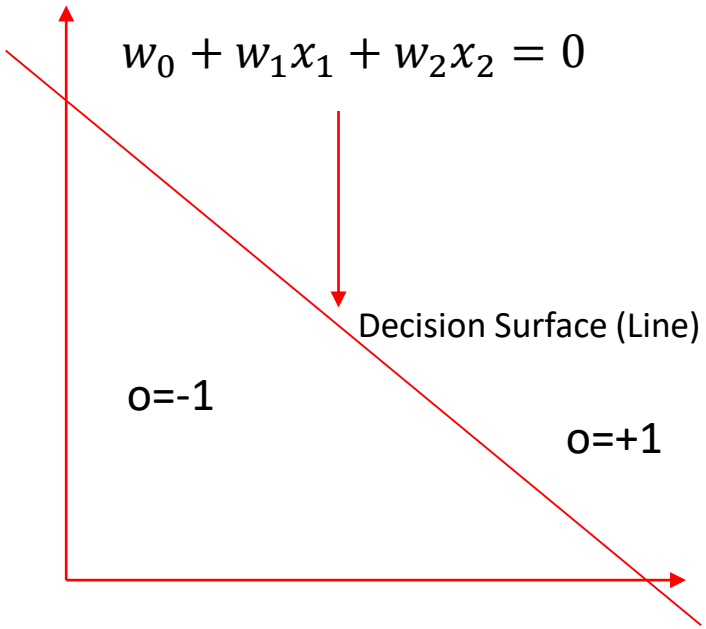
- Perceptron can be regarded as representing a hyperplane decision surface in the n-dimensional **feature space** of instances.

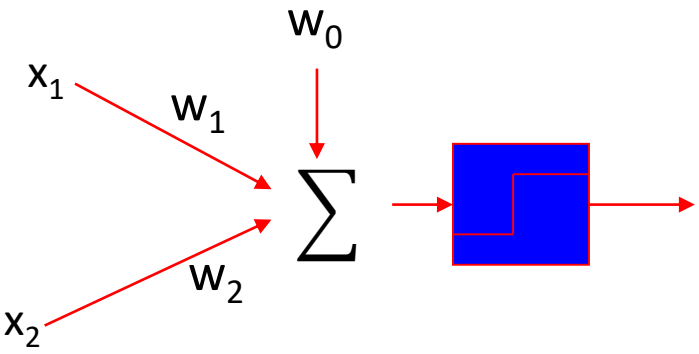
- The perceptron outputs a 1 for instances lying on one side of the hyperplane and a -1 for instances lying on the other side.

- This hyperplane is called the **Decision Surface**.

Perceptron – Decision Surface

In 2-dimensional space:

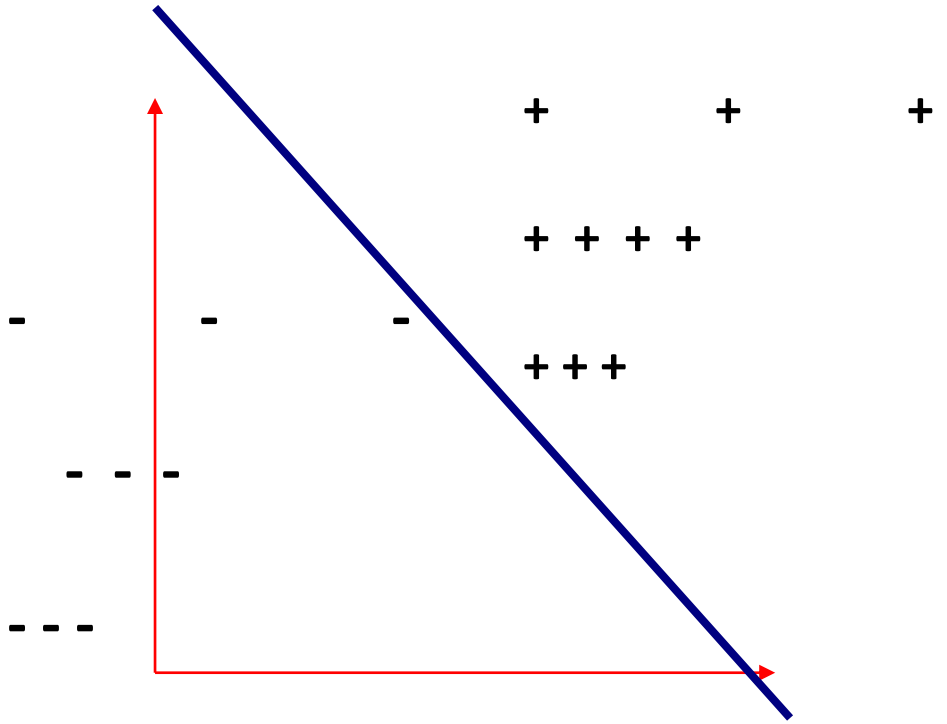




Perceptron – Representation Power

- The Decision Surface is linear.

- Perceptron can only solve **Linearly Separable Problems**.

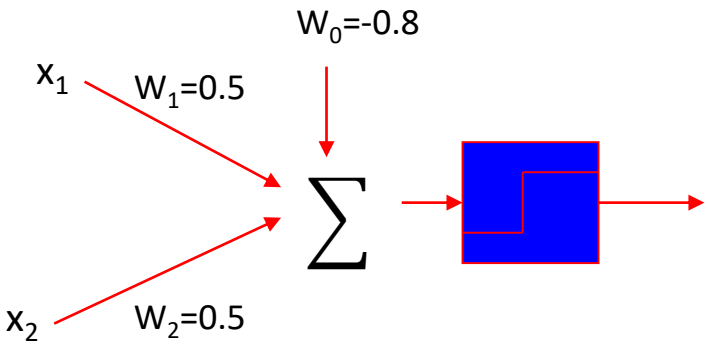


Perceptron – Representation Power

Can represent many Boolean functions, assuming Boolean values of 1 (true) and -1 (false).

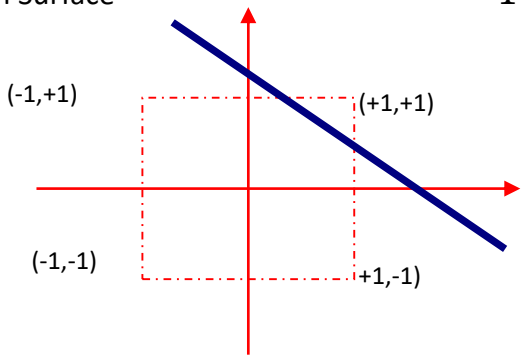
AND

x1	x2	D
-1	-1	-1
-1	+1	-1
+1	-1	-1
+1	+1	+1



Decision Surface

$$-0.8 + 0.5x_1 + 0.5x_2 = 0$$

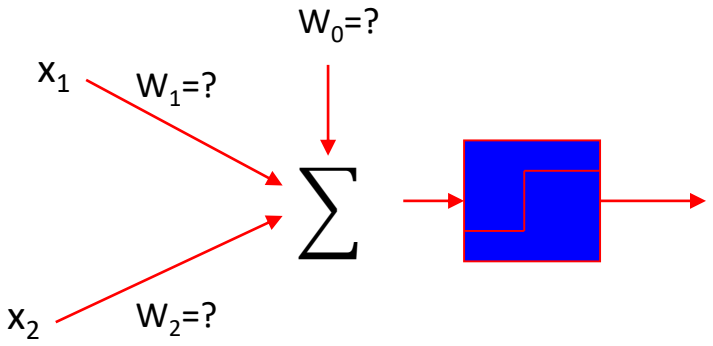


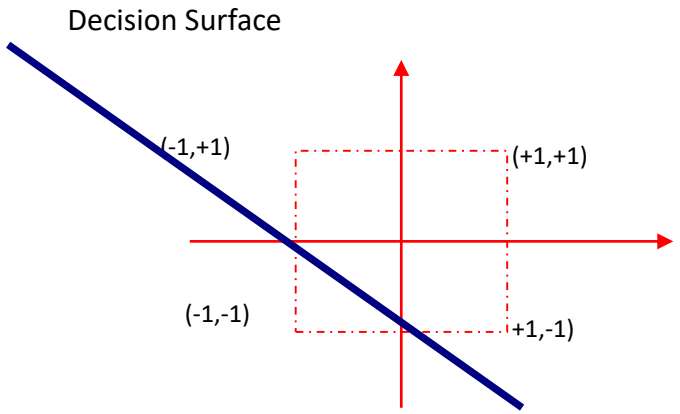
Perceptron – Representation Power

Can represent many Boolean functions, assuming Boolean values of 1 (true) and -1 (false).

OR

x1	x2	D
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	+1



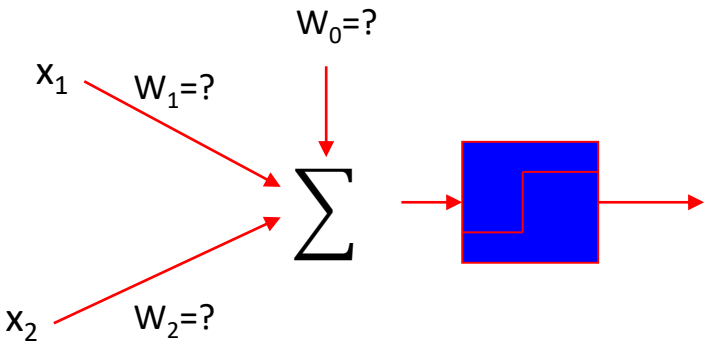


Perceptron – Representation Power

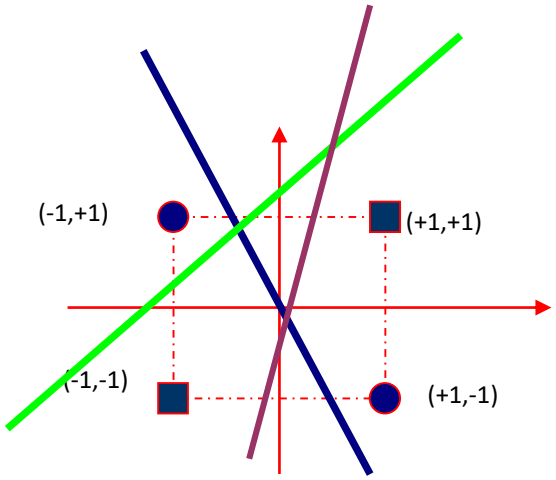
Some problems are linearly non-separable.

XOR

x1	x2	D
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1



Decision Surface:
It doesn't matter where you place the line (decision surface). It is impossible to separate the space such that on one side we have o = 1 and on the other we have o = -1.



Perceptron Cannot Solve such Problem!

Perceptron – Training Algorithm

Training sample pairs (X, d), where X is the input vector, d is the input vector's classification (+1 or -1) is iteratively presented to the network for training, *one at a time*, until the process converges.

Perceptron – Training Algorithm

The Procedure is as follows:

1. Set the weights to small random values, e.g., in the range (-1, 1)

2. Present X, and calculate

$$R = \sum_{i=0}^{n} w_i x_i \quad , \quad o = sign(R) = \begin{cases} +1, & if\ R > 0 \\ -1, & otherwise \end{cases}$$

let x_0=1

1. Update the weights

$$w_i \leftarrow w_i + \eta(d - o)x_i, i = 0,1, \dots, n$$

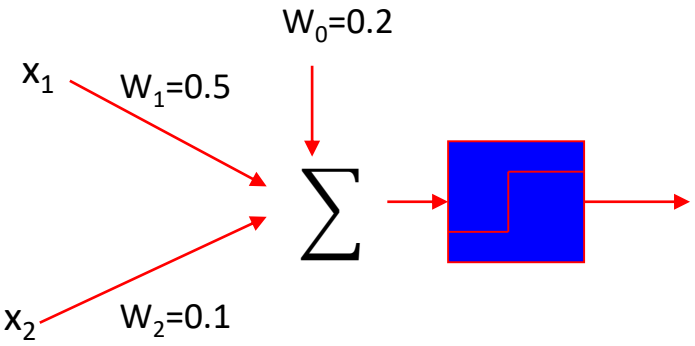
$$0 < \eta < 1, is\ the\ training\ rate$$

2. Repeat by going to step 2

Perceptron – Training Algorithm

Example

x1	x2	D
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	+1



$$w_i \leftarrow w_i + \eta(d - o)x_i, i = 1,2, \dots, n$$

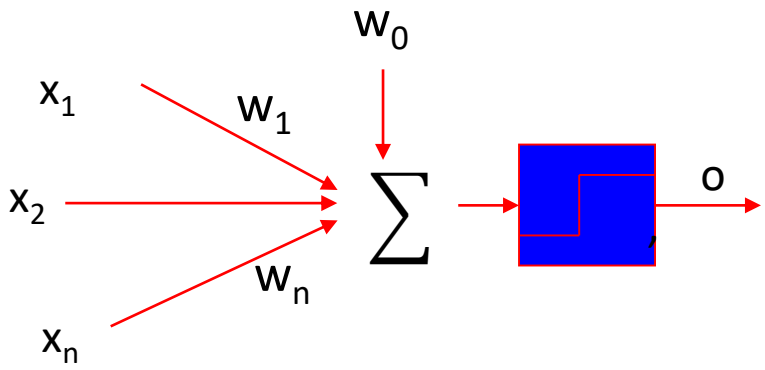
Perceptron – Training Algorithm

Convergence Theorem

The perceptron training rule will converge (finding a weight vector correctly classifies all training samples) within a finite number of iterations, **provided the training examples are linearly separable** and provided a sufficiently small η is used.

Adaptive Linear Element (ADLINE)

Perceptron

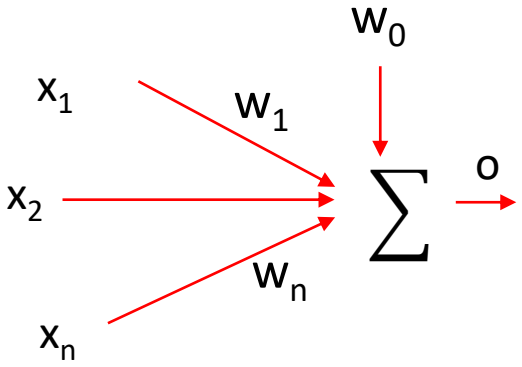


$$R = w_0 + \sum_{i=1}^{n} w_i x_i$$

$$o = sign(R) = \begin{cases} +1, & if\ R > 0 \\ -1, & otherwise \end{cases}$$

VS

ADLINE



$$o = w_0 + \sum_{i=1}^{n} w_i x_i$$

ADLINE VS Perceptron

- When the problem is not linearly separable, perceptron will fail to converge.

- ADLINE can overcome this difficulty by finding a best fit approximation to the target.

The Gradient Descent Rule

Gradient descent training procedure

- Initialise w_i to small vales, e.g., in the range of $(-1, 1)$, choose a learning rate, e.g., $\eta = 0.2$

- Until the termination condition is met, Do
  - For all training sample pair $(X(k), d(k))$, input the instance $X(k)$ and compute

$$\delta_i = -\sum_{k=1}^{K} (d(k) - o(k)) x_i(k)$$

  - For each weight w_i, Do

$$w_i \leftarrow w_i - \eta \delta_i$$

Batch Mode:

gradients accumulated over **ALL** samples first

Then update the weights

Stochastic (Incremental) Gradient Descent

Also called online mode, Least Mean Square (LMS), Widrow-Hoff, and Delta Rule

- Initialise w_i to small vales, e.g., in the range of $(-1, 1)$, choose a learning rate, e.g., $\eta = 0.01$ (should be smaller than batch mode)

- Until the termination condition is met, Do
  - For EACH training sample pair $(X(k), d(k))$, compute

$$\delta_i = -(d(k) - o(k))x_i(k)$$

  - For each weight w_i, Do

$$w_i \leftarrow w_i - \eta\delta_i$$

Online Mode:

Calculate gradient for **EACH (or a SUBSET of)**samples

Then update the weights

Training Iterations, Epochs

- Training is an iterative process; training samples will have to be used repeatedly for training.

- Assuming we have K training samples *[(X(k), d(k)), k=1, 2, ..., K]*; then an epoch is the presentation of all K sample for training once.

  – First epoch: Present training samples: *(X(1), d(1)), (X(2), d(2)), ... (X(K), d(K))*
  – Second epoch: Present training samples: *(X(K), d(K)), (X(K-1), d(K-1)), ... (X(1), d(1))*

  – Note the order of the training sample presentation between epochs can (and should normally) be different.

- Normally, training will take many epochs to complete.

Termination of Training

To terminate training, there are normally two ways

- When a pre-set number of training epochs is reached.
- When the error is smaller than a pre-set value.

$$E(W) \equiv \frac{1}{2}\sum_{k=1}^{K}(d(k) - o(k))^2$$

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

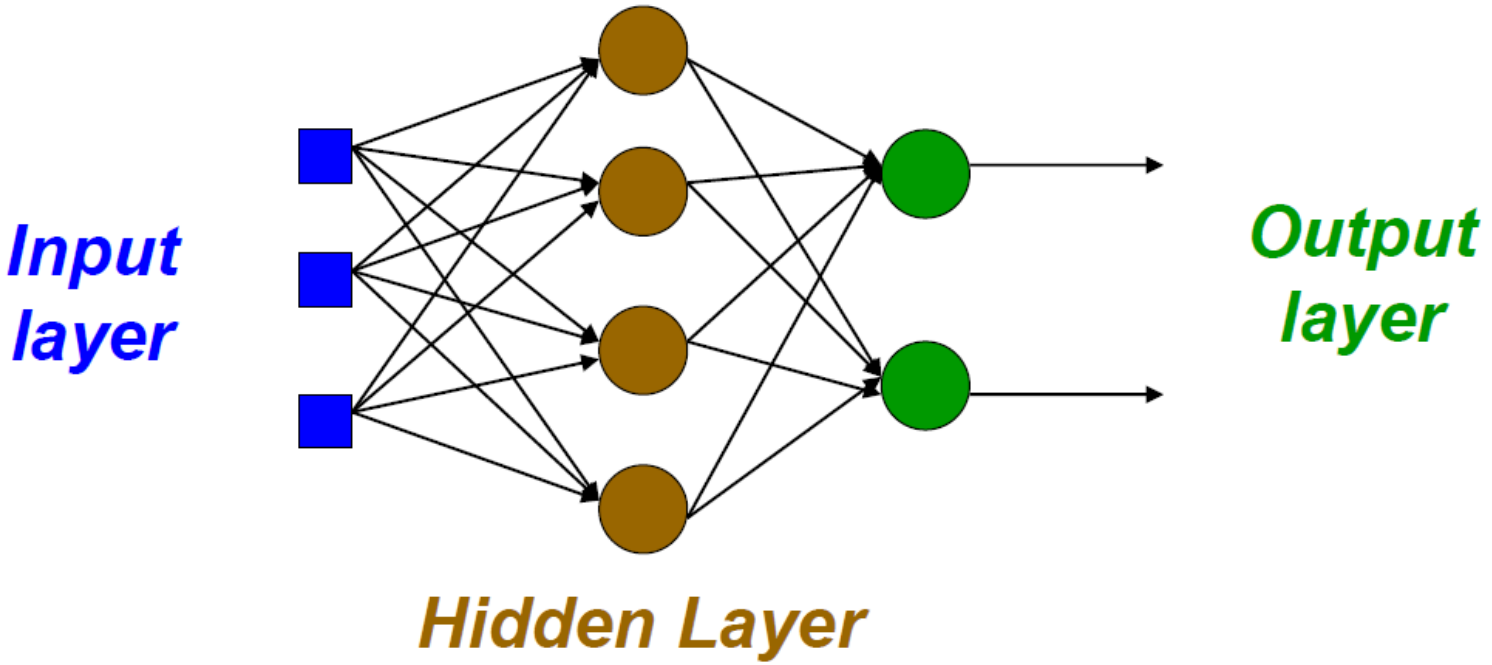
Topic 12 – Multilayer Perceptrons

Zheng Lu

2023 Autumn

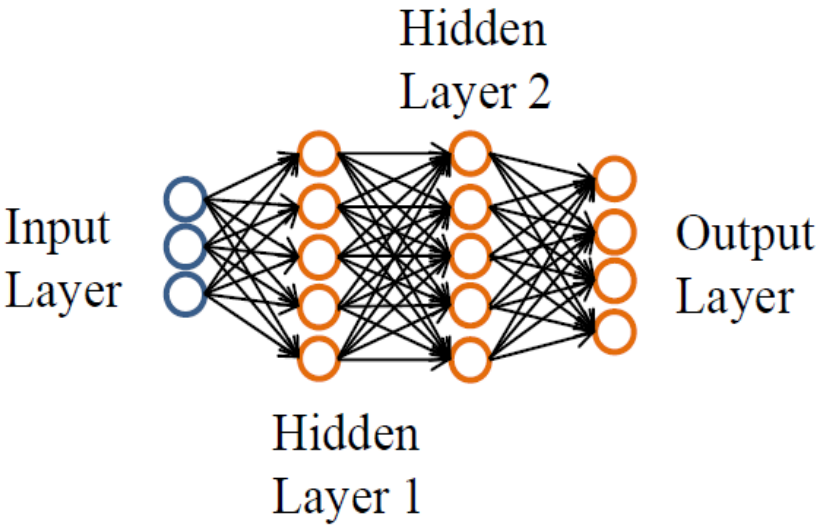
Multilayer Perceptron (MLP)

- A more general network architecture: between the input and output layers there are hidden layers
- Hidden nodes do not directly receive inputs nor send outputs to the external environment
- Fully connected between layers

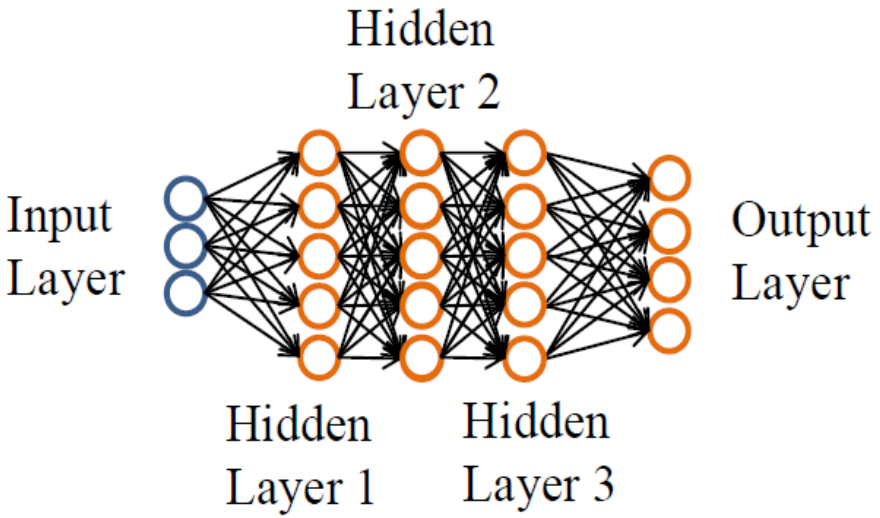


MLP Architecture

- Feedforward network: connections between the nodes do not form a cycle
- MLP usually interconnected in a feed-forward way
- The input layer does not count as a layer



3-layer feed-forward network



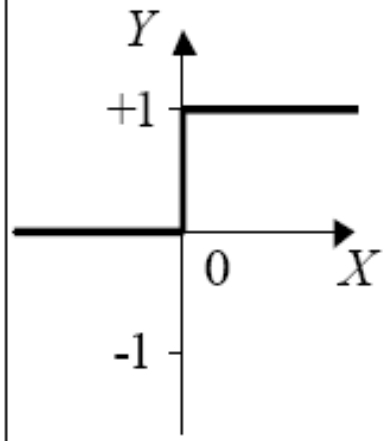
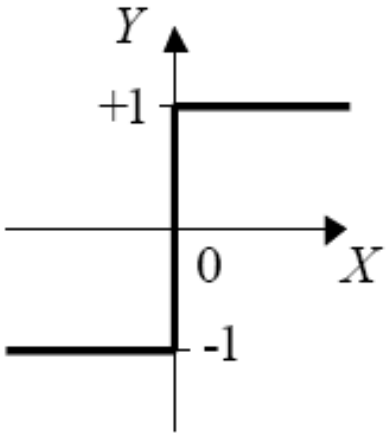
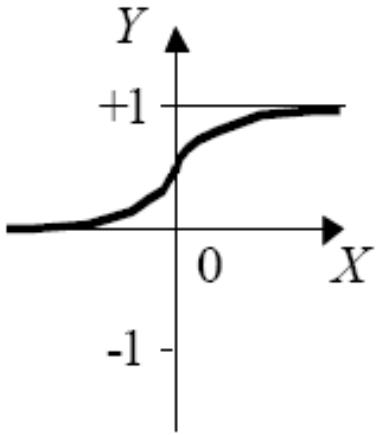
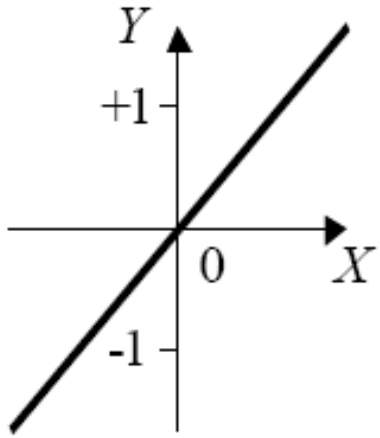
4-layer feed-forward network

Activation Function

- Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and **determines whether it should be activated ("fired") or not, based on whether each neuron's input is relevant for the model's prediction**. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

- The activation function can be considered as a mathematical "gate" in **between** the **input feeding the current neuron** and **its output going to the next layer**.

Activation Function

Step function	Sign function	Sigmoid function	Linear function
			
$Y^{step} = \begin{cases} 1, \text{ if } X \geq 0 \\ 0, \text{if } X < 0 \end{cases}$	$Y^{sign} = \begin{cases} +1, \text{if } X \geq 0 \\ -1, \text{if } X < 0 \end{cases}$	$Y^{sigmoid} = \frac{1}{1+e^{-X}}$	$Y^{linear} = X$

Back-propagation Summary

- Gradient descent over entire network weight vector.
- Will find a local, not necessarily a global error minimum.
- In practice, it often works well (can run multiple times).
- Minimizes error over all training samples.
  - Will it generalize to subsequent examples? i.e., will the trained network perform well on data outside the training sample.
- Training can take thousands of iterations.
- After training, use the network is fast.

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

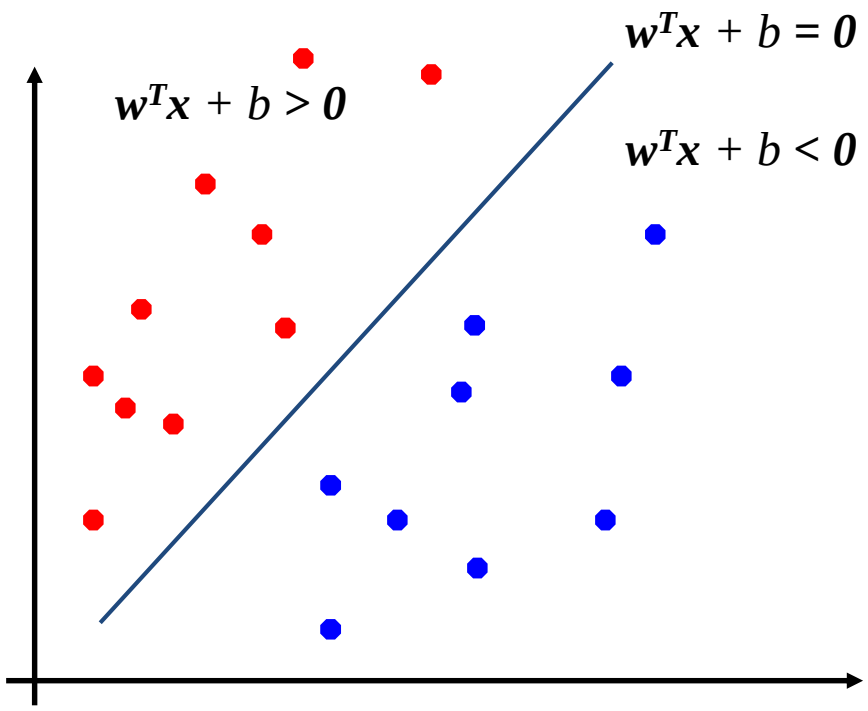
Topic 13 – Support Vector Machine

Zheng Lu

2024 Autumn

Perceptron Revisited: Linear Separators

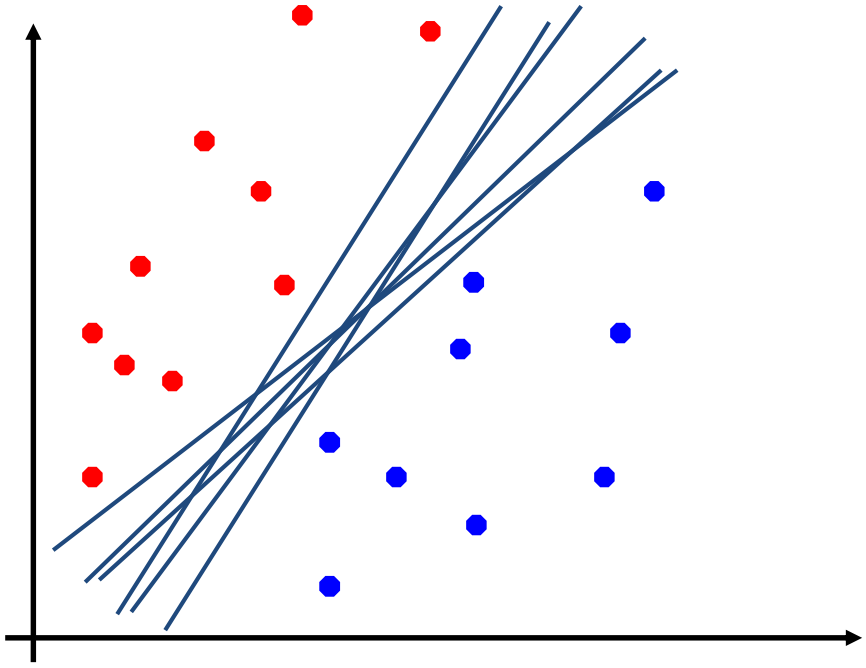
Binary classification can be viewed as the task of separating two classes in feature space:



$$f(x) = sign(w^T x + b)$$

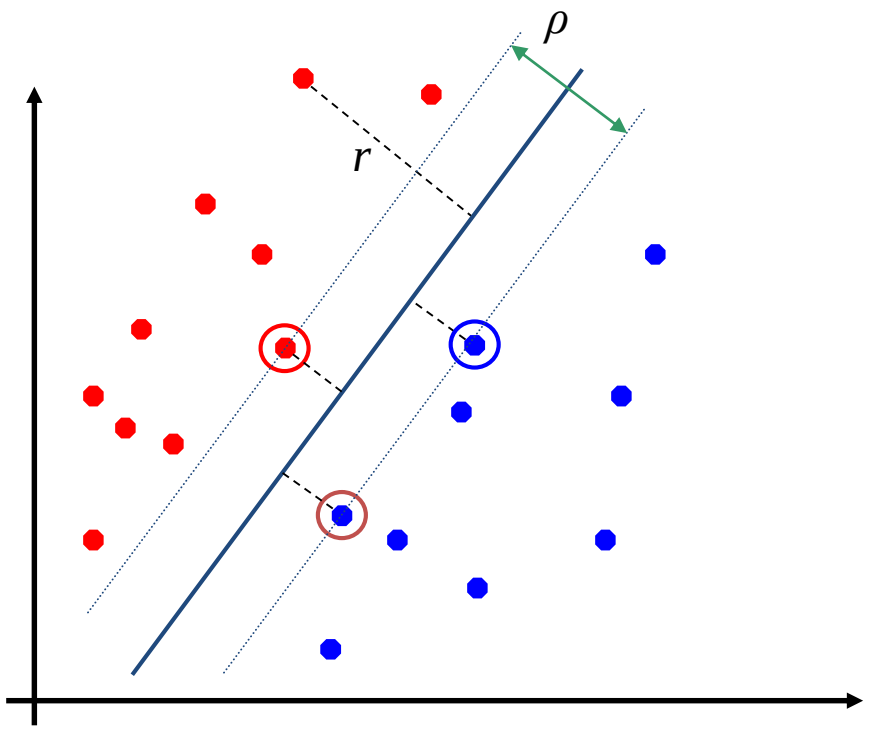
Linear Separator

Which of the linear separators is optimal?



Classification Margin

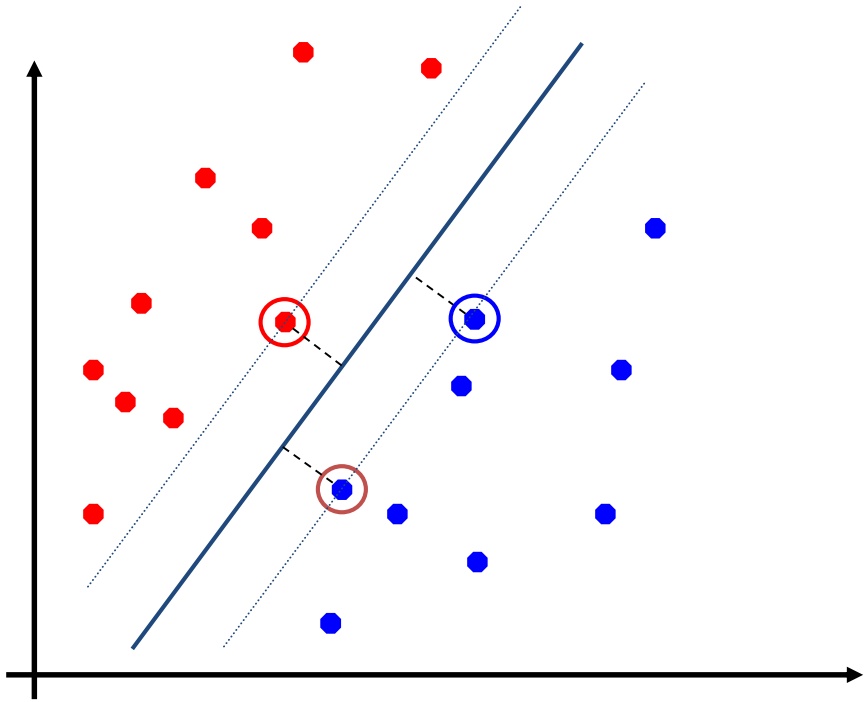
- Distance from example x_s to the separator is $r = \frac{|w^T x_s + b|}{\|w\|}$.
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.



Maximum Margin Classification

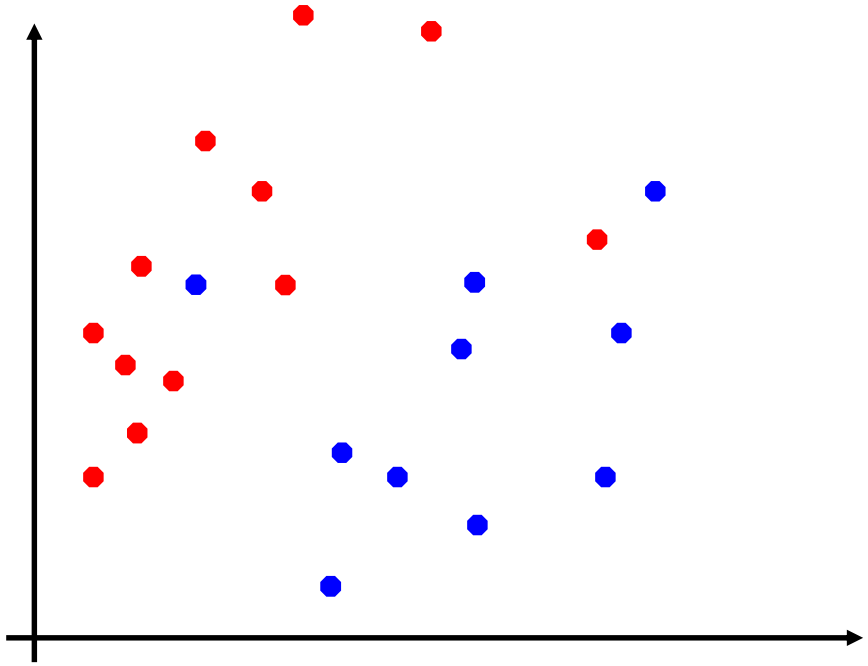
- Maximizing the margin is good according to intuition.

- Implying that only support vectors matter; other training examples are ignorable.



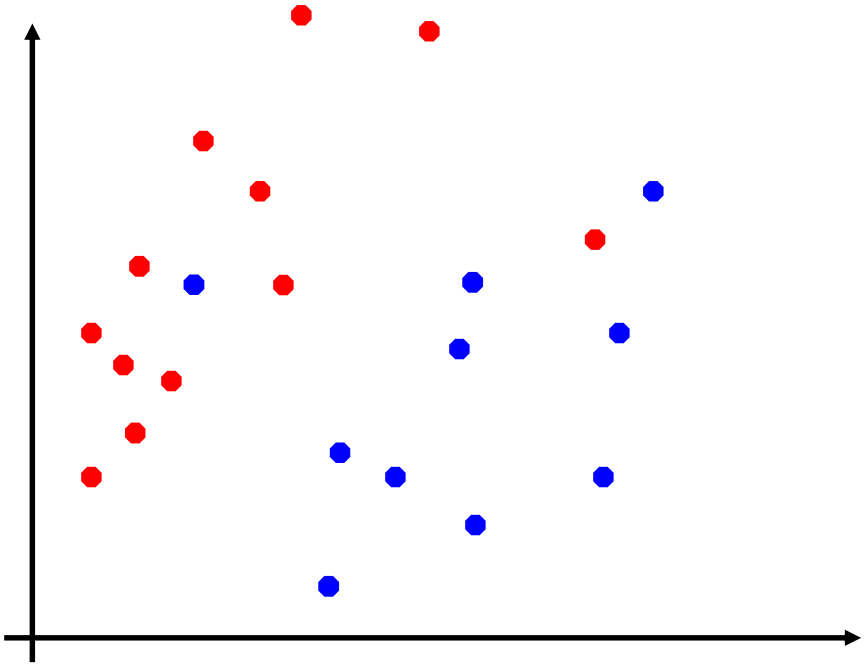
Soft Margin Classification

- What if the training set is not linearly separable?



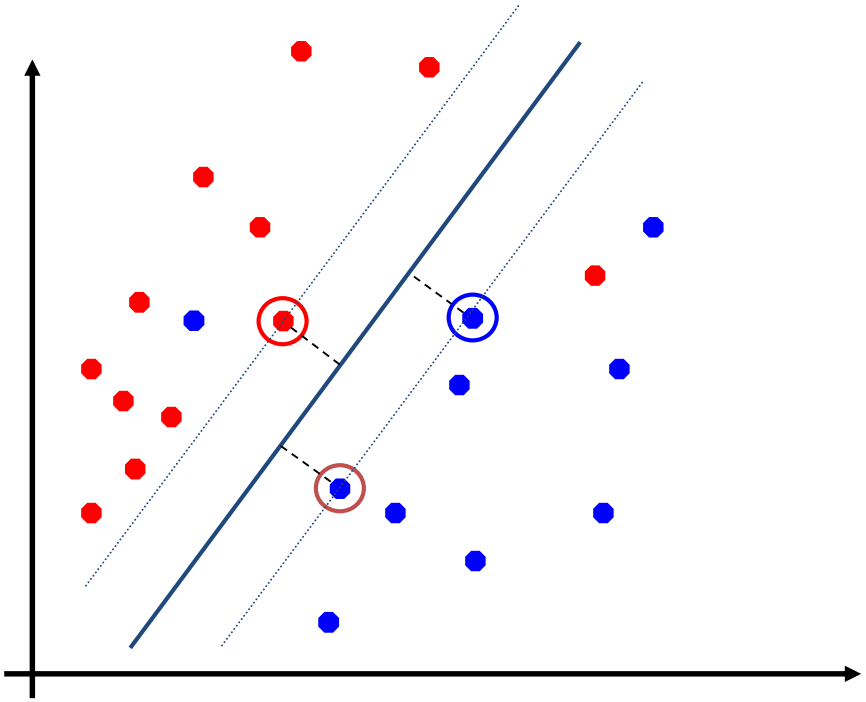
Soft Margin Classification

- What if the training set is not linearly separable?
- ***Slack variables*** ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called ***soft***.



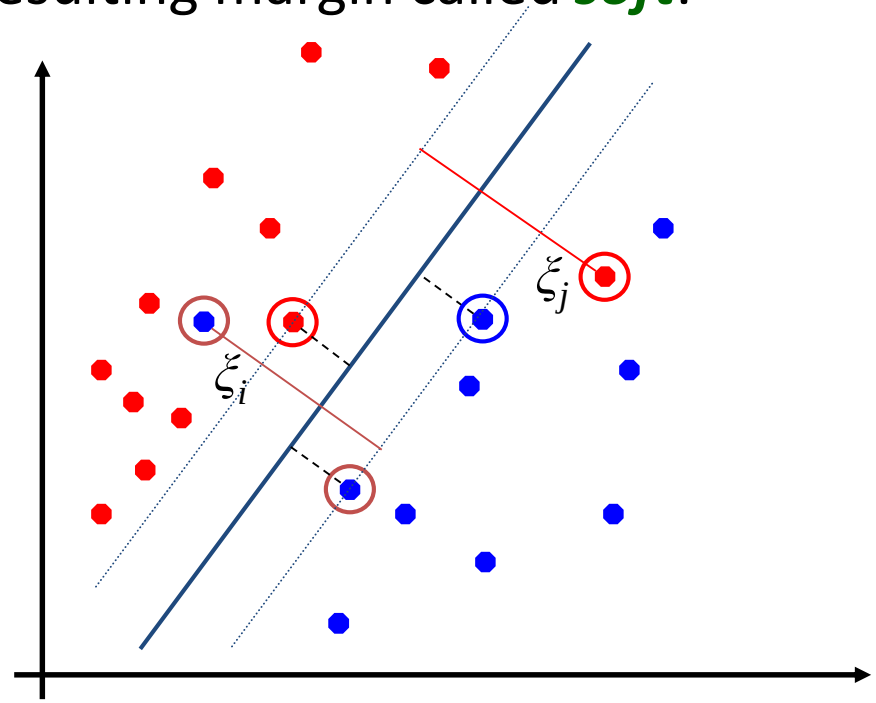
Soft Margin Classification

- What if the training set is not linearly separable?
- ***Slack variables*** ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called ***soft***.



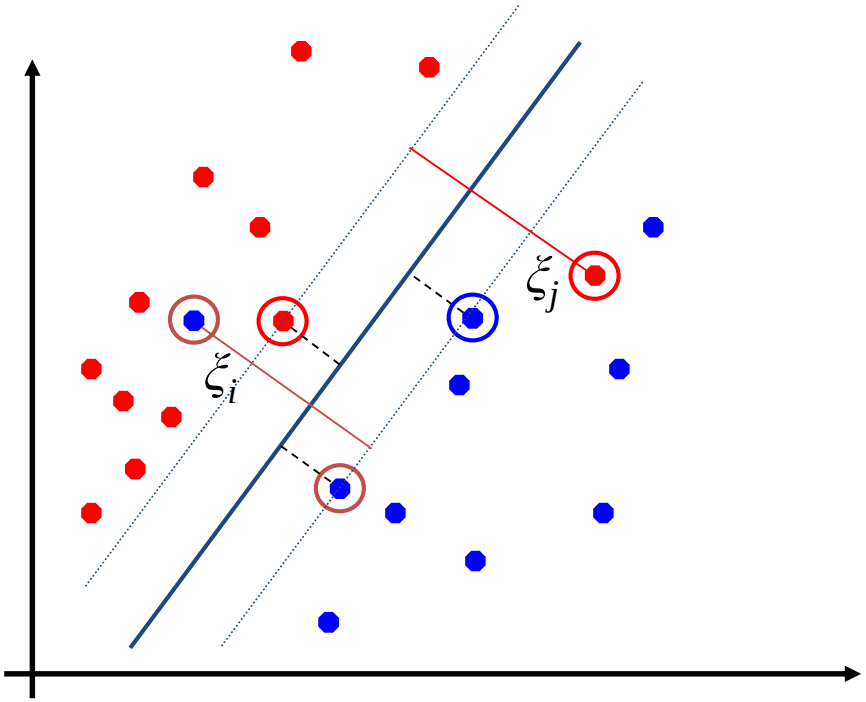
Soft Margin Classification

- What if the training set is not linearly separable?
- ***Slack variables*** ξ_i which measures the distance of the point to its marginal hyperplane if it is on the wrong side, otherwise 0, can be added to allow misclassification of difficult or noisy examples, resulting margin called ***soft***.



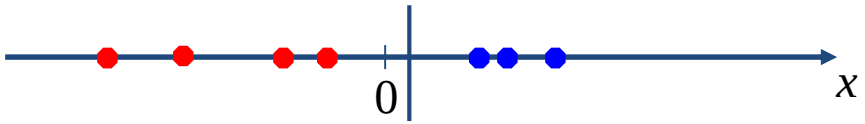
Soft Margin Classification

- Applying Soft Margin, SVM tolerates a few dots to get misclassified and tries to balance the trade-off between finding a line that maximizes the margin and minimizes the misclassification.



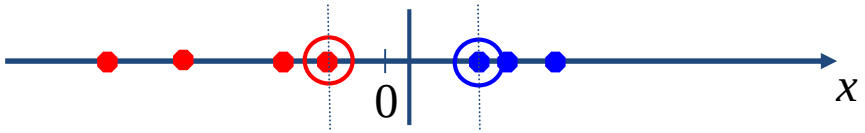
Non-Linear SVM

- Datasets that are linearly separable with some noise work out great:



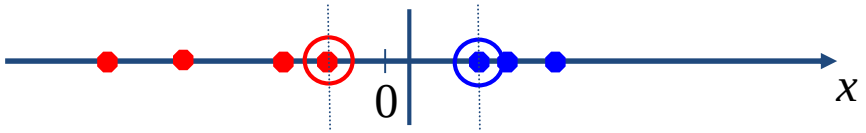
Non-Linear SVM

- Datasets that are linearly separable with some noise work out great:



Non-Linear SVM

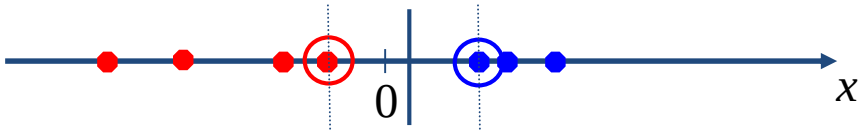
- Datasets that are linearly separable with some noise work out great:



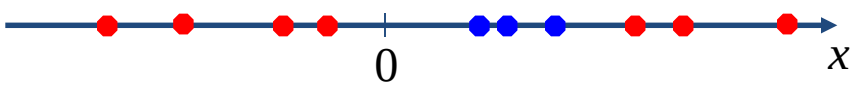
- But what are we going to do if the dataset is just too hard?

Non-Linear SVM

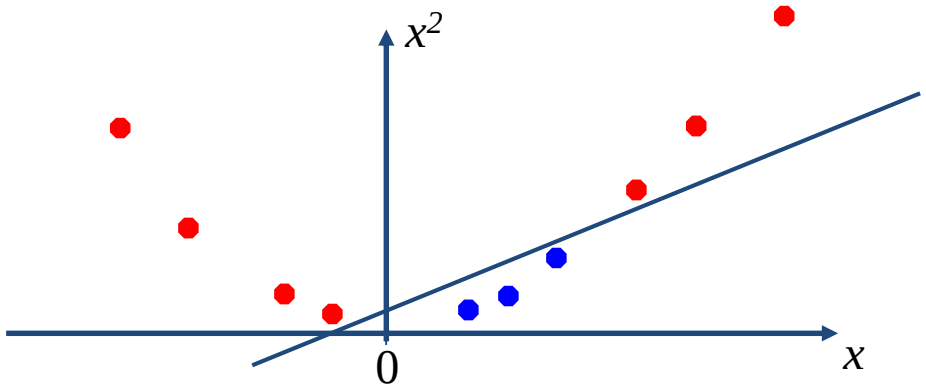
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

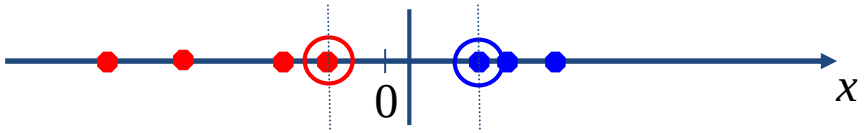


- How about… mapping data to a higher-dimensional space:

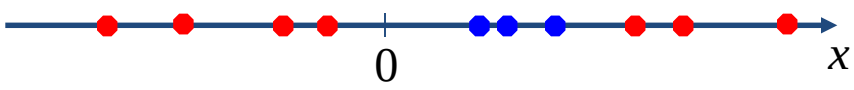


Non-Linear SVM

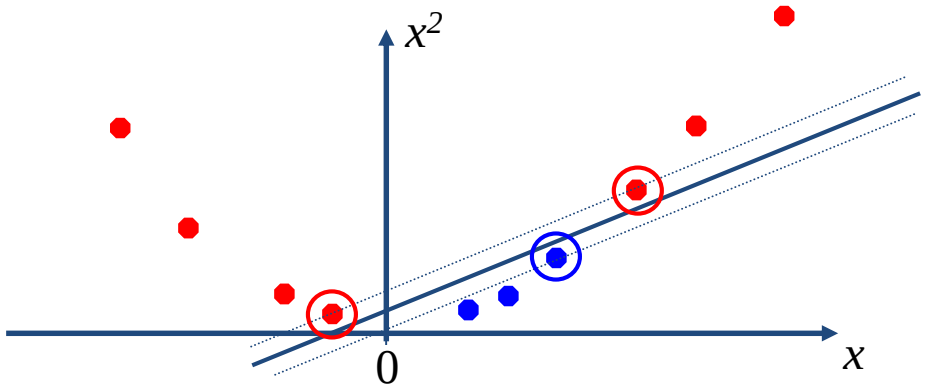
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

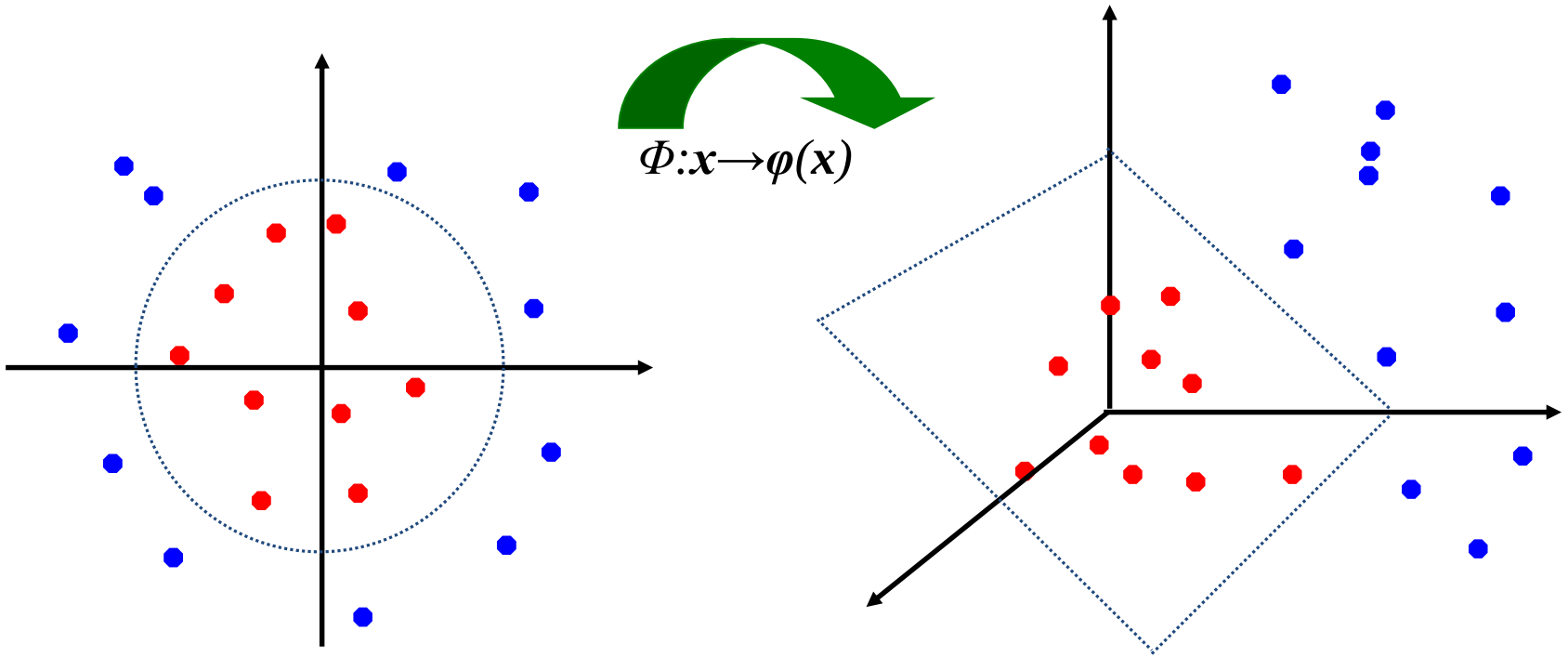


- How about… mapping data to a higher-dimensional space:



Non-Linear SVM: Feature Spaces

- **General idea**: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

- The linear classifier relies on inner product between vectors $K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{x}_i^T\boldsymbol{x}_j$
- If every data point is mapped into high-dimensional space via some transformation $\Phi:\mathbf{x}\rightarrow\varphi(\mathbf{x})$, the inner product becomes:

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)= \boldsymbol{\varphi}(\boldsymbol{x}_i)^T\boldsymbol{\varphi}(\boldsymbol{x}_j)$$

- A ***kernel function*** is a function that is equivalent to an inner product in some feature space.
- Thus, a kernel function ***implicitly*** maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\varphi}(\boldsymbol{x})$ explicitly).

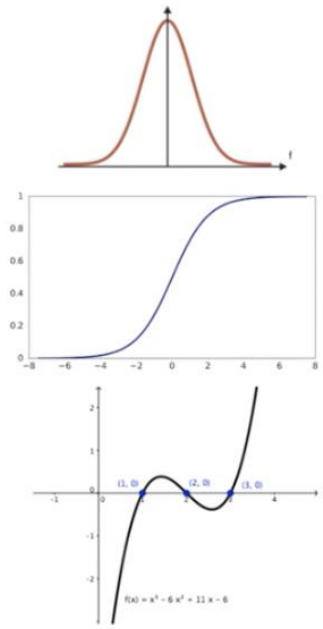
Examples of Kernel Function

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^p; \text{ polynomial kernel.} \quad (1.22)$$

$$K(x_i, x_j) = e^{\frac{-1}{2\sigma^2}(x_i - x_j)^2}; \text{ Gaussian kernel; Special case of Radial Basis Function.}$$

$$K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}; \text{ RBF Kernel}$$

$$K(x_i, x_j) = \tanh(\eta\, x_i \cdot x_j + \nu); \text{ Sigmoid Kernel; Activation function for NN.}$$



Gaussian RBF Kernel $K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x} - \vec{l}^i\|^2}{2\sigma^2}}$

Sigmoid Kernel $K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$

Polynomial Kernel $K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$

Multi-Class Classification

- Some algorithms are designed for binary classification problems:
  - Logistic Regression
  - Perceptron
  - Support Vector Machines
- Instead, heuristic methods can be used to split a multi-class classification problem into multiple binary classification datasets and train a binary classification model each
  - One-vs-All (OVA)
  - One-vs-One (OvO)

One-vs-All (OVA)

- Every class is paired with the remaining classes
- The base classifier needs to produce a real-valued confidence score for its decision, rather than just a class label
- Making decisions means applying all classifiers to an unseen sample *x* and predicting the label *k* for which the corresponding classifier reports the highest confidence score

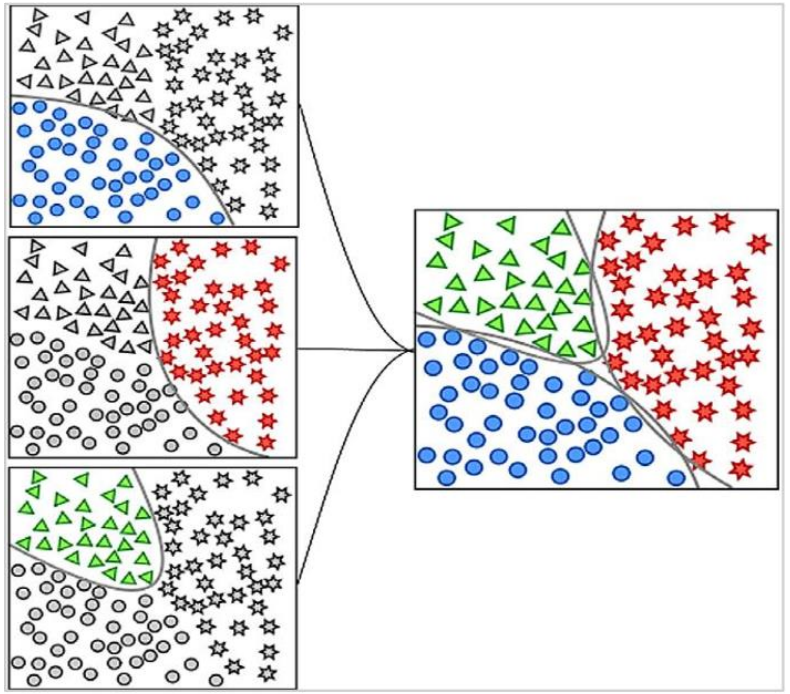


Figure 3. One vs all Strategy using 3 class problem

One-vs-One (OVO)

- Every class is paired with the every other class
- At prediction time, a voting scheme is applied: all classifiers are applied to an unseen sample and the class that got the highest number of "+1" predictions gets predicted by the combined classifier

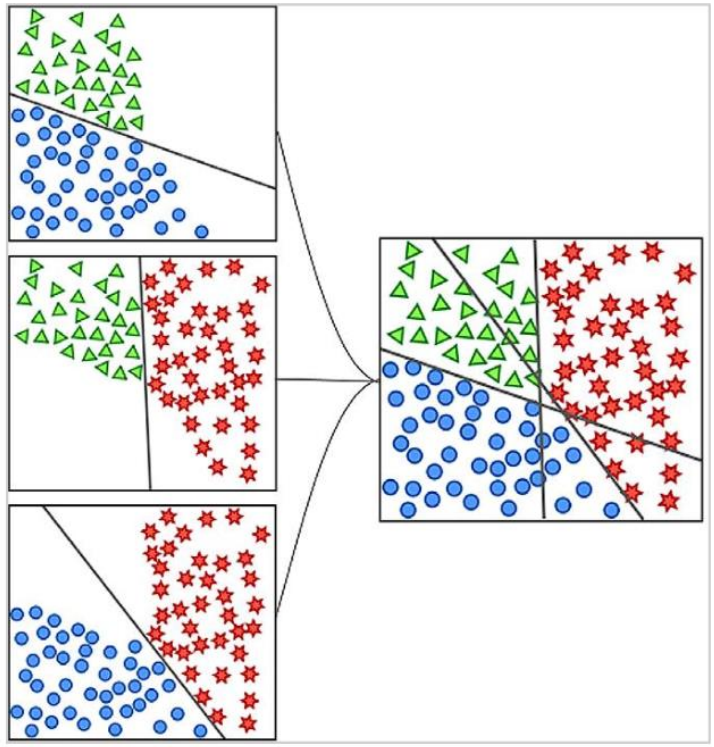


Figure 2. One vs one strategy using 3 class problem

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

Topic 14 – Deep Learning - Basics

Zheng Lu

2024 Autumn

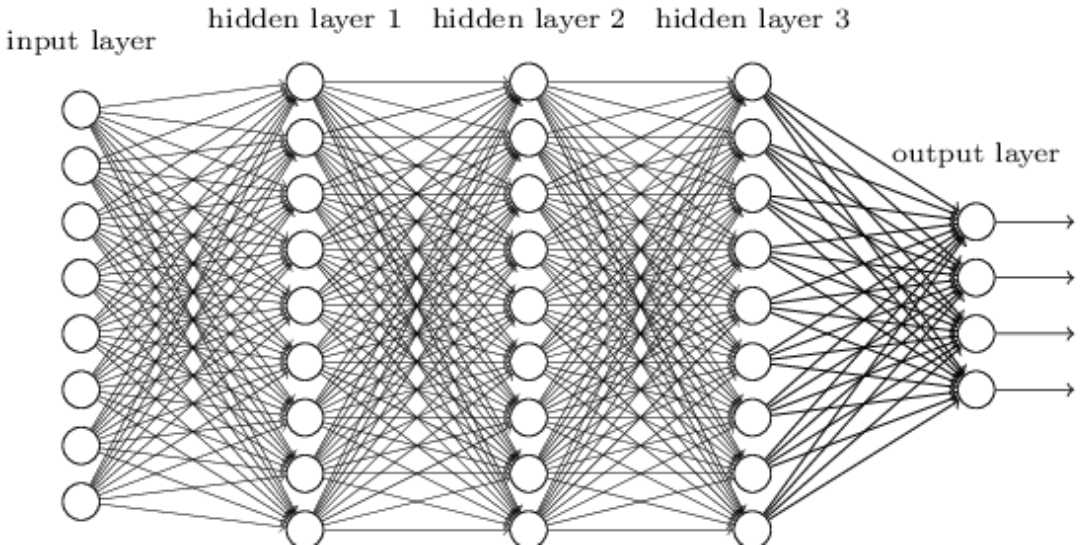
What is Deep Learning?

- Most machine learning methods work well because of human-designed input features or representations
  - Our job is to find the best features to send to learning techniques such as SVM.

- Machine learning becomes just optimizing weights to best make a final prediction.

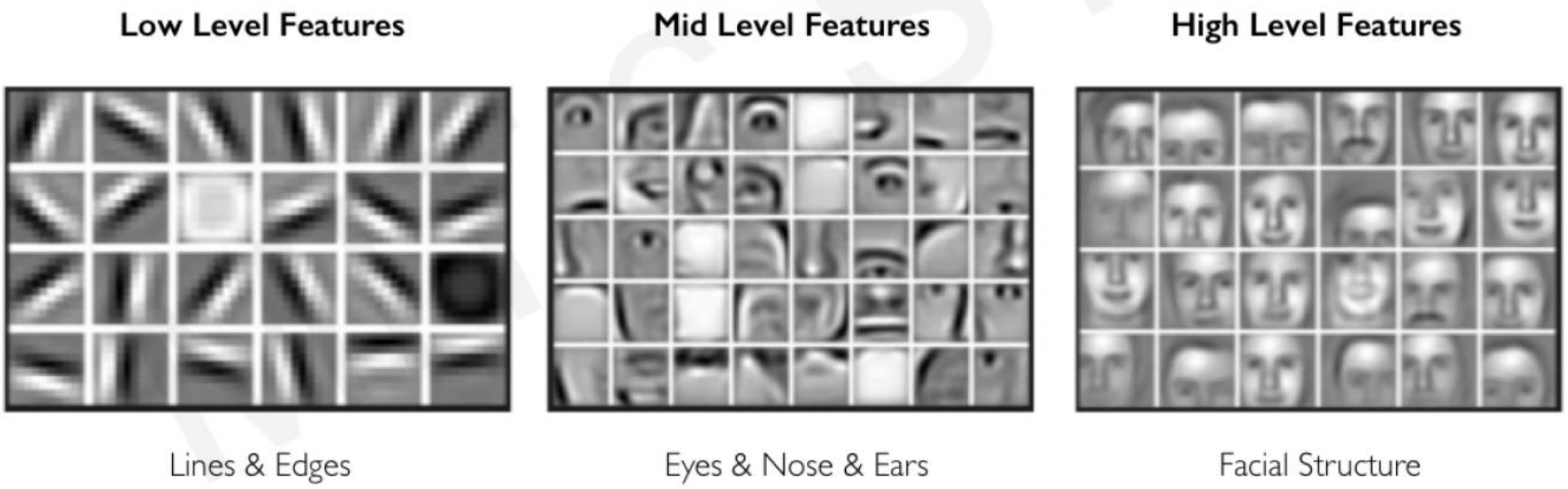
What is Deep Learning?

- In contrast to standard machine learning
- **Representation learning** attempts to automatically learn good features or representations
- **Deep learning algorithms** learn multiple levels of representations (here: hidden layer 1, 2, 3) and an output layer
- From "raw" inputs x (e.g. sound, pixels, characters, or words)
- **Neural networks** are the currently successful method for deep learning
- A.k.a. "**Differentiable Programming**"



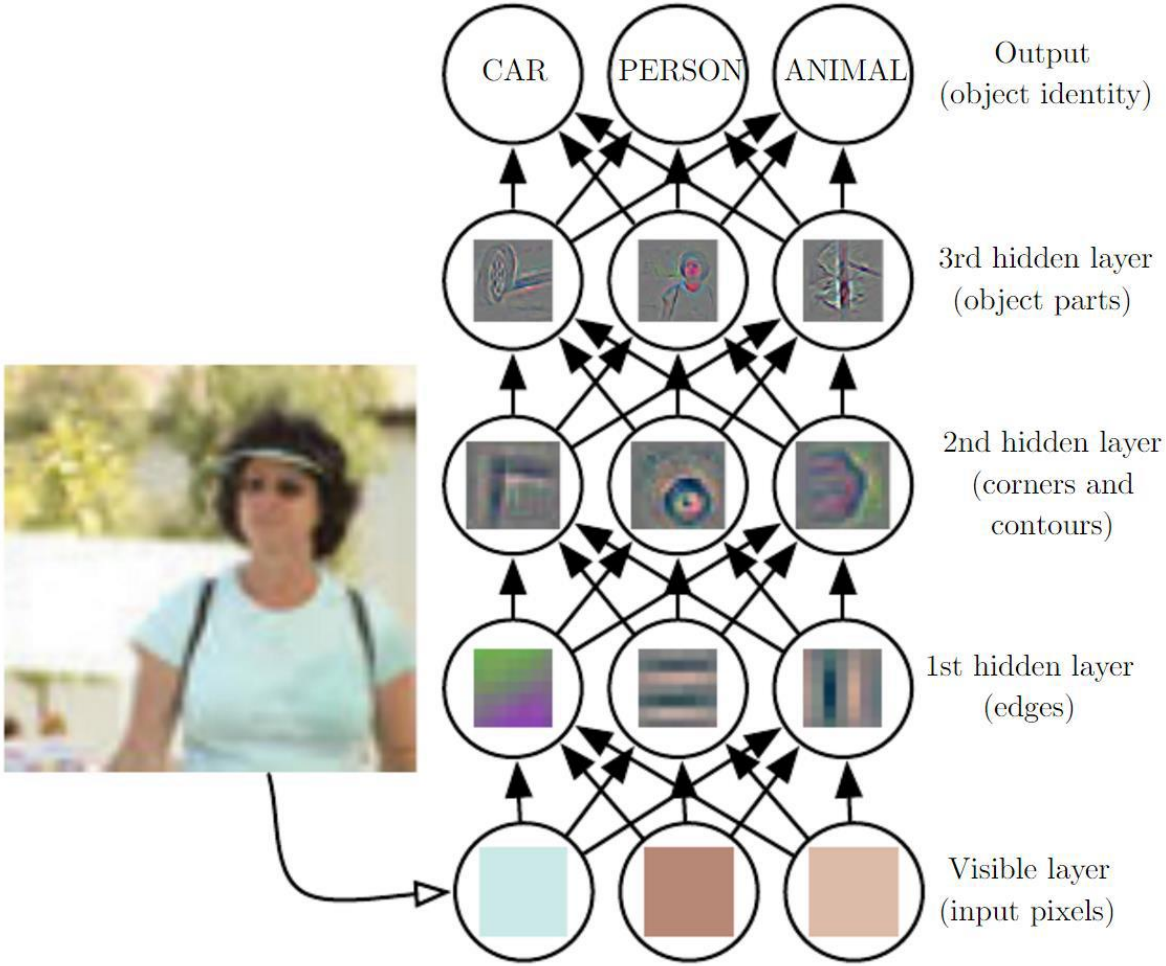
Why Deep Learning?

- To learn the underlying features directly



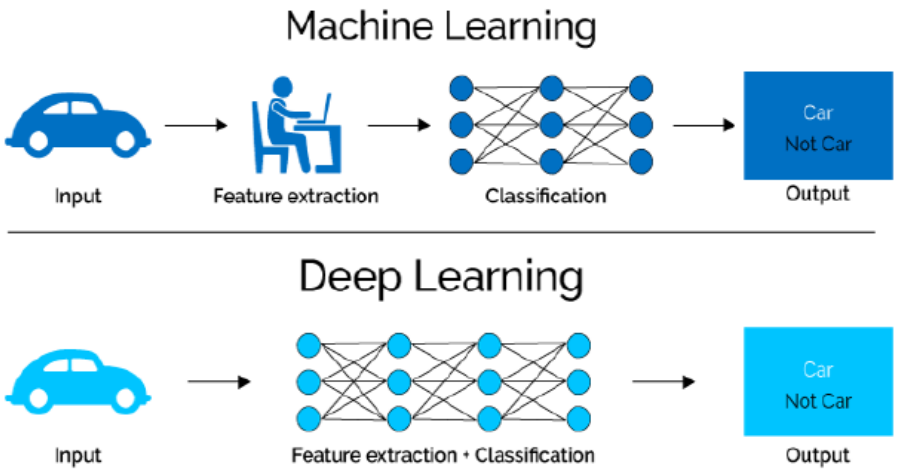
Why Deep Learning?

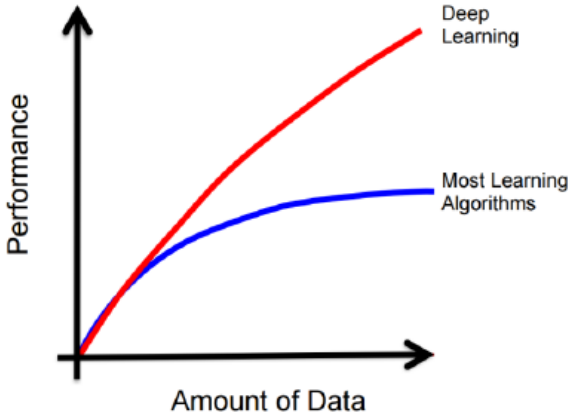
- To learn the underlying features directly



Why Deep Learning?

- Scalable machine learning

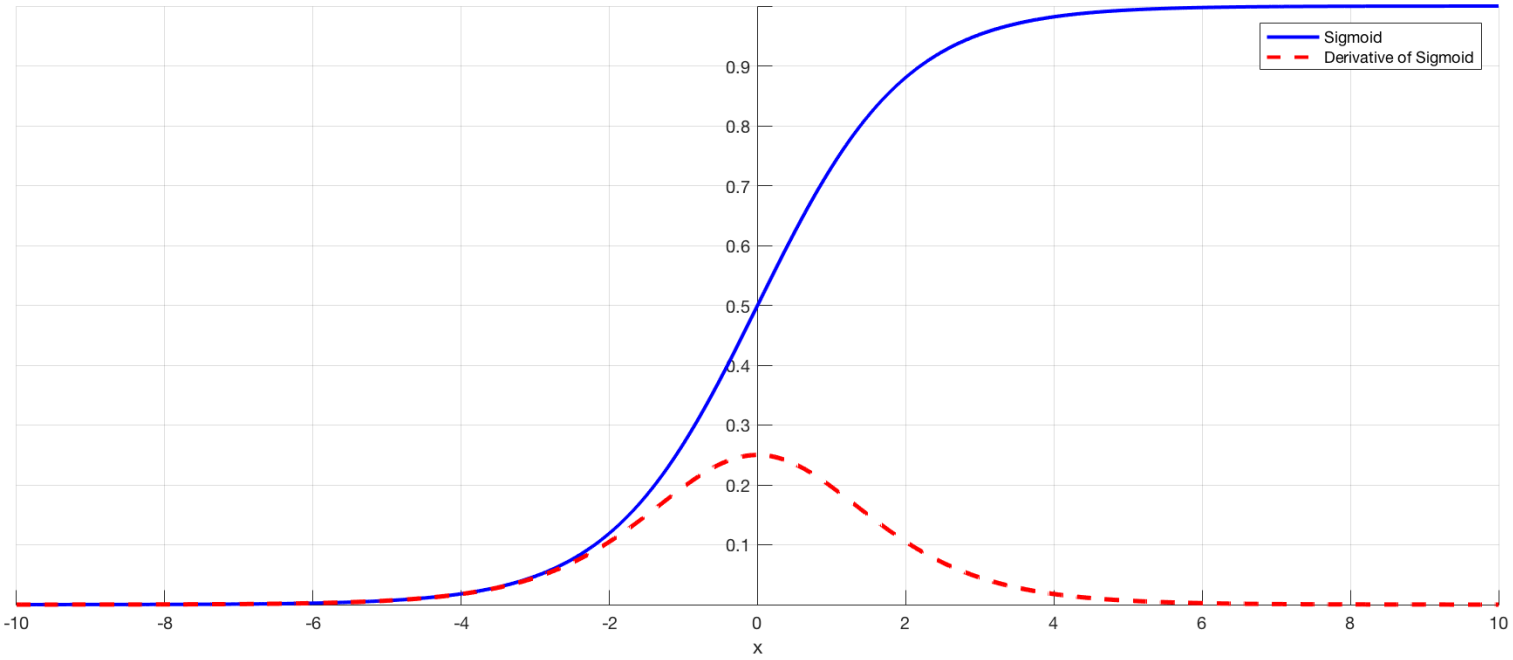




Problems with Going Deeper

- Vanishing gradients
  - As more layers using certain activation functions are added to neural networks, the gradients of the loss function approaches zero, making the network hard to train
  - Certain activation functions, like the sigmoid function, squishes a large input space into a small input space between 0 and 1. Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small

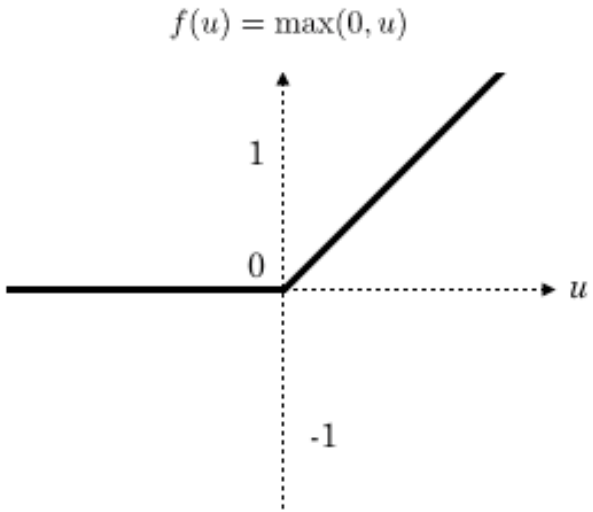
Vanishing Gradients



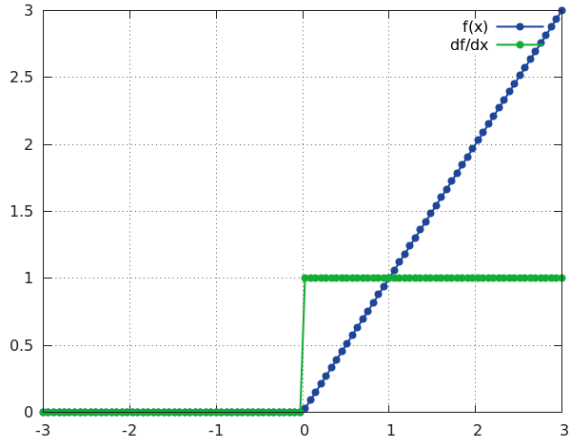
Vanishing Gradients

Use better activation function

- Rectified Linear Units (ramp)
  - $f(x) = \max(0, x)$
  - Derivative: All in or all out (unit step)
    - $f'(x) = 1 \; if \; x > 0 \; else \; 0$



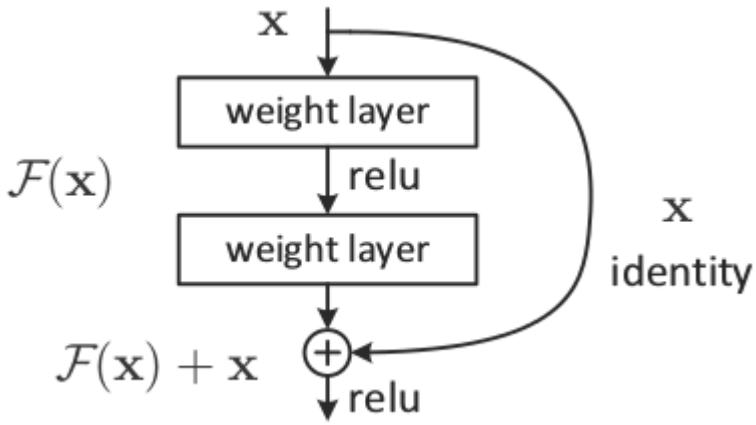
- Dead ReLUs
  - LeakyReLU: $f(x) = \max(x, 0.01x)$
  - PReLU: $f(x) = \max(x, ax)$



Vanishing Gradients

Use better architecture

- Residual networks
  - provide residual connections straight to earlier layers
  - This residual connection doesn't go through activation functions that "squashes" the derivatives, resulting in a higher overall derivative of the block

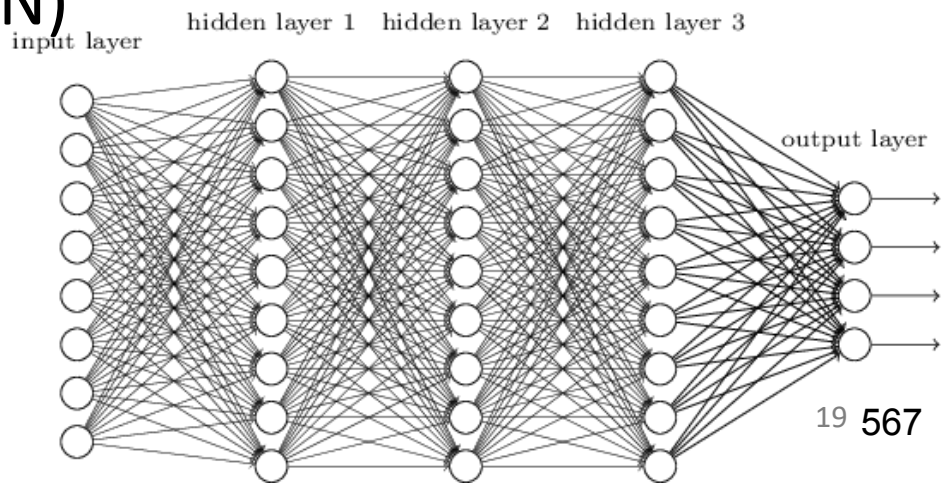


Use normalization

- Batch normalization

Problems with Going Deeper

- Parameter explosion
  - Too many weights to optimize as we go deeper
  - Search space is much harder to navigate
- Proposal: shared weights
  - Spatially shared (CNN)
  - Temporally shared (RNN)



University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

Topic 15 – Convolutional Neural Network

Zheng LU

2024 Autumn

CNN Layers

- **INPUT**: will hold the raw data (e.g., pixel values of the image)
- **CONV**: will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume
- **RELU**: will apply an elementwise activation function
- **POOL**: will perform a downsampling operation along the spatial dimensions (width, height)
- **FC**: will compute the class scores

Convolution Layer

Those are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Matrix

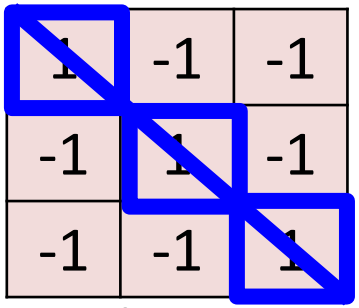
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

Matrix

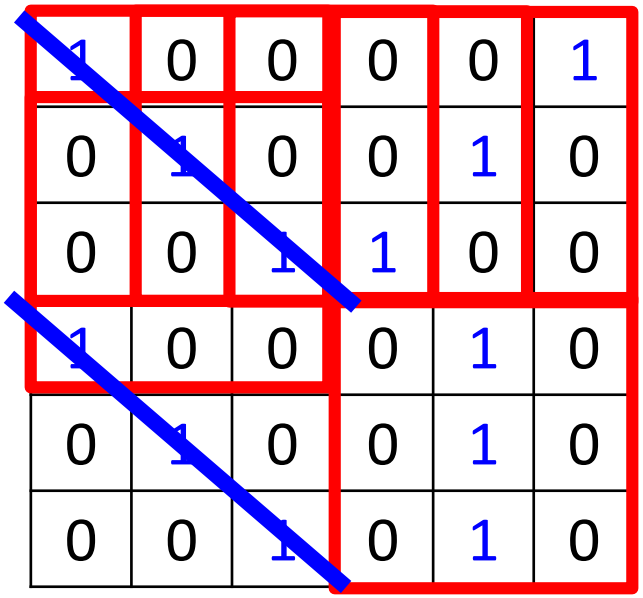
Each filter detects a small pattern (3 x 3).

CNN – Convolution

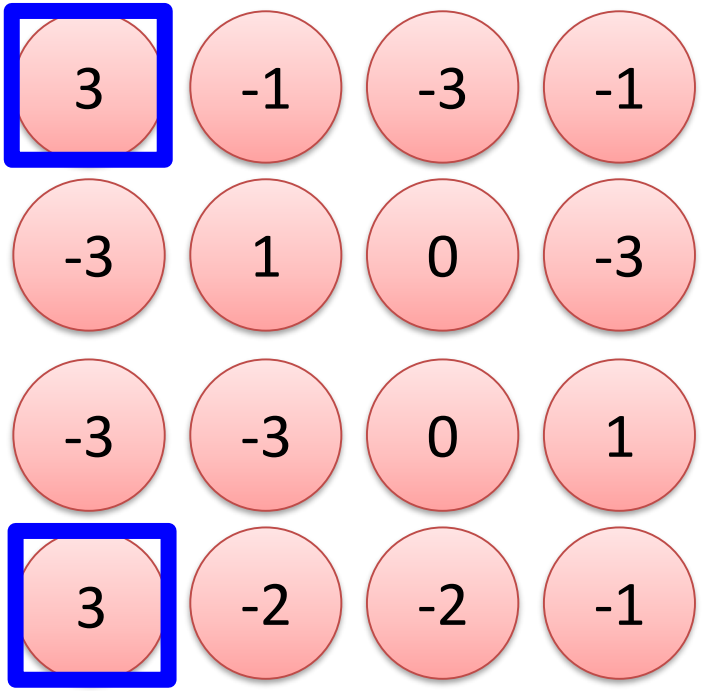


Filter 1

stride=1



6 x 6 image



CNN – Convolution

-1	1	-1
-1	1	-1
-1	1	-1

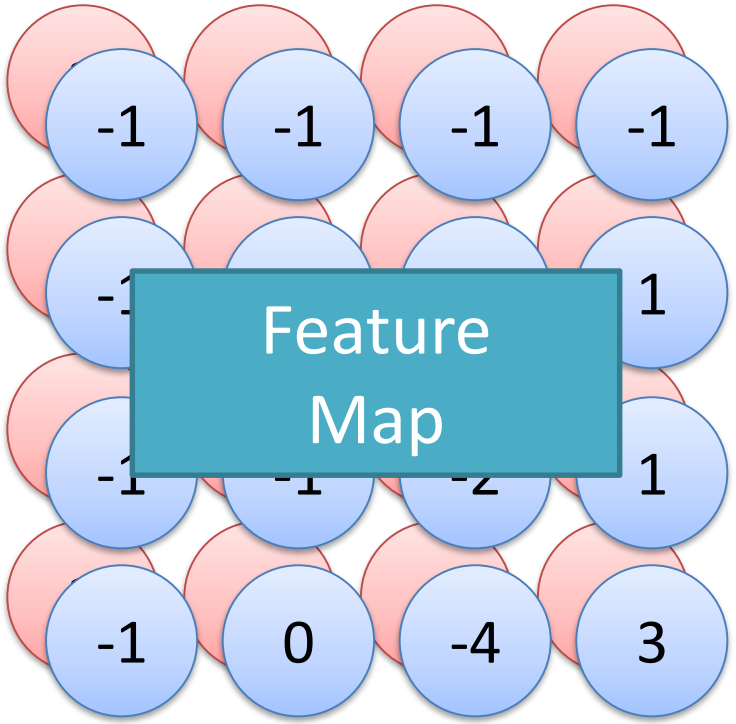
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

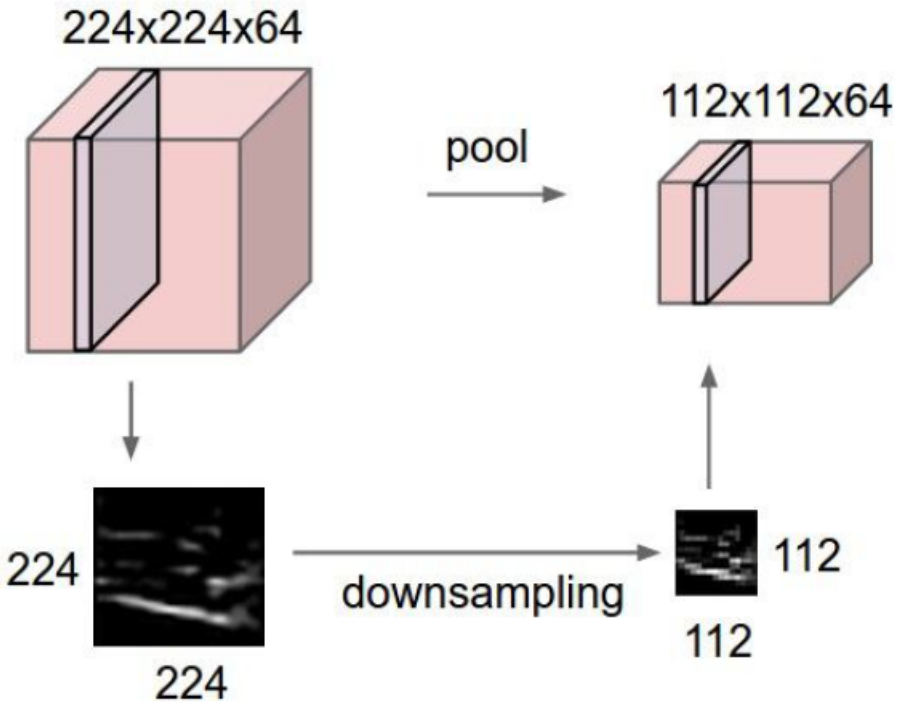
Do the same process for every filter



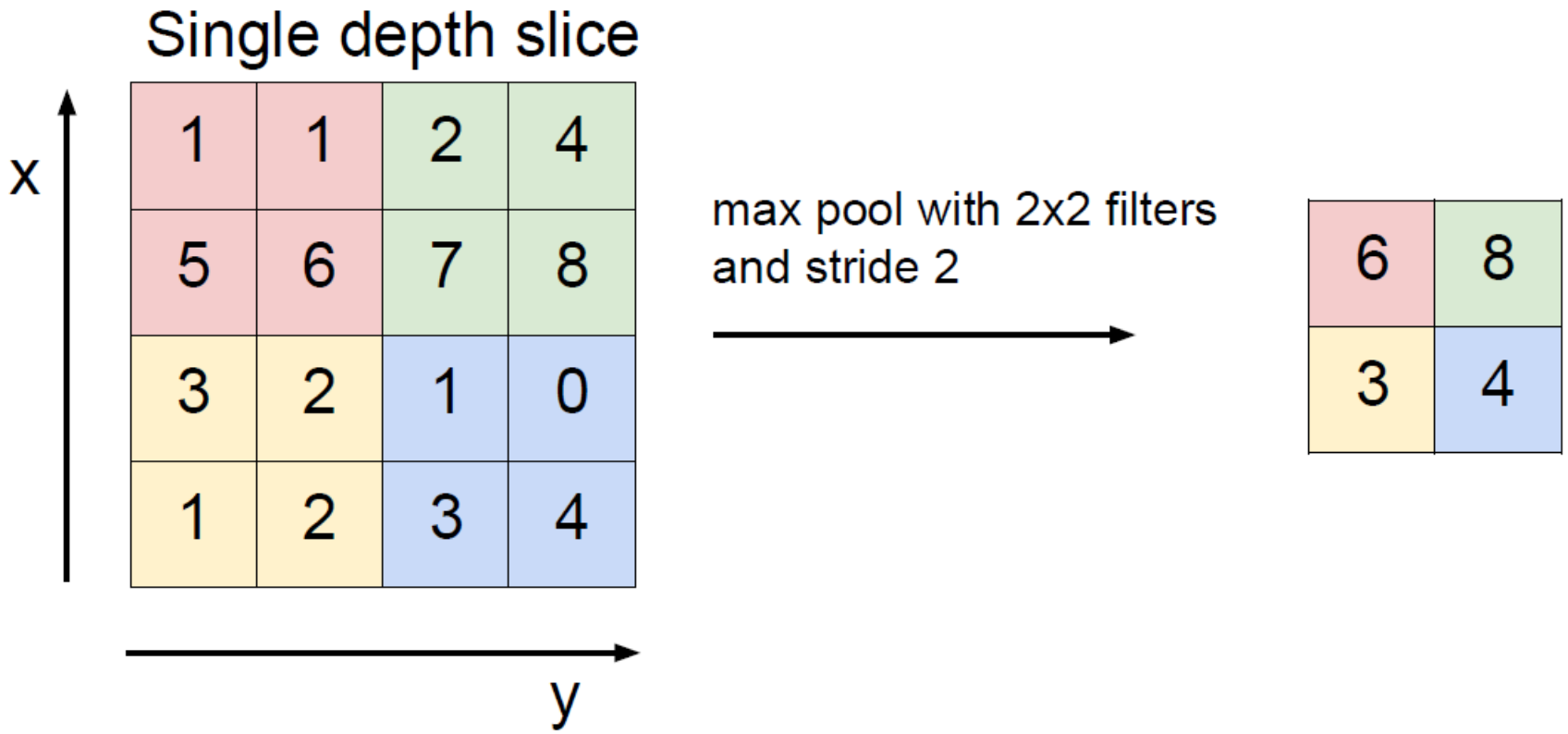
4 x 4 image

Pooling Layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



Pooling Layer



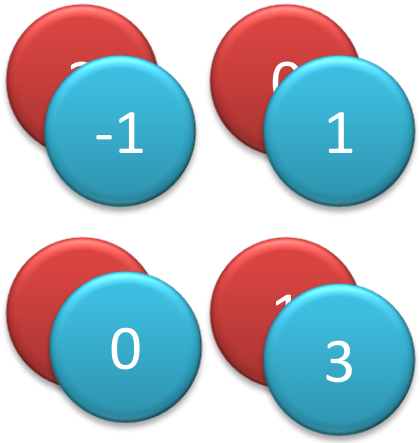
Pooling Layer

- The result of using a pooling layer and creating down sampled or pooled feature maps is a **summarized version** of the **features detected** in the input.
- They are useful as small changes in the location of the feature in the input detected by the convolutional layer will result in a pooled feature map with the feature in the same location.
- This capability added by pooling is called the model's **invariance to local translation**.

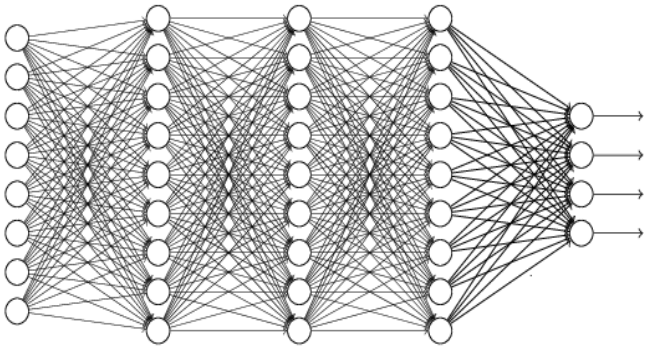
Pooling Layer

- *"In all cases, pooling helps to make the representation become approximately invariant to small translations of the input. Invariance to translation means that if we translate the input by a small amount, the values of most of the pooled outputs do not change"*

Flatten



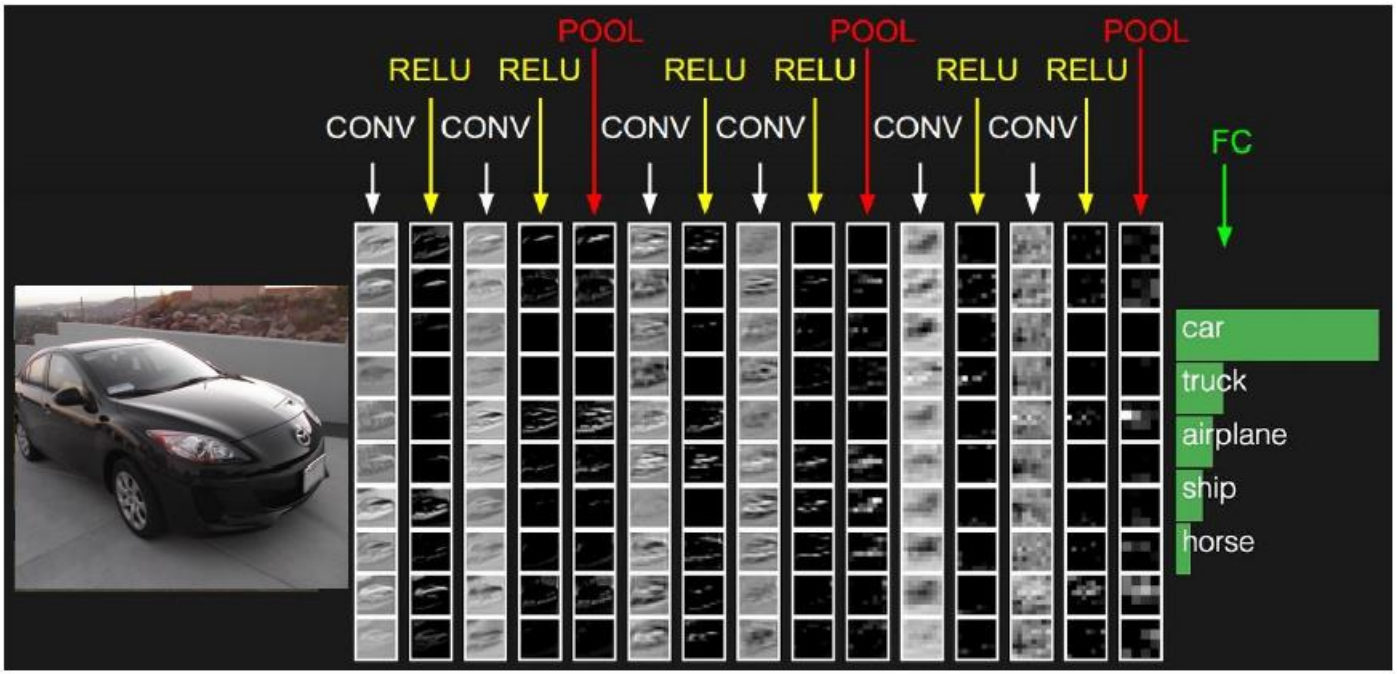
Flatten



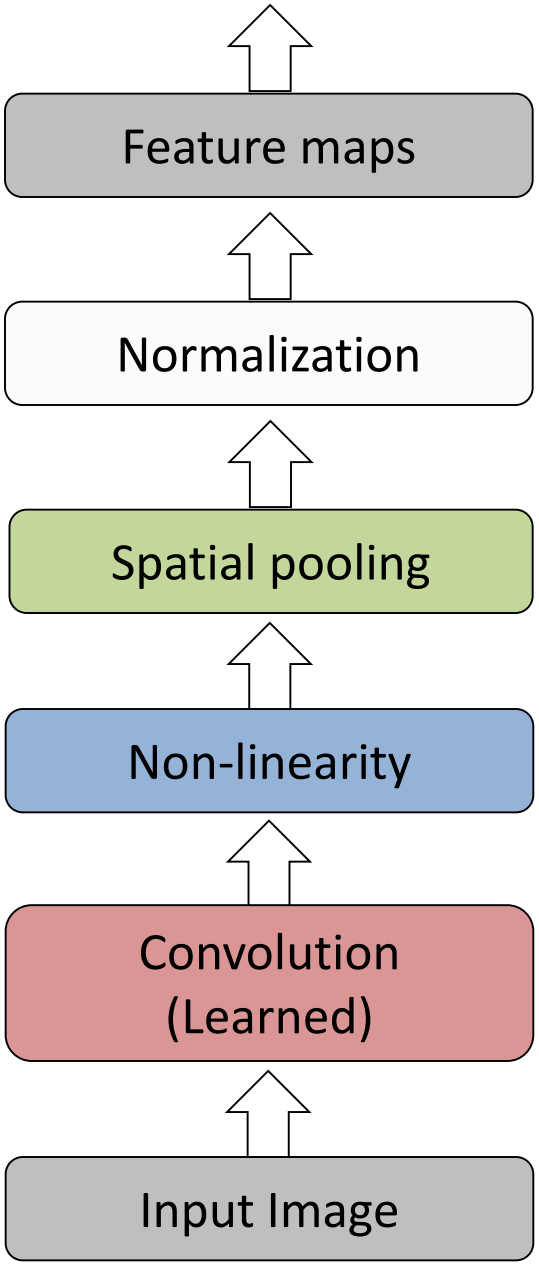
Fully Connected Feedforward network

Fully Connected Layer

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks

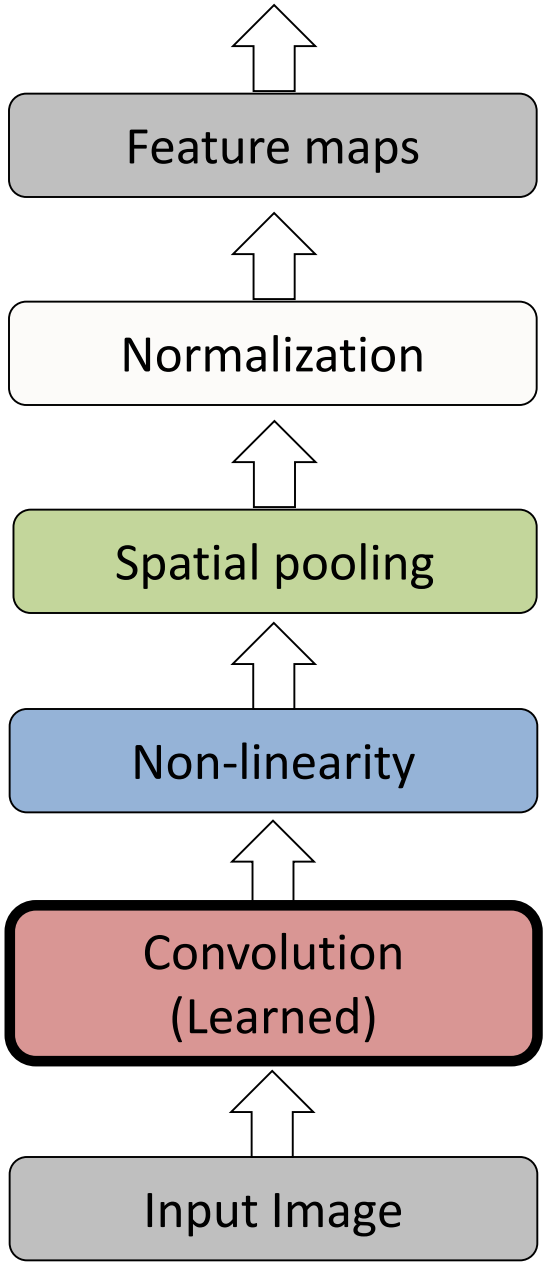


Convolutional Neural Networks



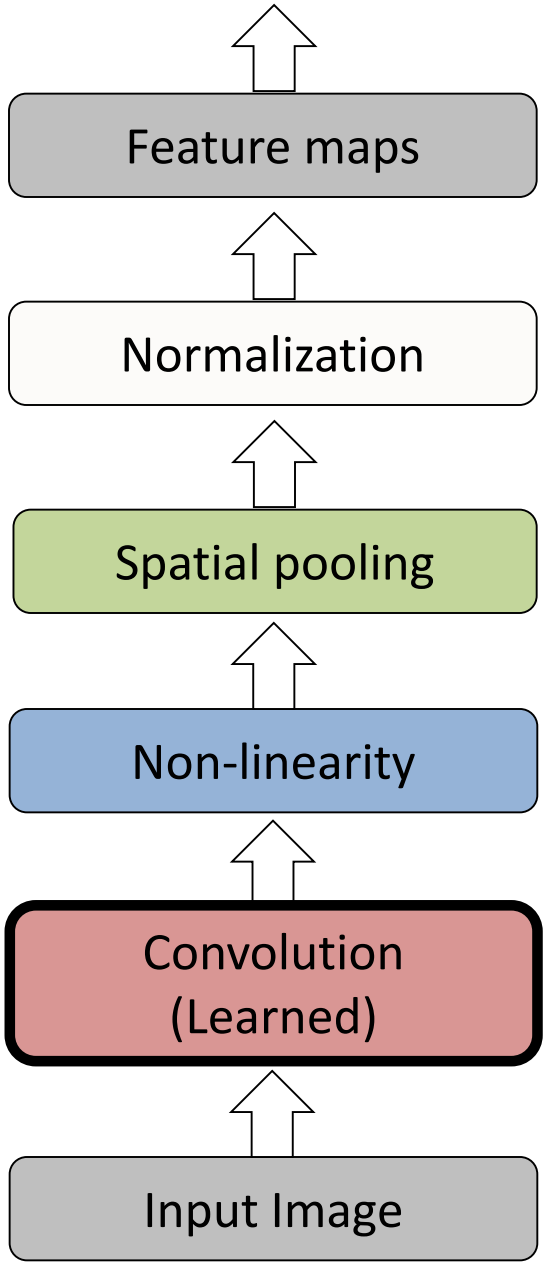
slide credit: S. Lazebnik

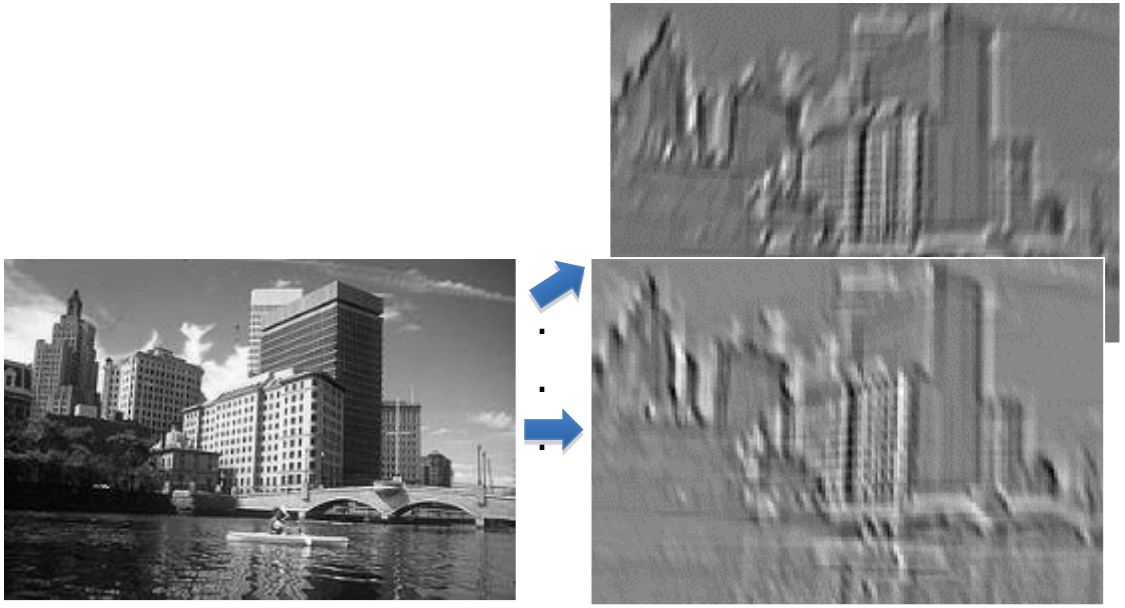
Convolutional Neural Networks



slide credit: S. Lazebnik

Convolutional Neural Networks



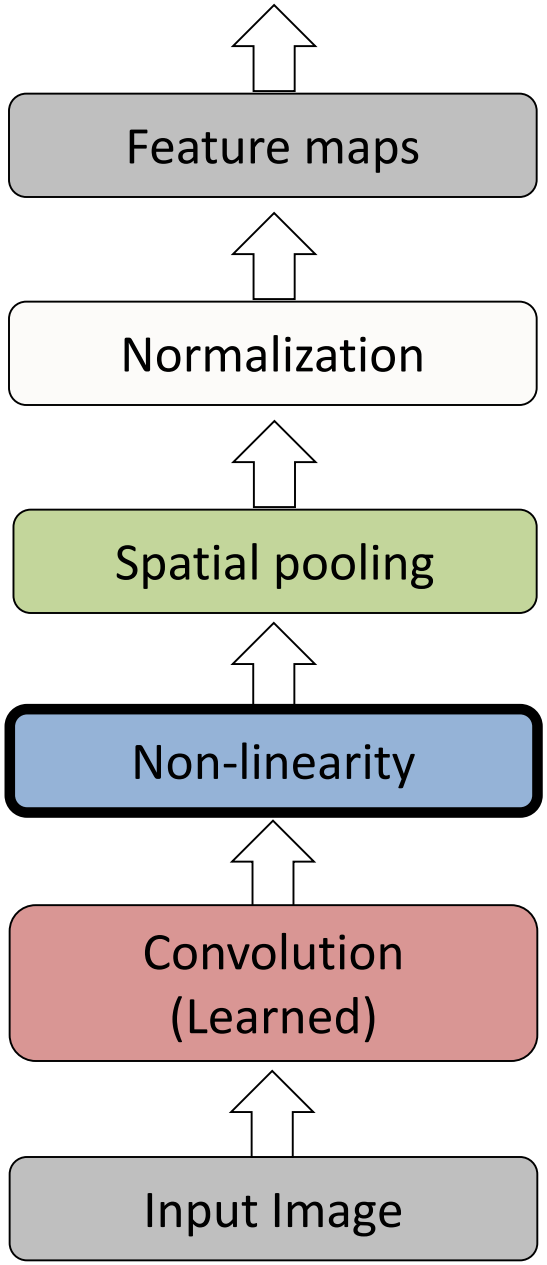


Input

Feature Map

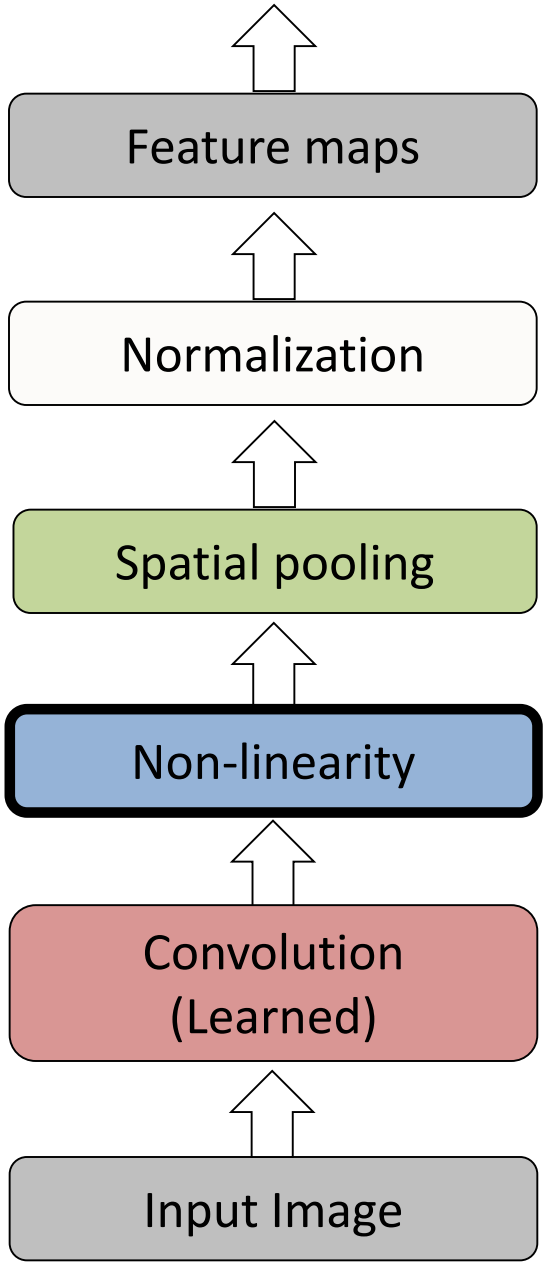
slide credit: S. Lazebnik

Convolutional Neural Networks

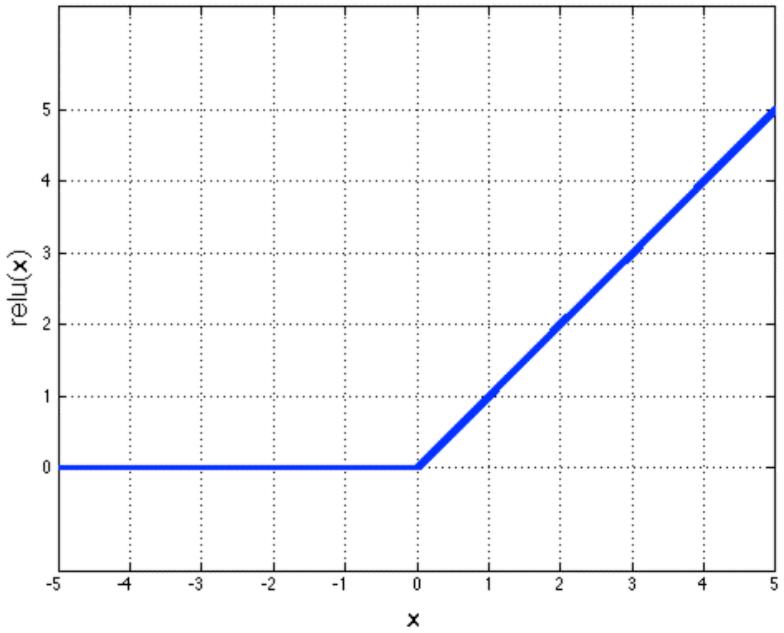


slide credit: S. Lazebnik

Convolutional Neural Networks

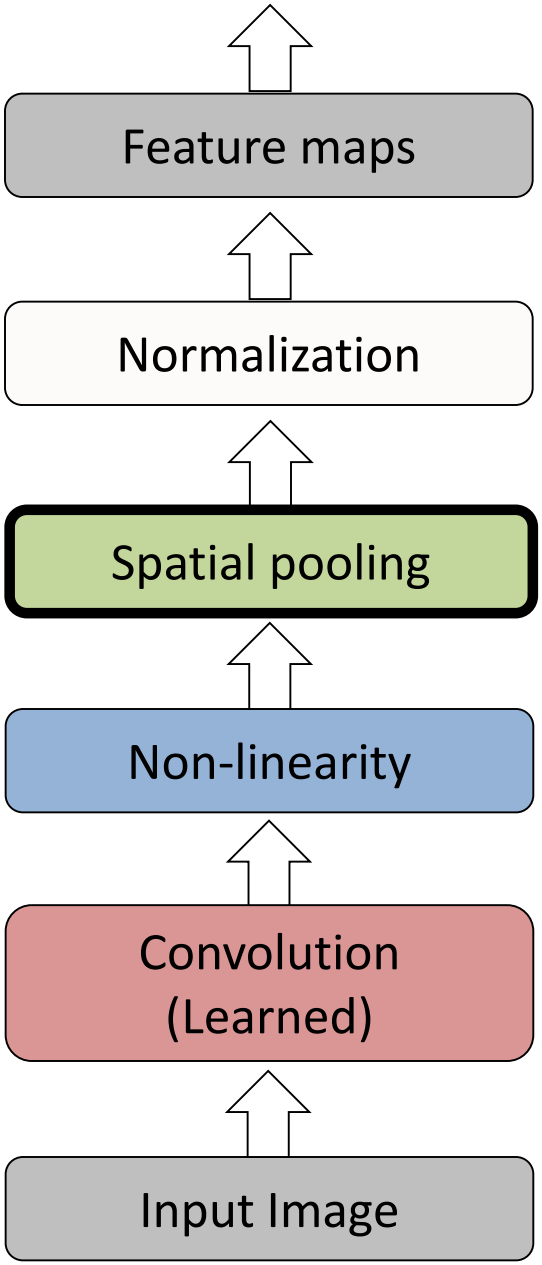


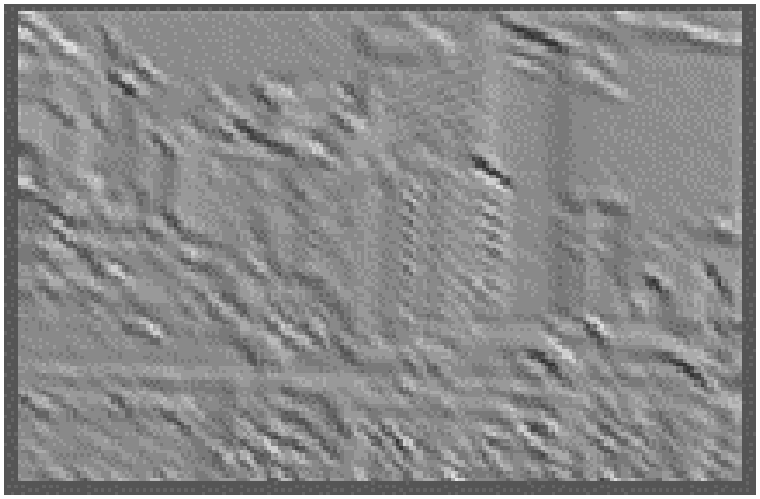
Rectified Linear Unit (ReLU)



slide credit: S. Lazebnik

Convolutional Neural Networks

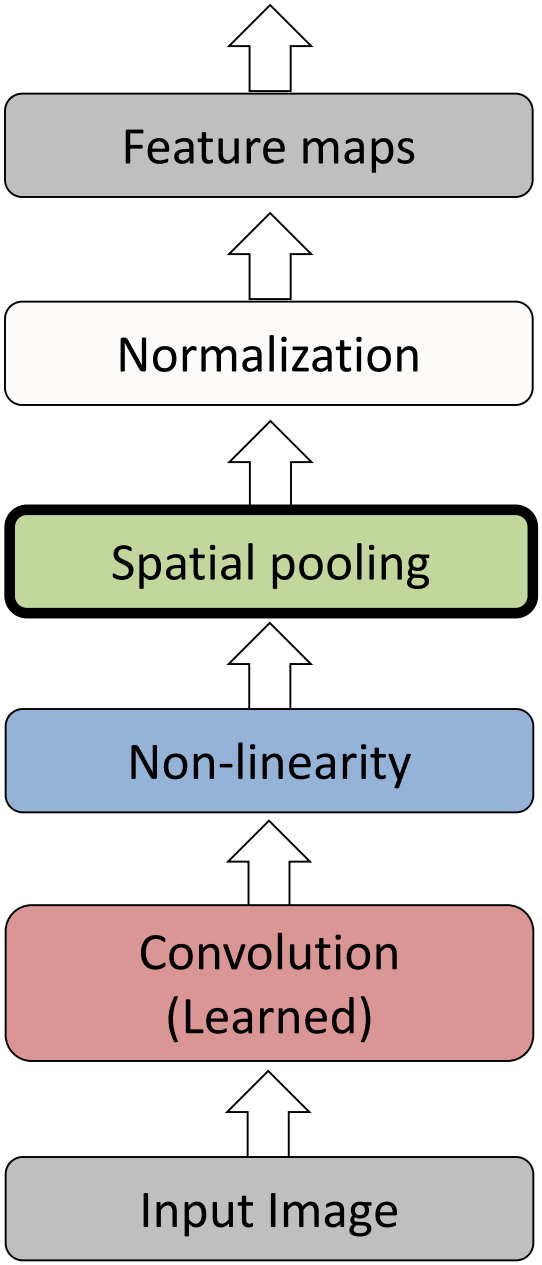


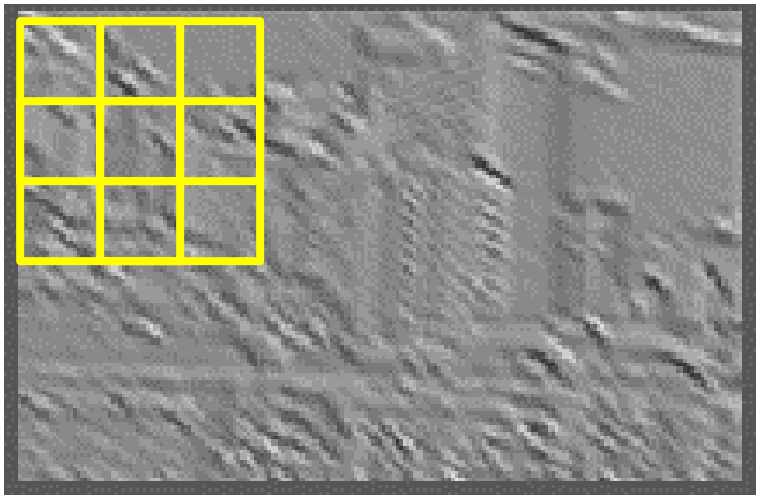


Max-pooling: a non-linear down-sampling

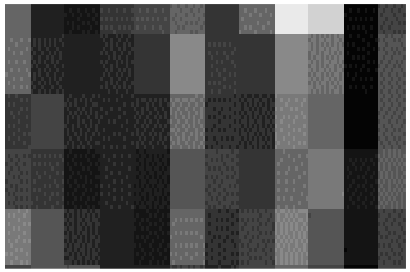
slide credit: S. Lazebnik

Convolutional Neural Networks





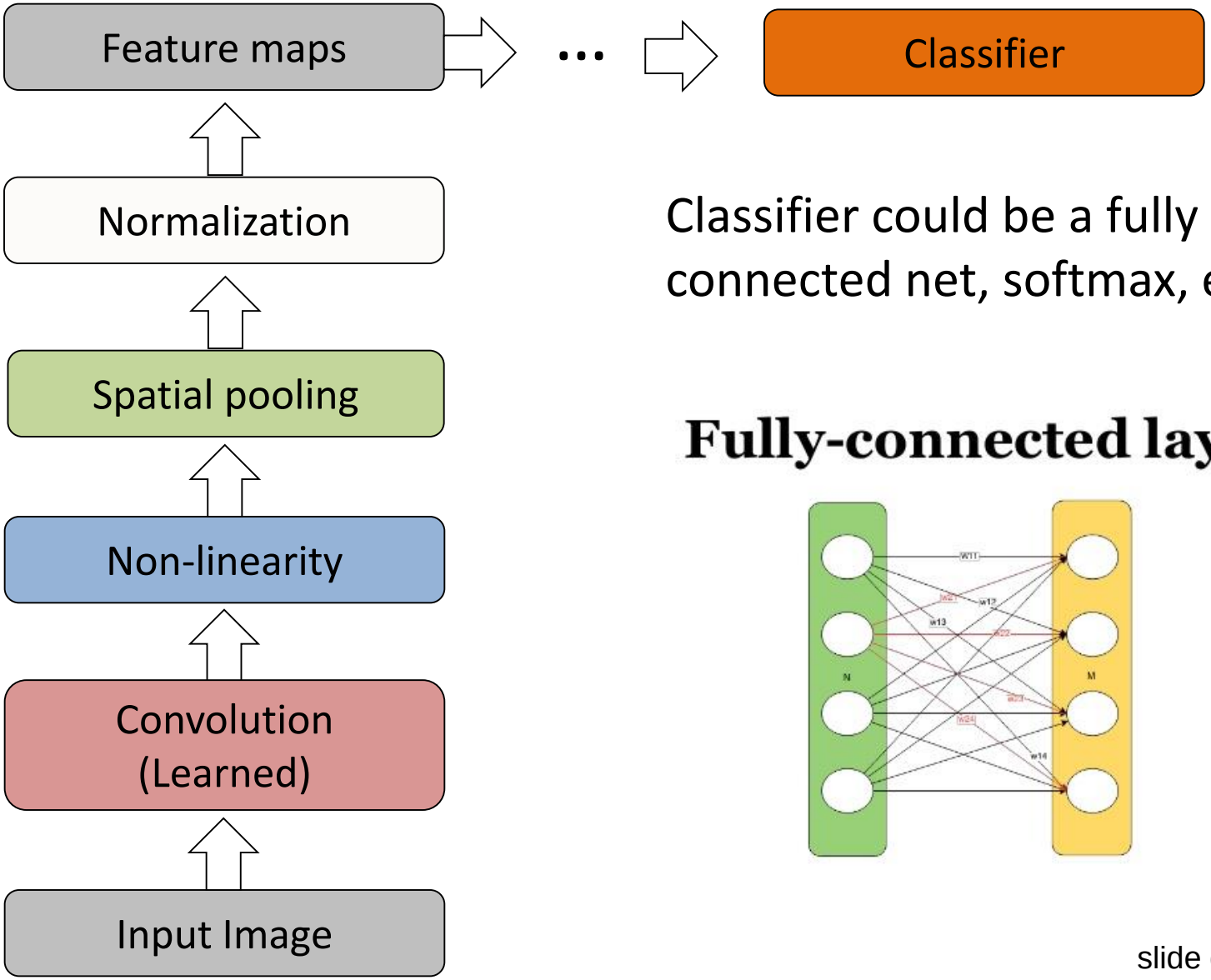
Max pooling



Max-pooling: a non-linear down-sampling

slide credit: S. Lazebnik

Convolutional Neural Networks

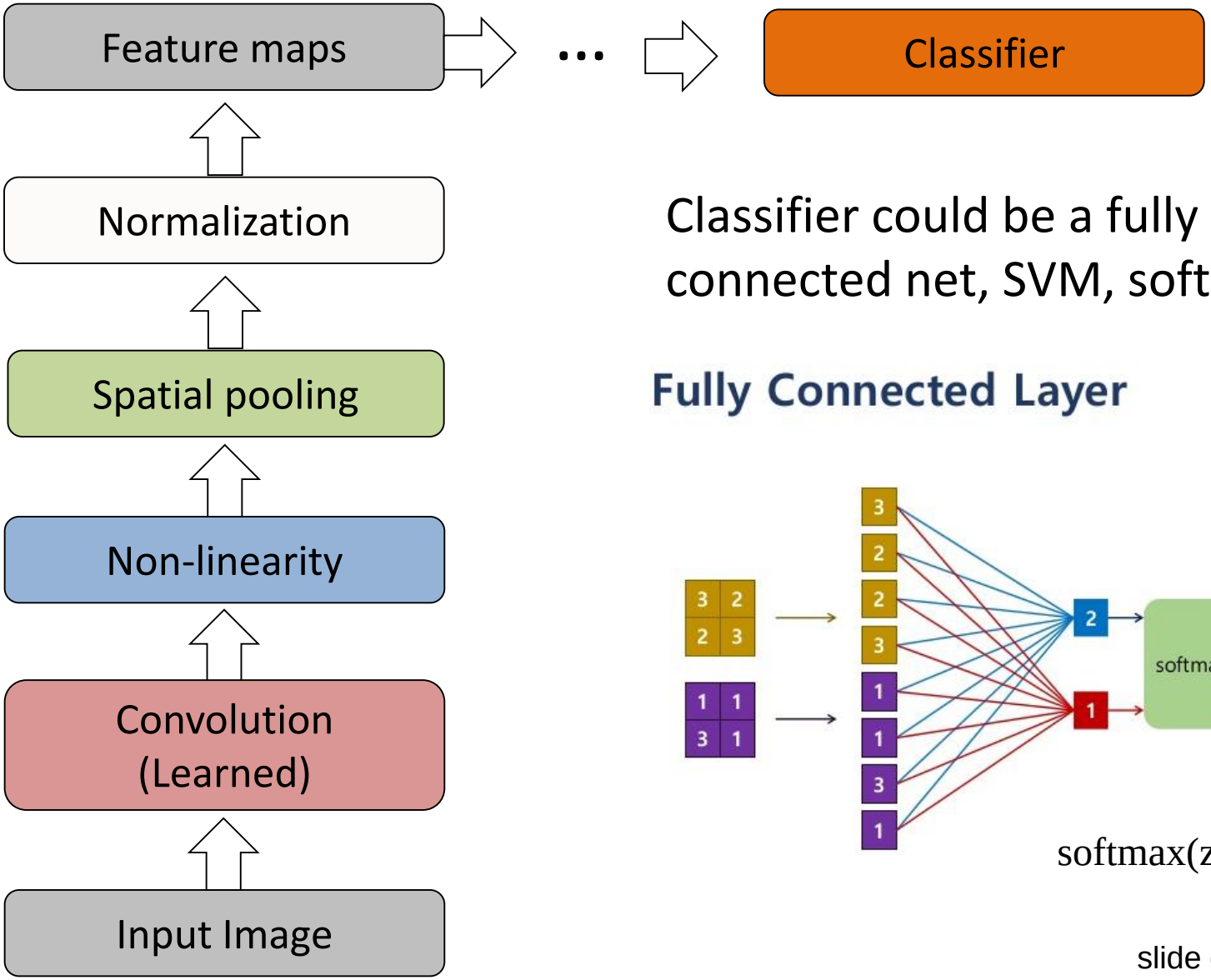


Classifier could be a fully connected net, softmax, etc.

Fully-connected layer

slide credit: S. Lazebnik

Convolutional Neural Networks



Classifier could be a fully connected net, SVM, softmax, etc.

slide credit: S. Lazebnik

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

Topic 16 – RNN,LSTM

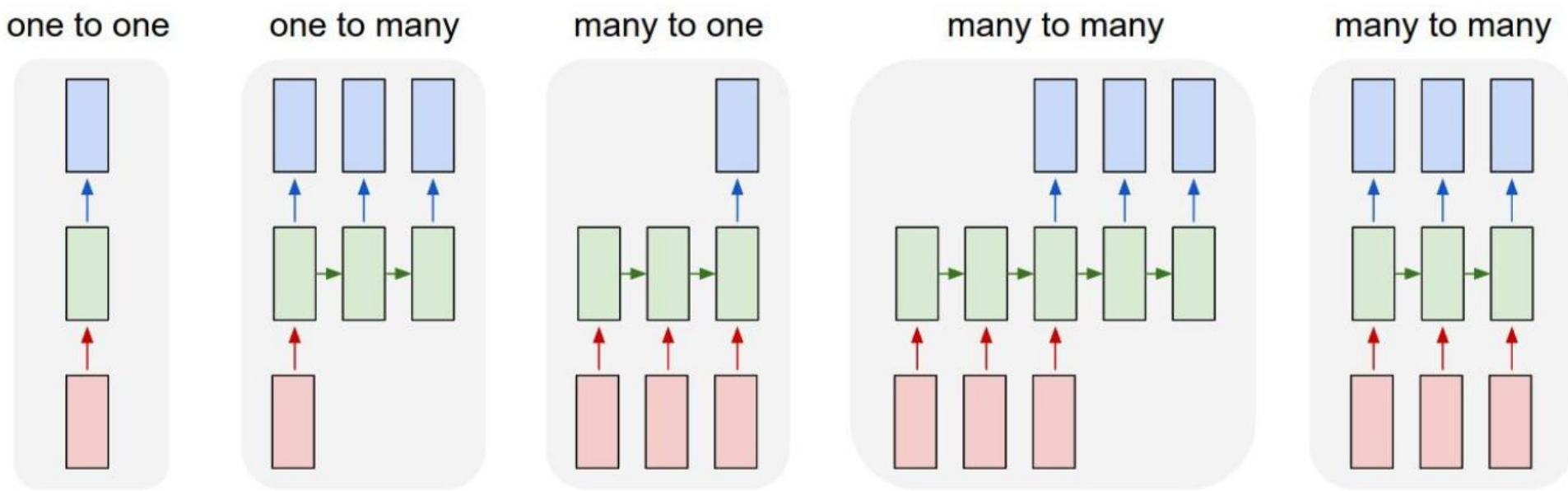
Zheng Lu

2024 Autumn

Recurrent Neural Networks

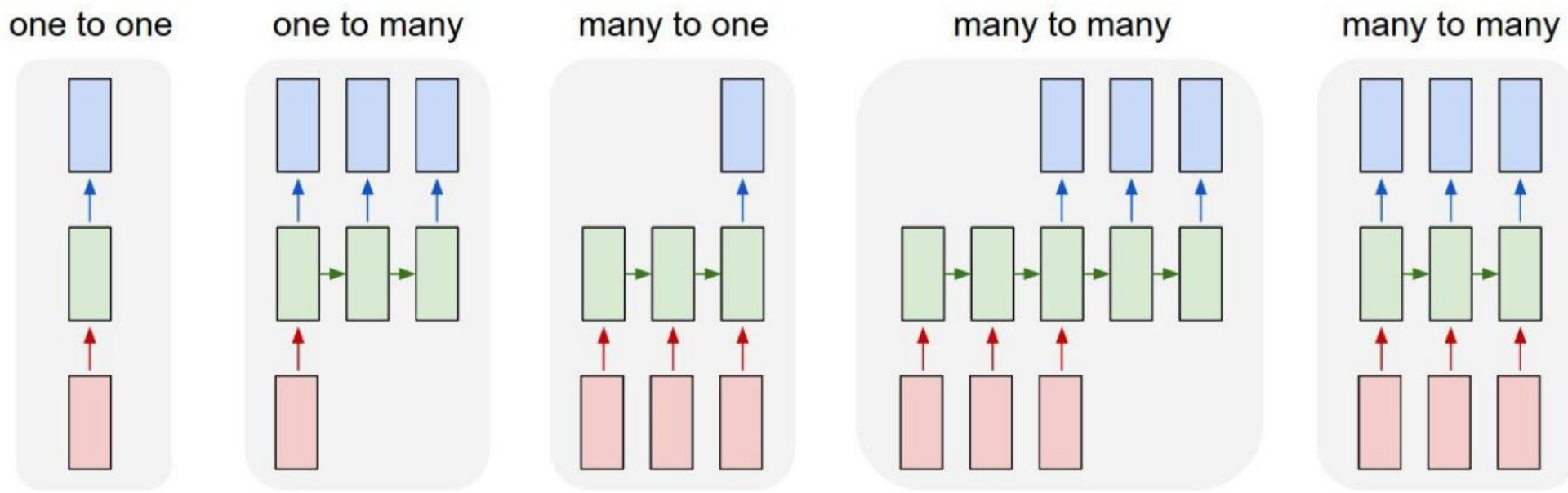
- Excellent models for problems more than one-to-one
  - Time series prediction and classification.
  - Sequence prediction and classification.
  - Simplify some problems that are difficult for multi-layer perceptron.

RNN: Process Sequences



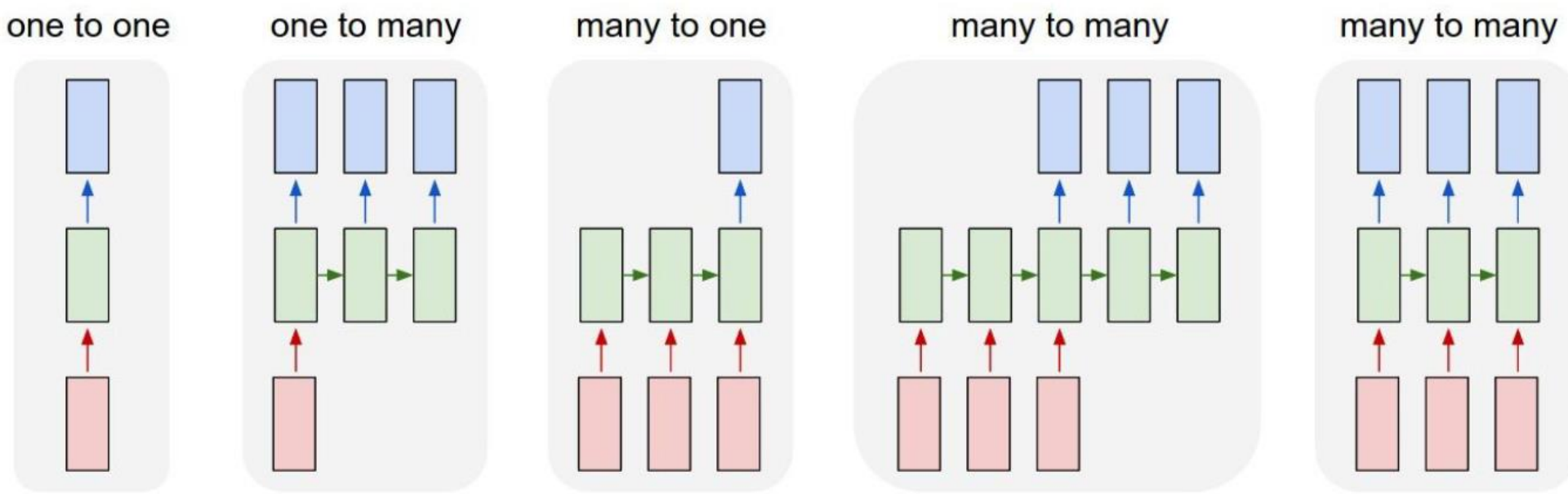
e.g. **Image Captioning**
image -> sequence of words

RNN: Process Sequences



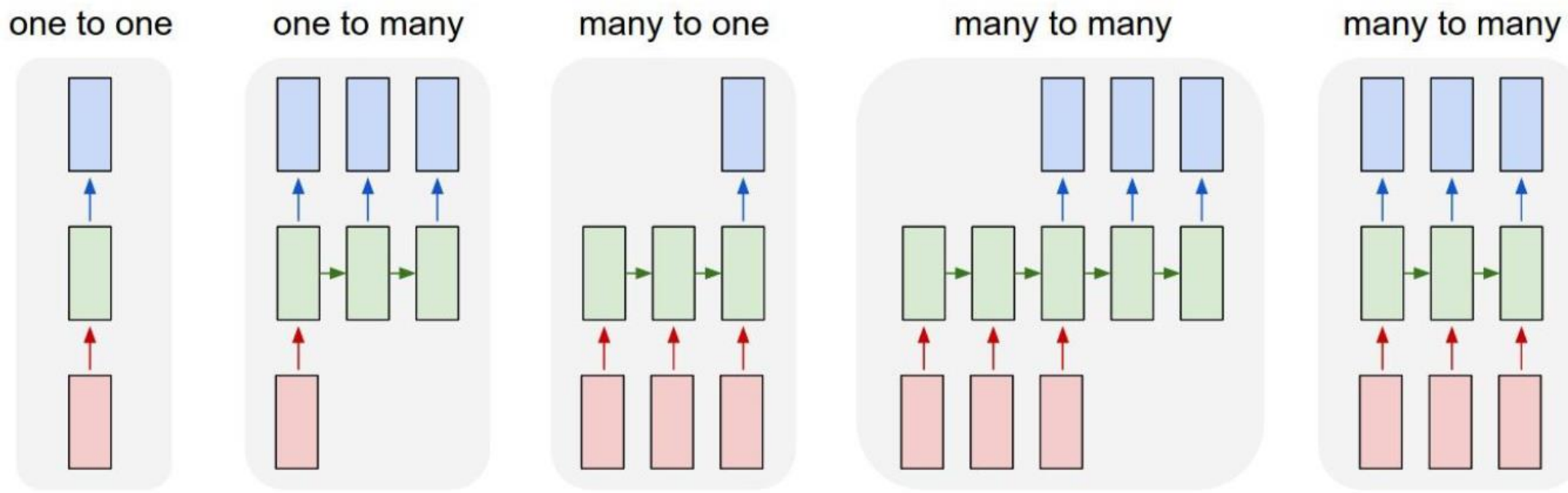
e.g. **Sentiment Classification**
sequence of words -> sentiment

RNN: Process Sequences



e.g. **Machine Translation**
seq of words -> seq of words

RNN: Process Sequences



e.g. **Video classification on frame level**

RNN

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

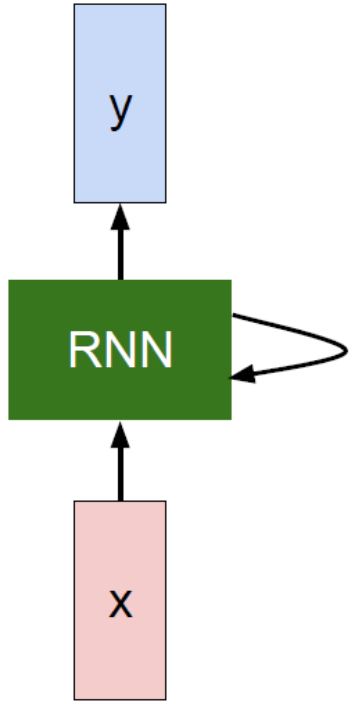
$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

input vector at some time step

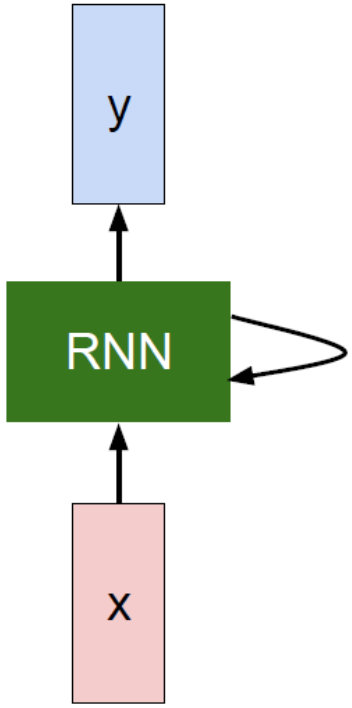
some function with parameters W



RNN

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

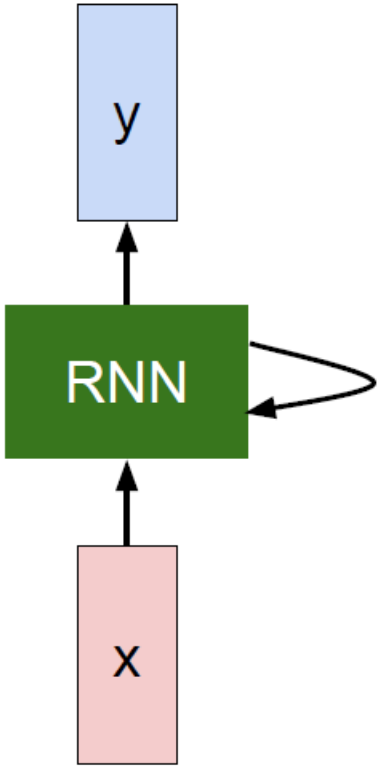
$$h_t = f_W(h_{t-1}, x_t)$$



Notice: the same function and the same set of parameters are used at every time step.

RNN

The state consists of a single *"hidden"* vector **h**:

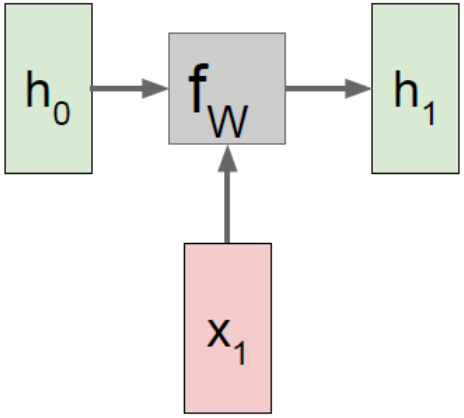


$$h_t = f_W(h_{t-1}, x_t)$$

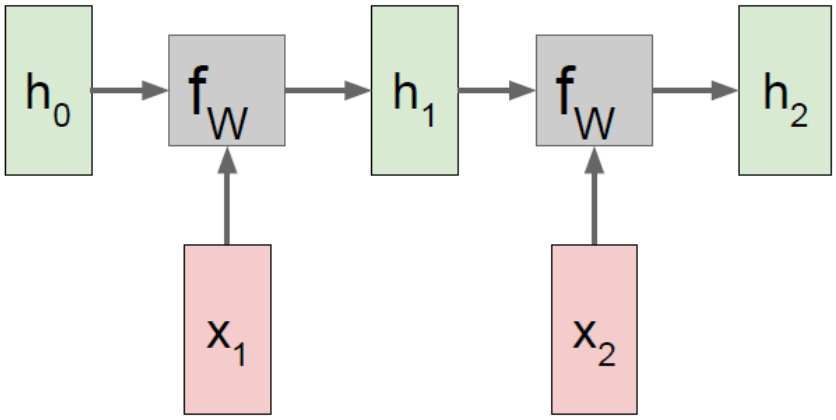
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

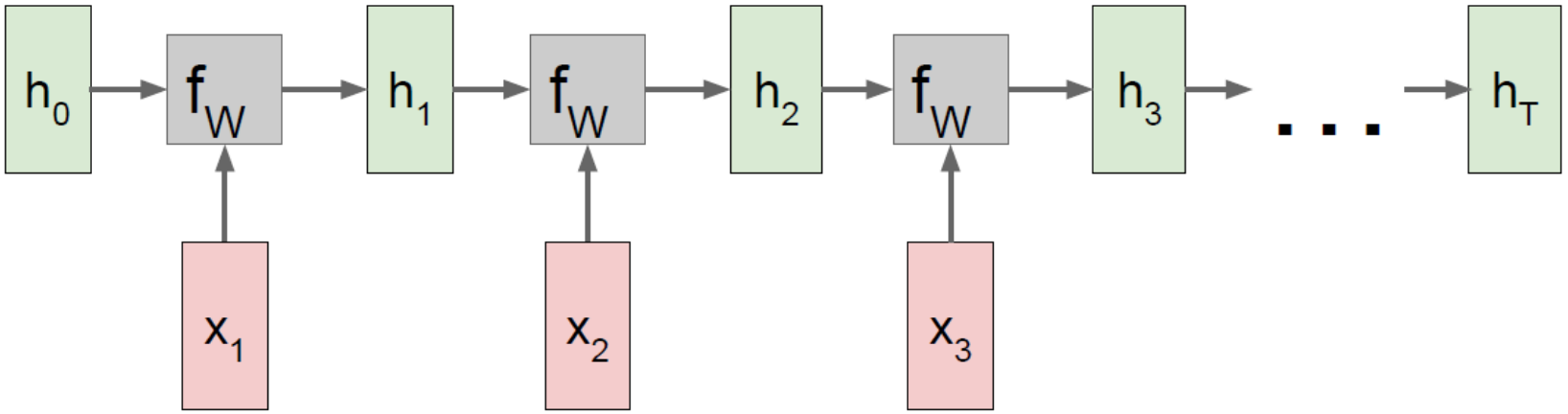
RNN: Computational Graph



RNN: Computational Graph

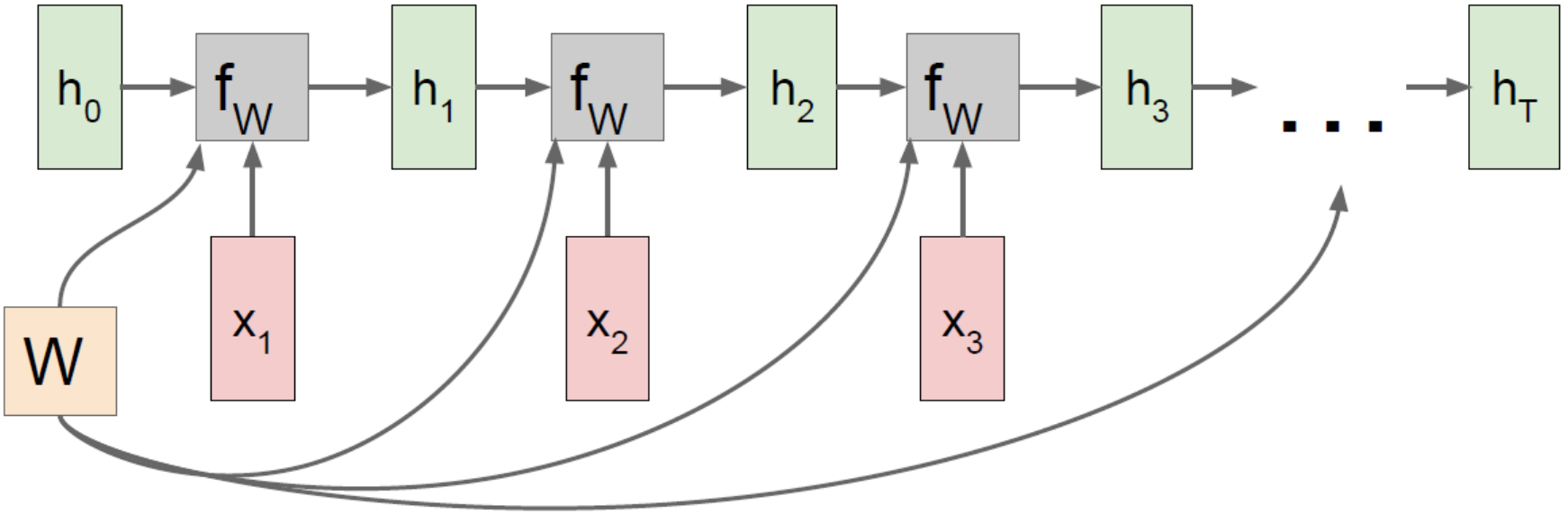


RNN: Computational Graph

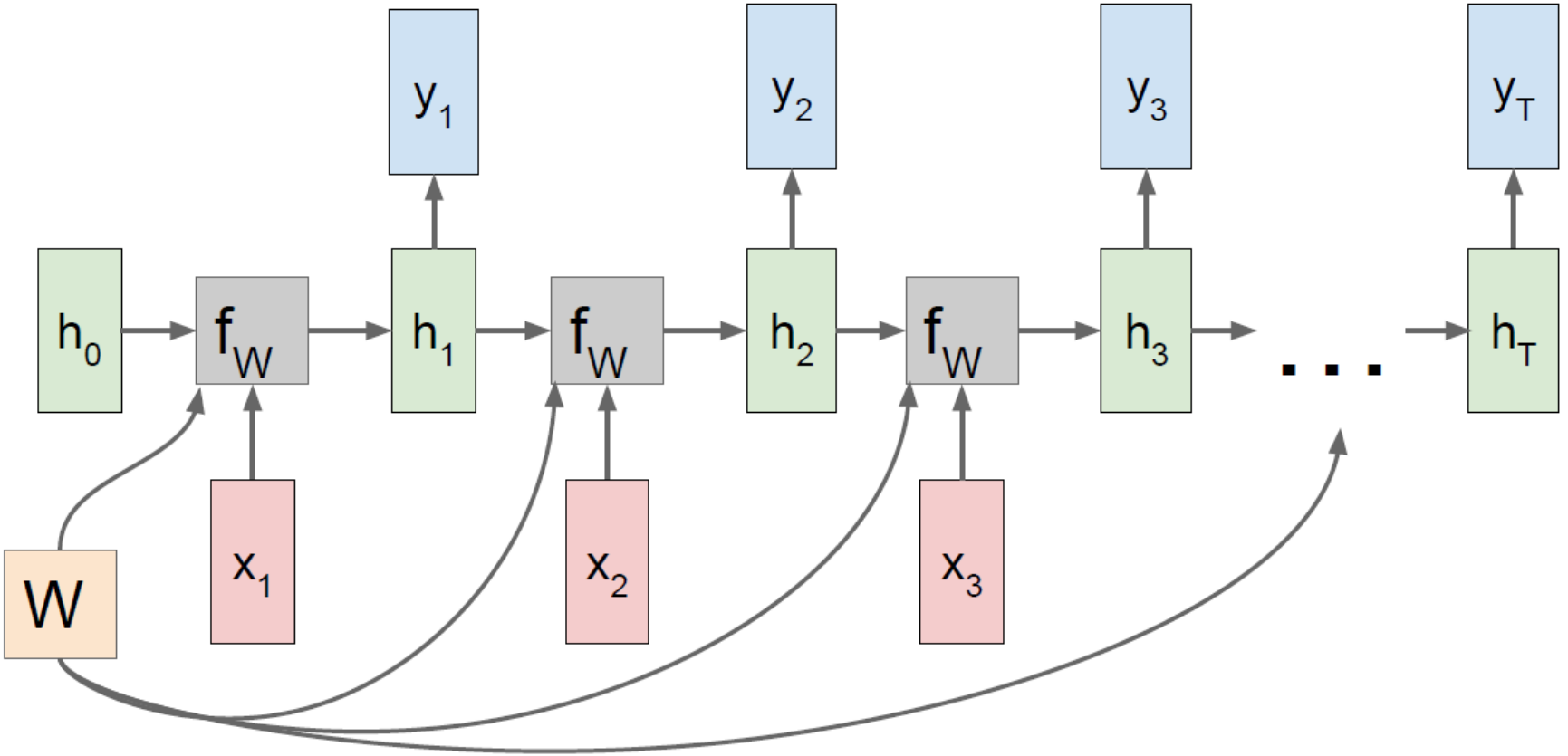


RNN: Computational Graph

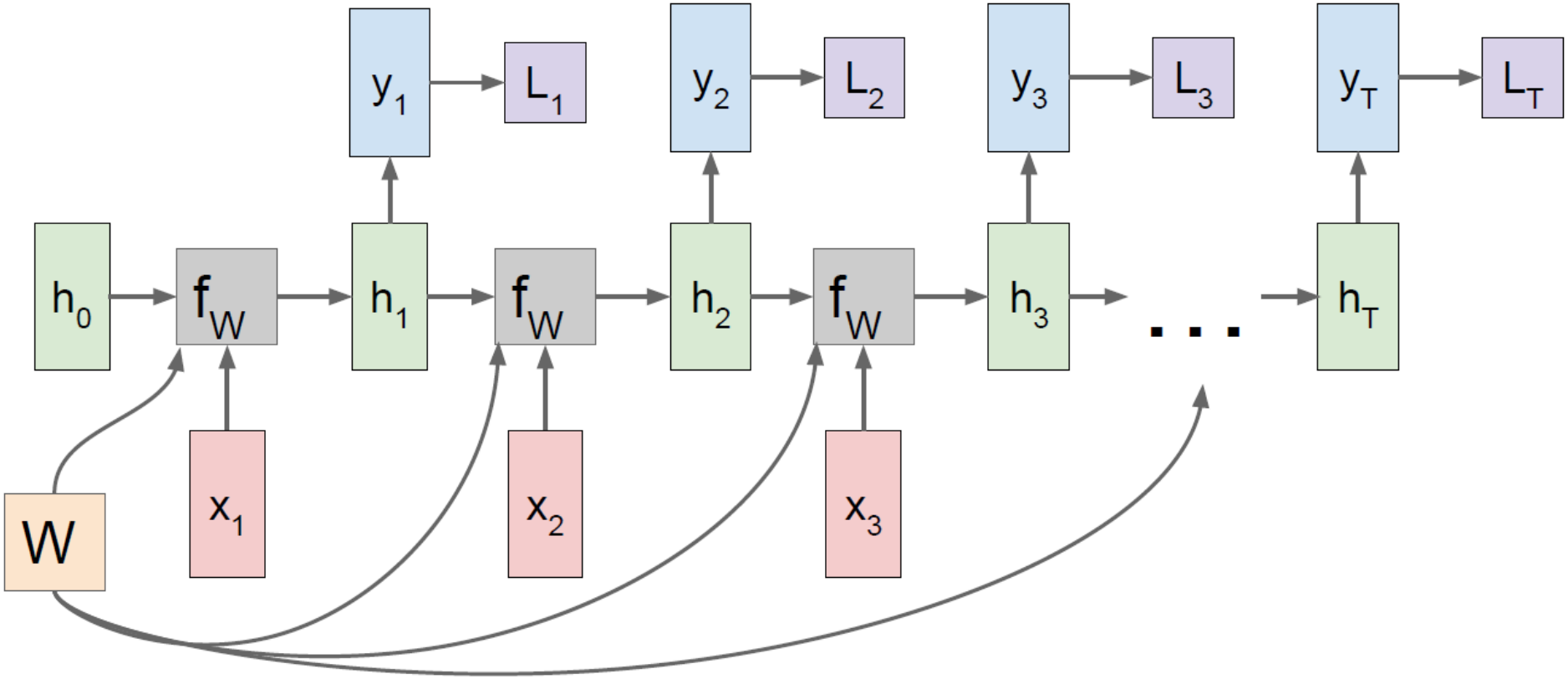
Re-use the same weight matrix at every time-step



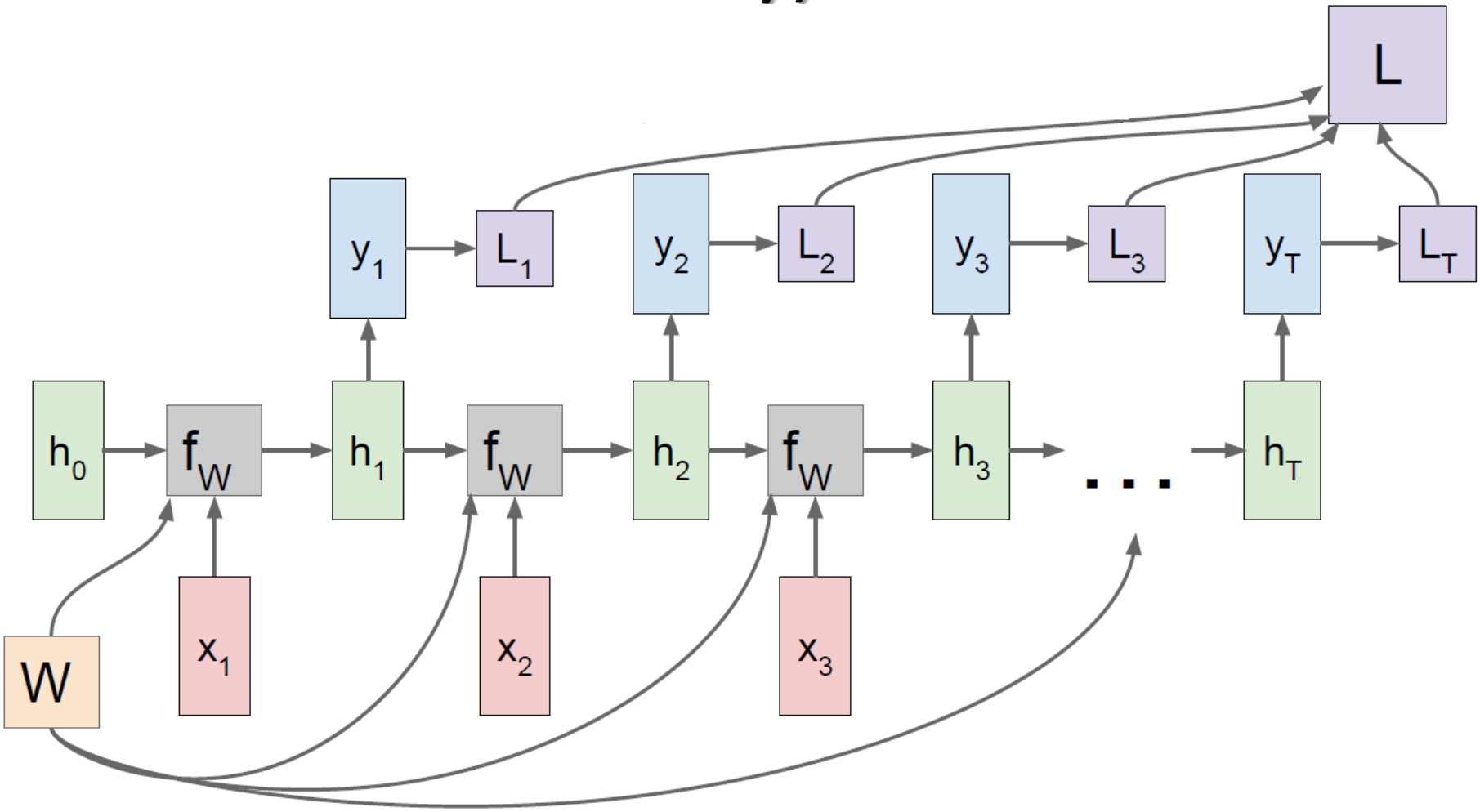
RNN: Computational Graph (Many to Many)



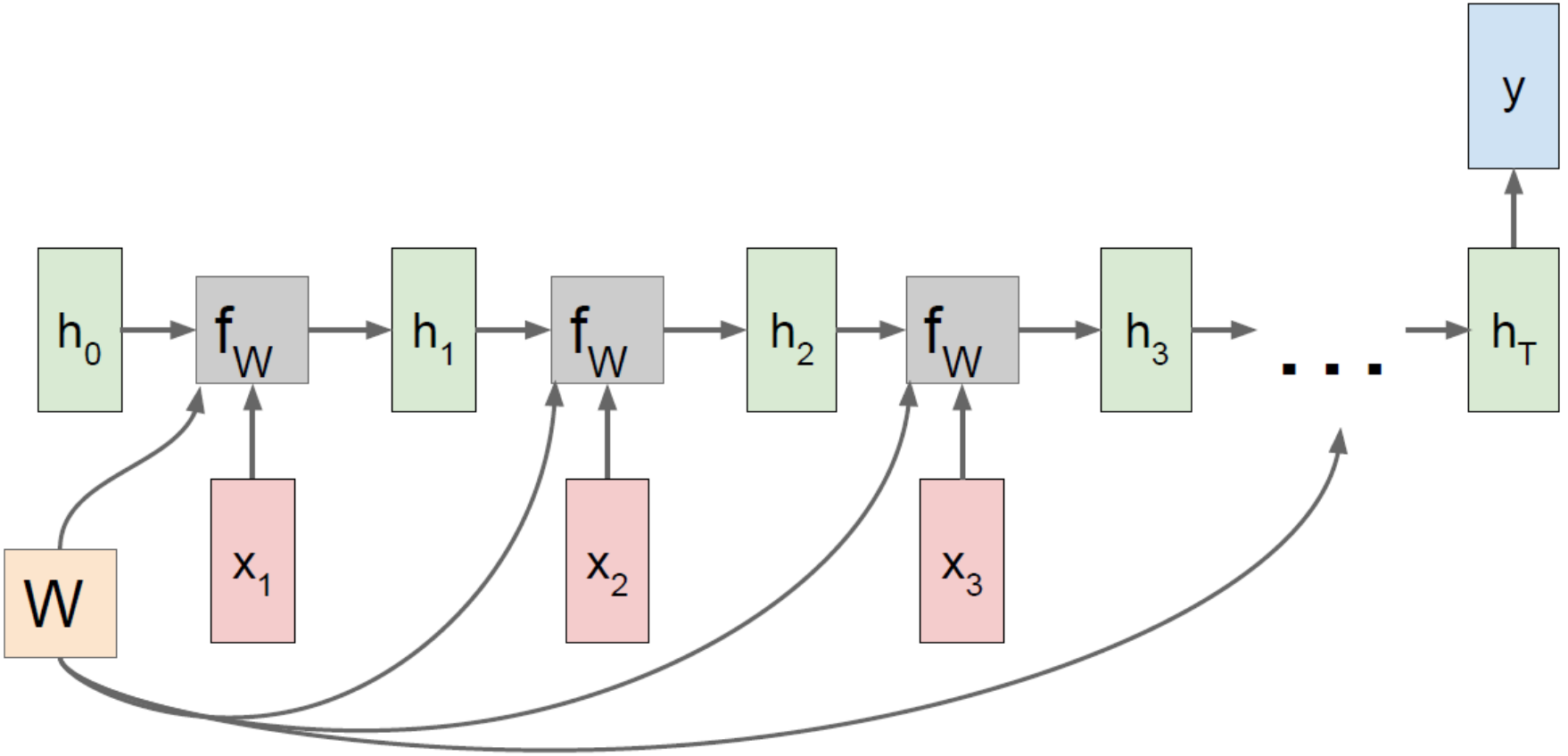
RNN: Computational Graph (Many to Many)



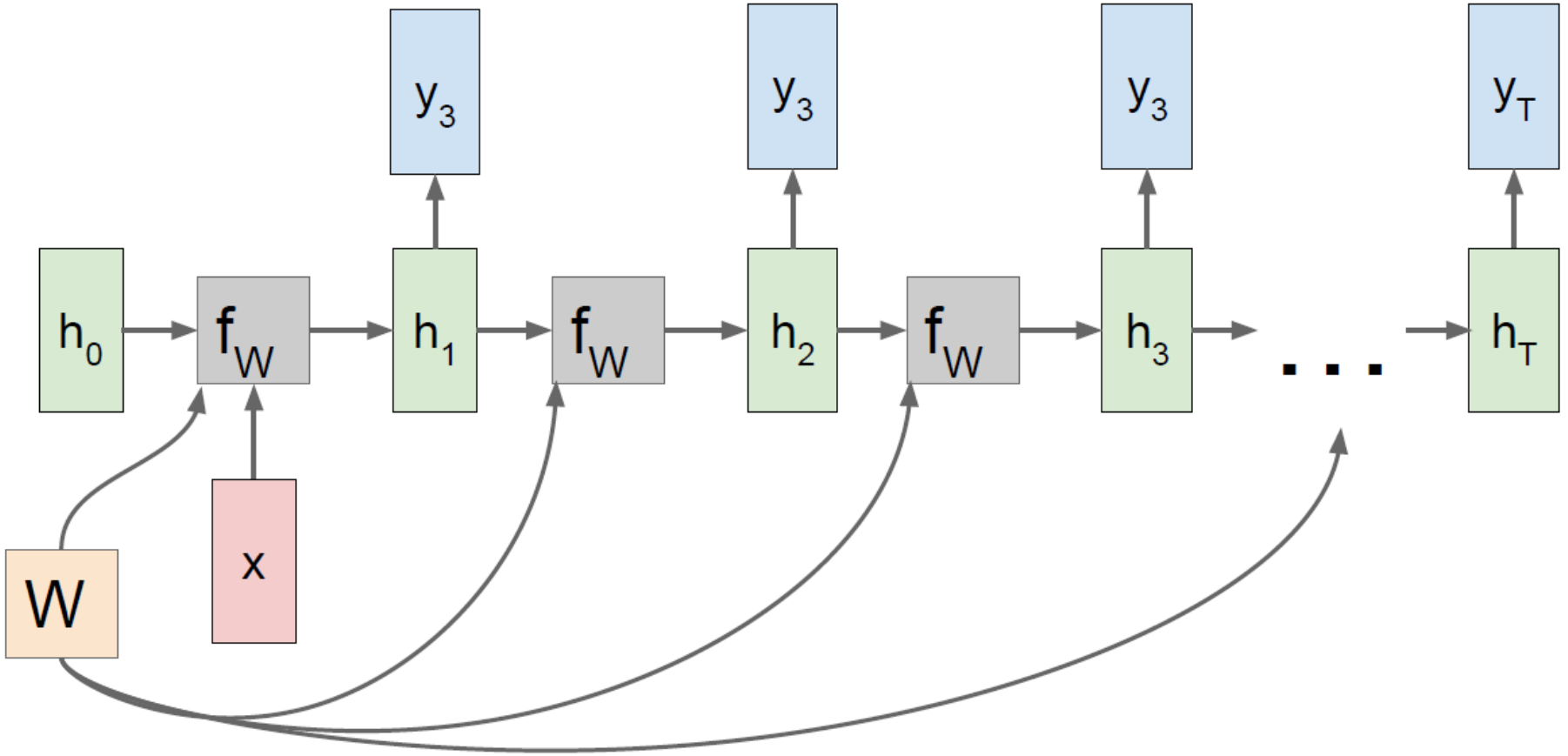
RNN: Computational Graph (Many to Many)



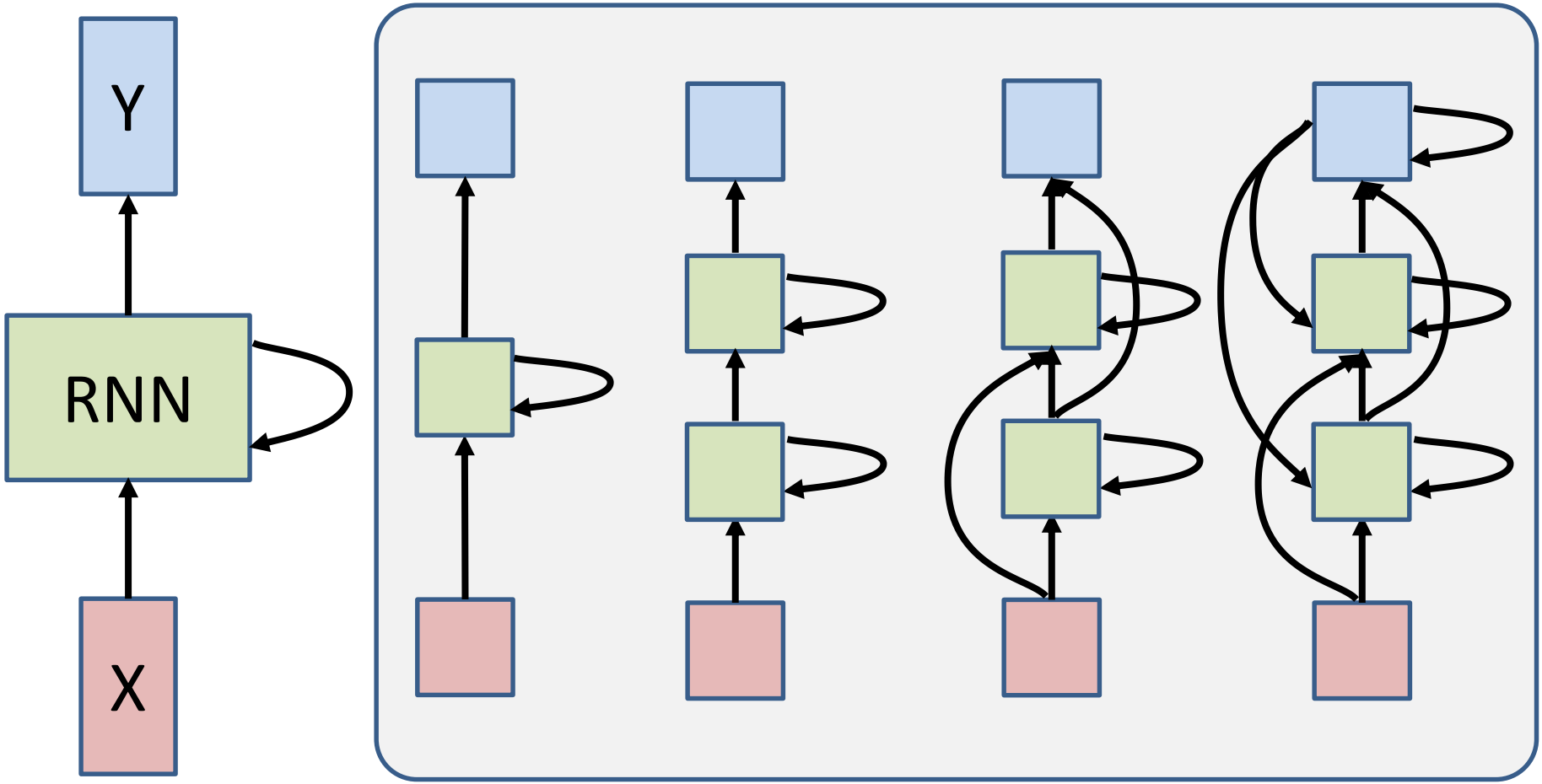
RNN: Computational Graph (Many to One)



RNN: Computational Graph (One to Many)



RNN: other design



RNN can be designed very sophisticatedly with different layers different ways of recurrency

RNN

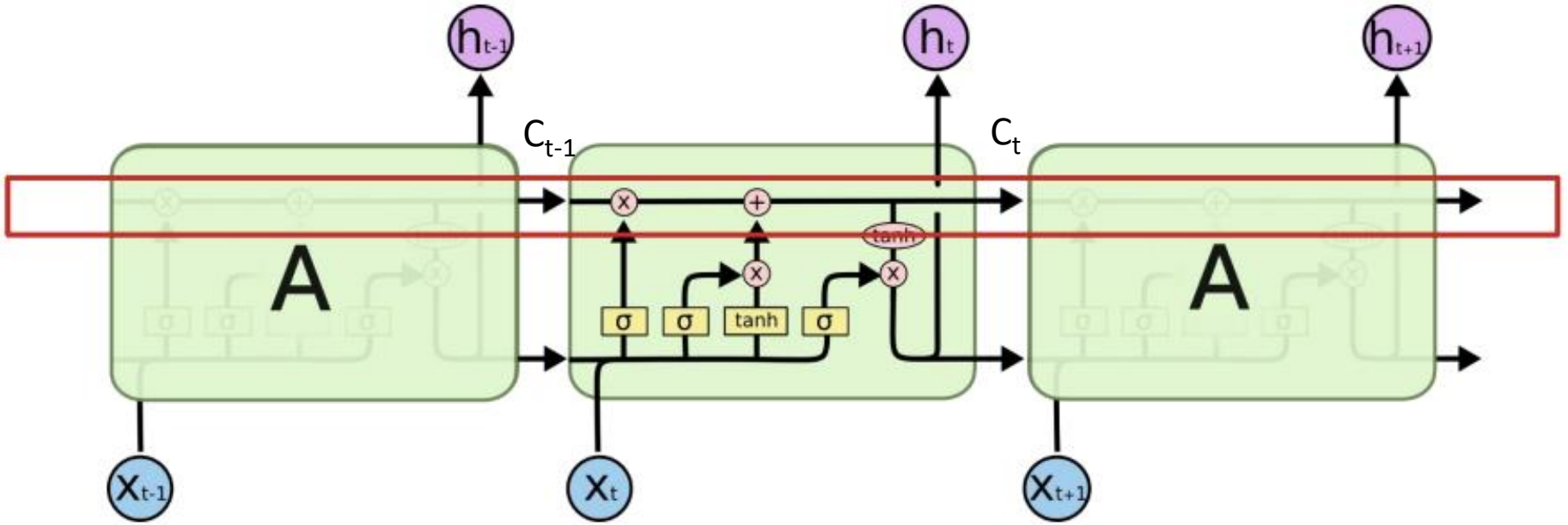
- In theory RNN retains information from the infinite past.
  - All past hidden state has influence on the future state.

Prolems of RNN

- In practice RNN has little response to the early states.
  - Little memory over what seen before.
  - The hidden outputs blowup or shrink to zeros.
  - The “memory” also depends on activation functions.
  - ReLU and Sigmoid do not work well. Tanh is OK but still not “memorize” for too long.

- Vanishing gradient problem
  - Deeper layers do not have meaningful weights.

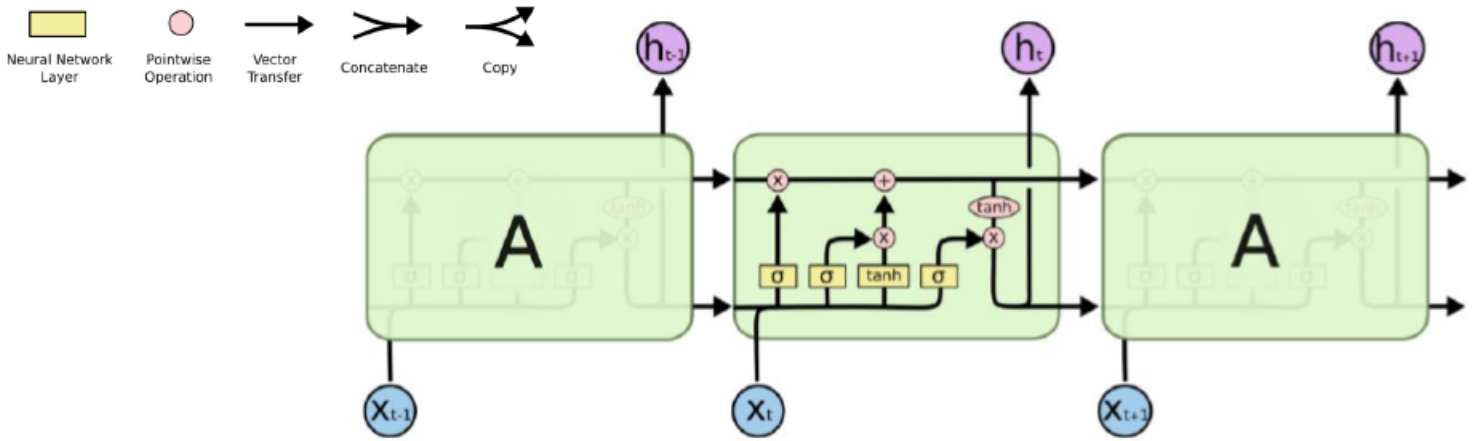
LSTM



Constant Error Carousel

- Key of LSTM: a remembered cell state
- C_t is the linear history carried by the constant error carousel.
- Carries information through and only effected by a gate
  - Addition of history (gated).

LSTM

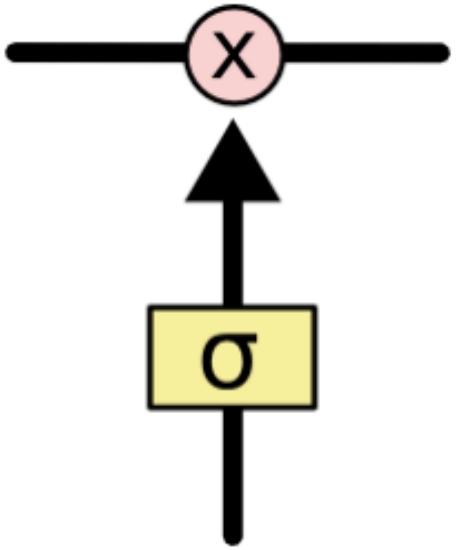


Bob and Alice are having lunch. Bob likes apples. Alice likes oranges.
She is eating an orange.

Conveyer belt for **previous state** and **new data**:

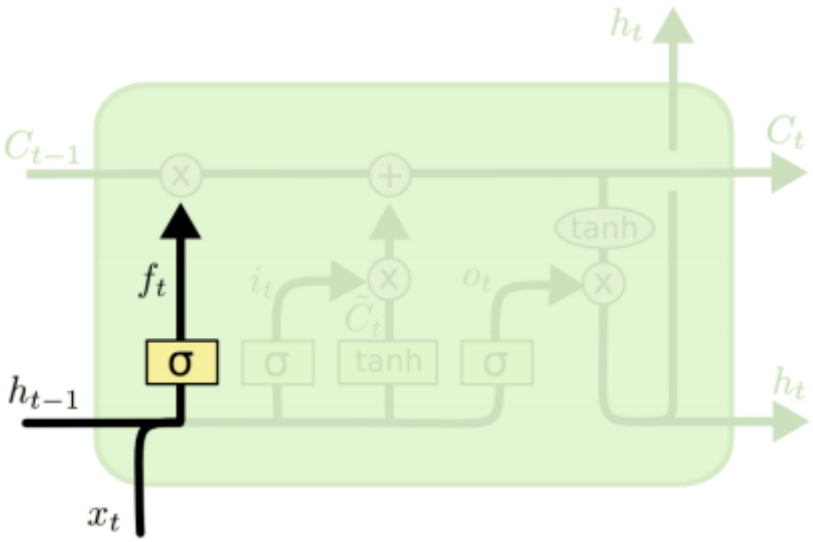
1. Decide what to forget (state)
2. Decide what to remember (state)
3. Decide what to output (if anything)

LSTM - Gate



- A simple sigmoid function to project output in range (0, 1).
  - Information is let through (~1)
  - Information is not let through (~0)
- ⊗: element-wise multiplication.

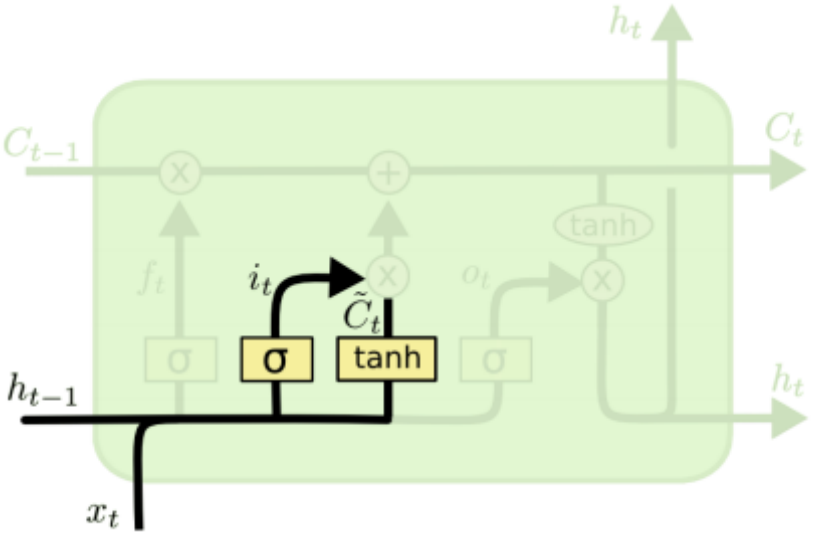
LSTM – Forget Gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- The first gate determines whether to carry over the history or forget it
  - Called “forget” gate.
  - Actually, determine how much history to carry over.
  - The memory C and hidden state h are distinguished.

LSTM – Input Gate



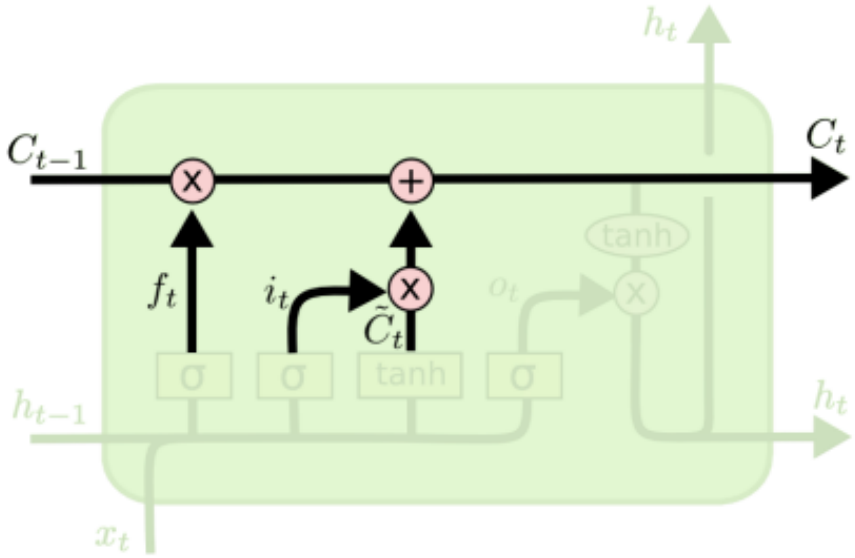
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

The second gate has two parts

- A *tanh* unit determines if there is something new or interesting in the input.
- A gate decides if it is worth remembering.

LSTM – Memory Cell Update

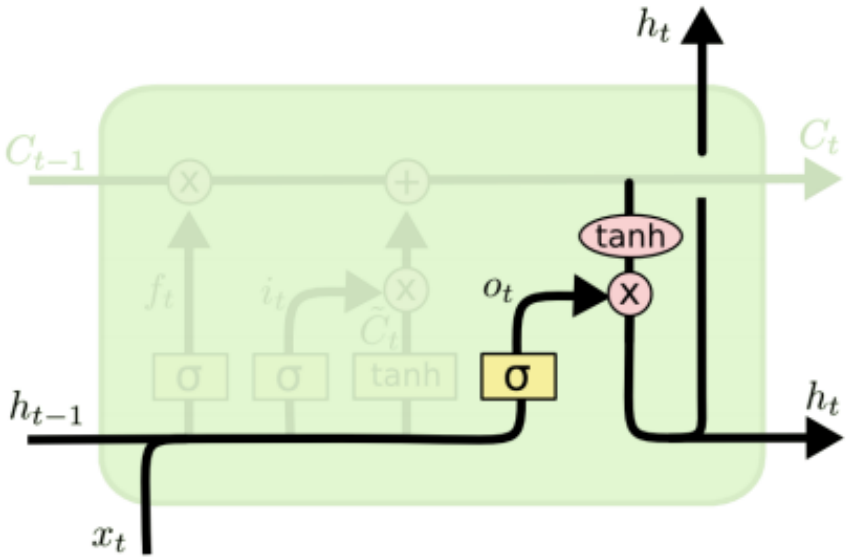


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Add the output of input gate to the current memory cell

- After the forget gate.
- $\bigoplus$: Element-wise addition.
- Perform the forgetting and the state update

LSTM – Output and Output Gate



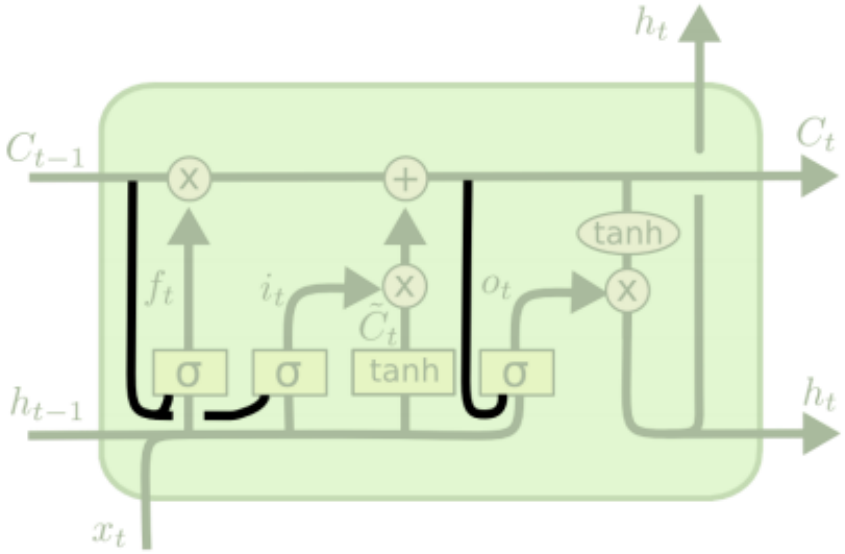
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

The output of the memory cell

- Similar to input gate.
- A *tanh* unit over the memory to output in range [-1, 1].
- A *sigmoid* unit [0,1] decide the filtering.
- Note the memory is carried through without *tanh*.

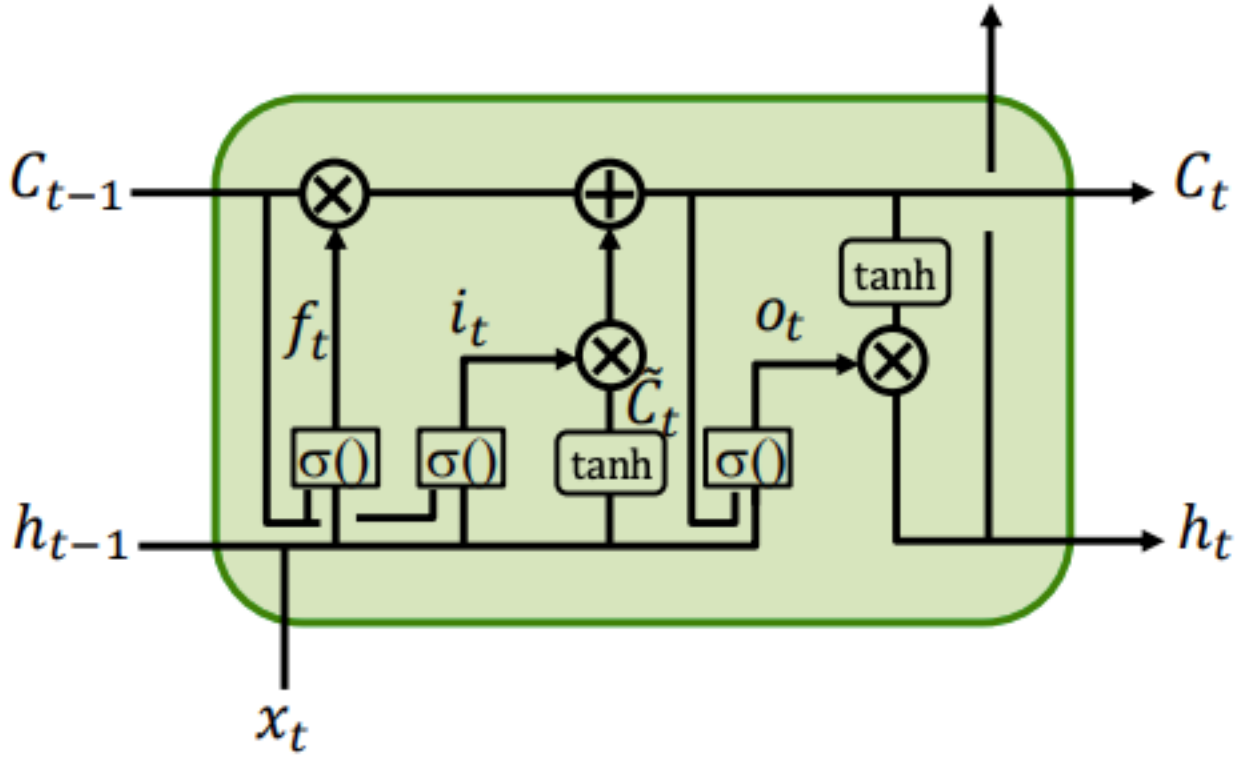
LSTM – the "Peephole" Connection



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Let the memory cell directly influence the gates!

The Complete LSTM Unit



Input, output, forget gates with peephole connection

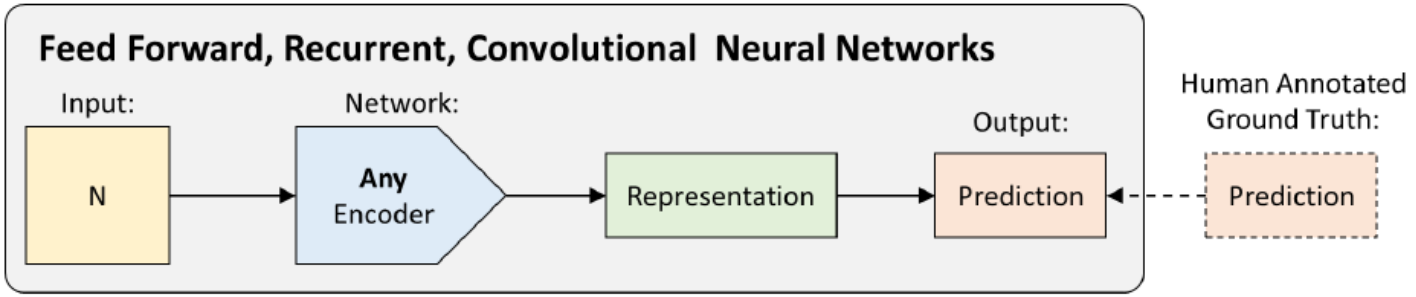
University of Nottingham
UK | CHINA | MALAYSIA

COMP3055
Machine Learning

Topic 17 – Deep Reinforcement Learning Intro

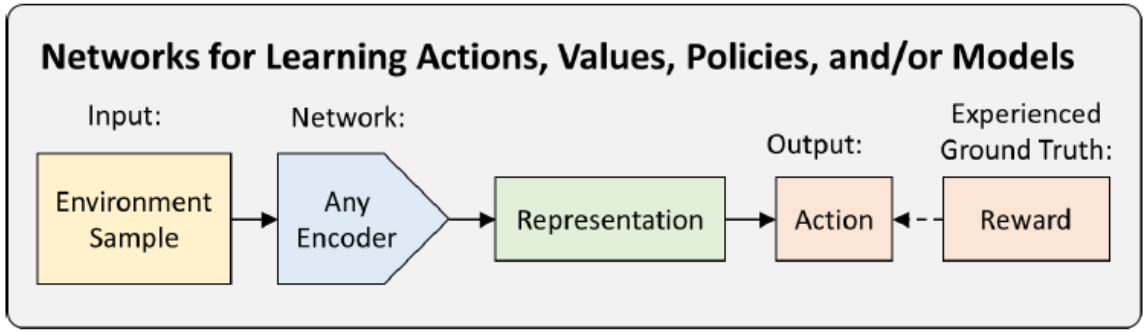
Zheng Lu
2024 Autumn

Deep Reinforcement Learning



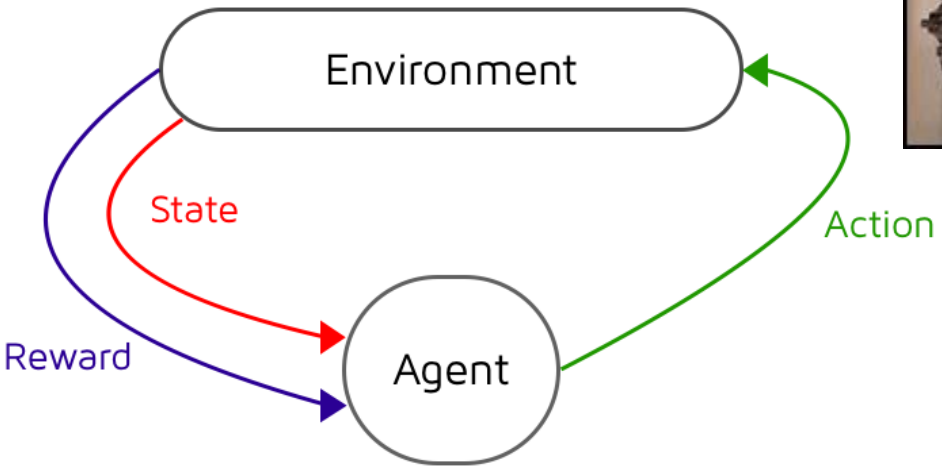
Supervised learning is "teach by **example**":
Here's some examples, now learn patterns in these example.

Reinforcement learning is "teach by **experience**":
Here's a world, now learn patterns by exploring it.



RL Framework

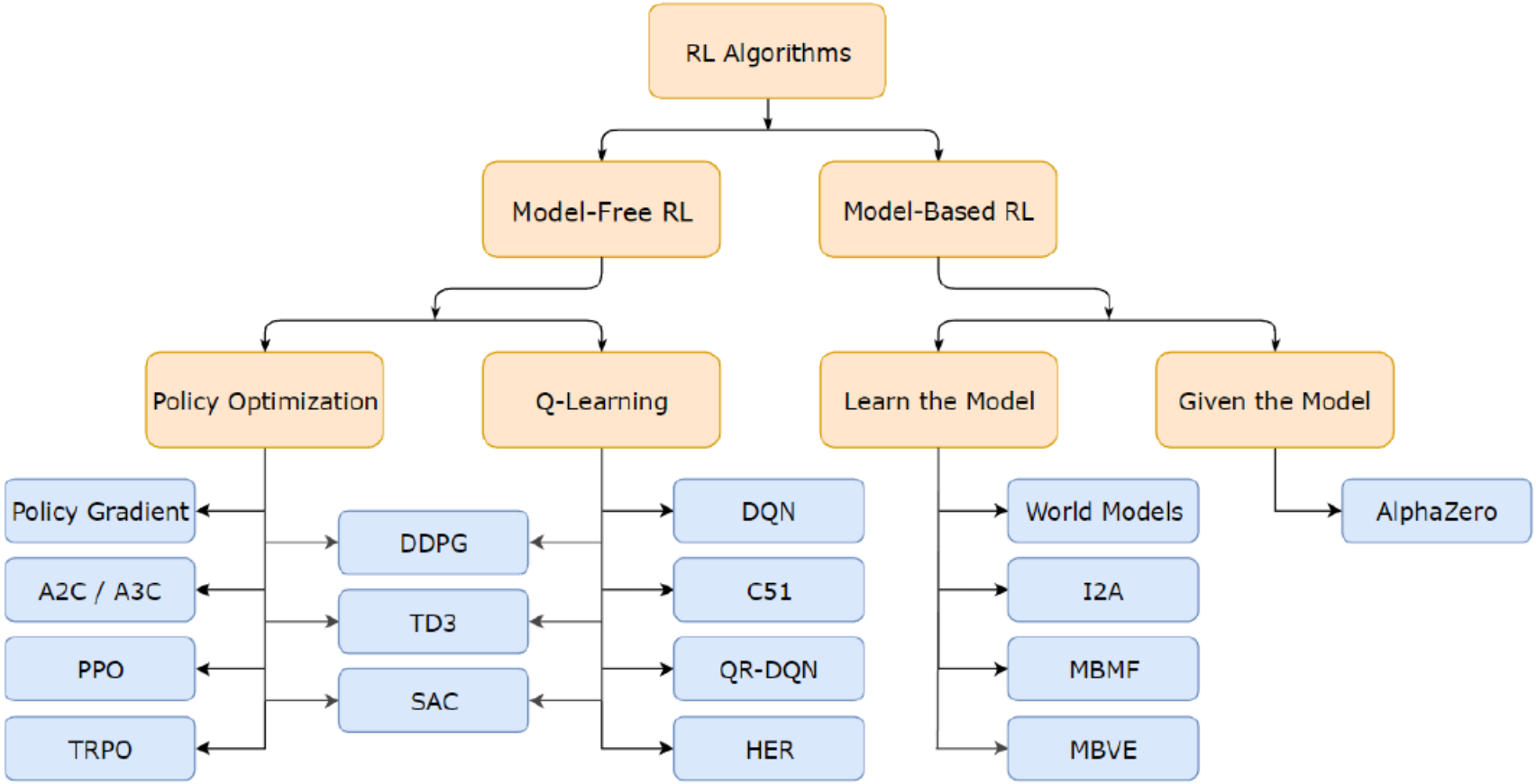
- At each step, the agent should:
  - Executes **action**
  - Observes new **state**
  - Receives **reward**



RL Methods

- **Model-based**
  - Learn the model of the world, then plan using the model
  - Update model often
  - Re plan often
- **Value-based**
  - Learn the state or state action value
  - Act by choosing best action in state
  - Exploration is a necessary add on
- **Policy-based**
  - Learn the stochastic policy function that maps state to action
  - Act by sampling policy
  - Exploration is baked in

RL Methods


RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero

Actor-Critic

- Combine DQN (value-based) and REINFORCE (policy-based)
- Two neural networks (Actor and Critic):
  - **Actor** is policy-based: Samples the action from a policy
  - **Critic** is value-based: Measures how good the chosen action is

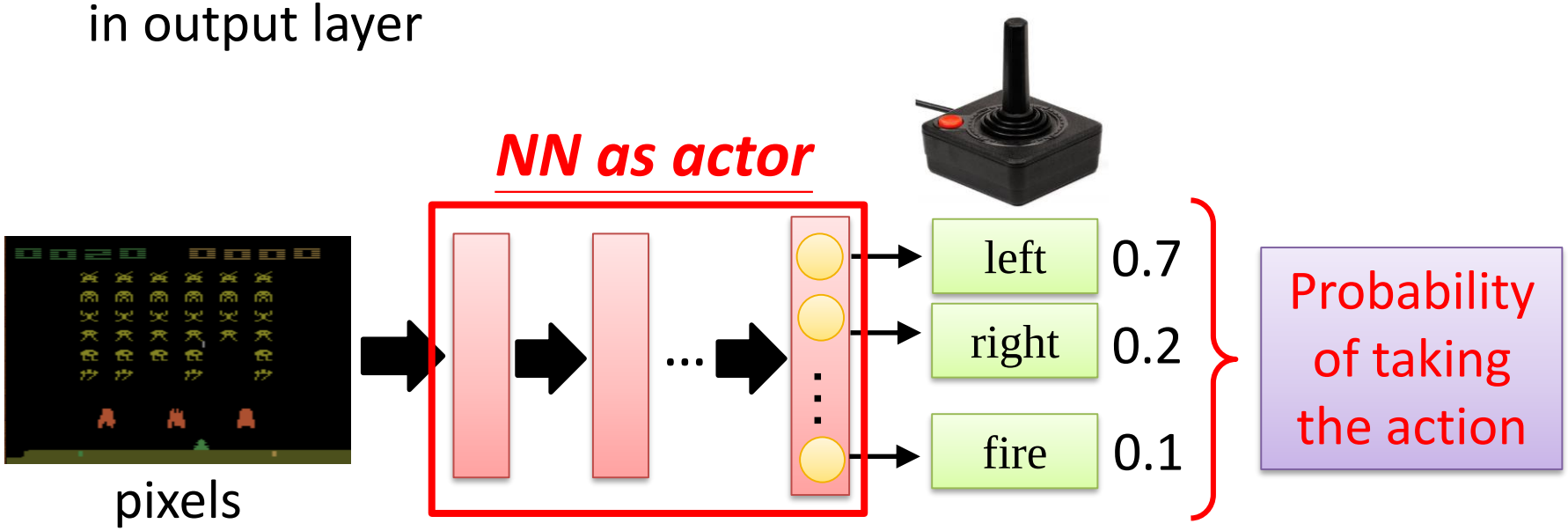
Policy Update: $$\Delta\theta = \alpha * \nabla_\theta * (log\ \pi(S_t, A_t, \theta)) * R(t)$$

New update: $$\Delta\theta = \alpha * \nabla_\theta * (log\ \pi(S_t, A_t, \theta)) * Q(S_t, A_t)$$

- Update at each time step - temporal difference (TD) learning

Neural network as Actor

- Input of neural network: the observation of machine represented as a vector or a matrix
- Output neural network : each action corresponds to a neuron in output layer

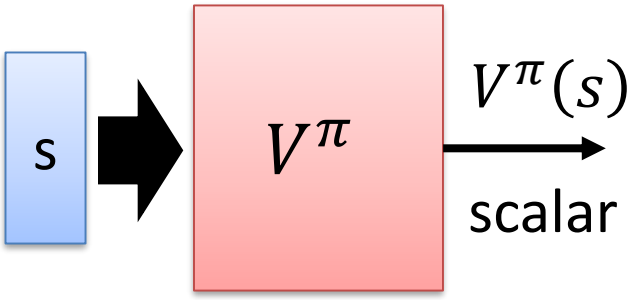


Critic

- A critic does not determine the action.
- Given an actor, it evaluates the how good the actor is

Critics

- A critic is a function depending on the actor π it is evaluated
  - The function is represented by a neural network
- State value function $V^{\pi}(s)$
  - When using actor π, the *cumulated* reward expects to be obtained after seeing observation (state) s

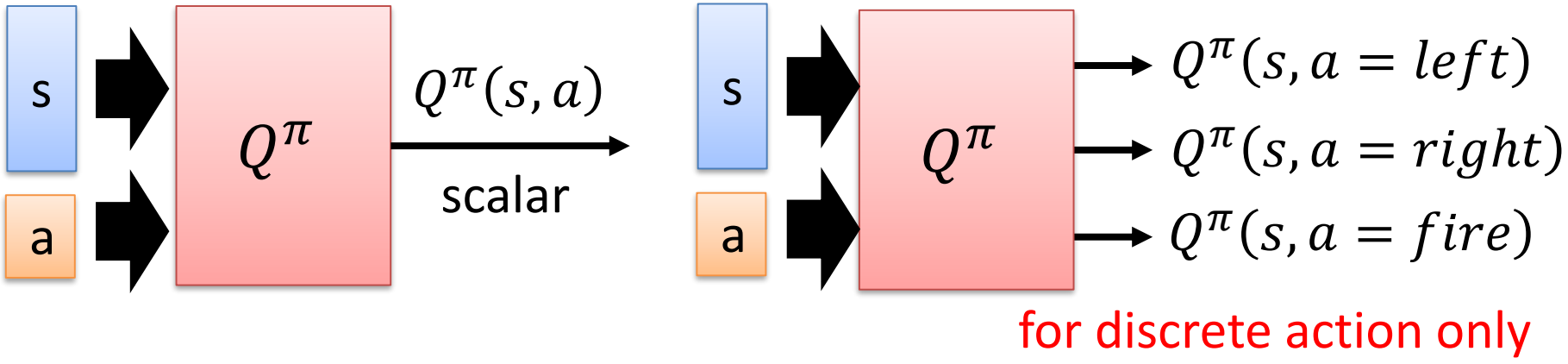


$V^{\pi}(s)$ is large

$V^{\pi}(s)$ is smaller

Critics

- State-action value function $Q^{\pi}(s, a)$
  - When using actor π, the *cumulated* reward expects to be obtained after seeing observation s and taking a

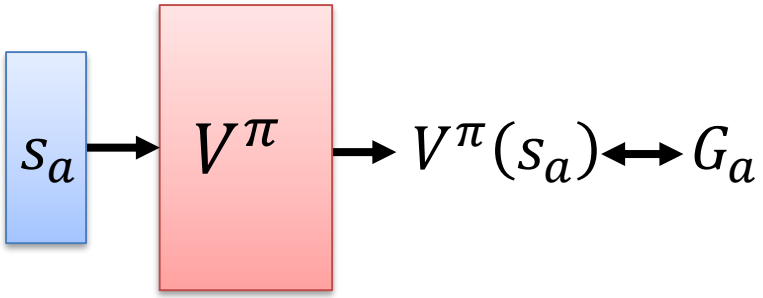


How to estimate $V^{\pi}(s)$

- Monte-Carlo based approach
  - The critic watches π playing the game

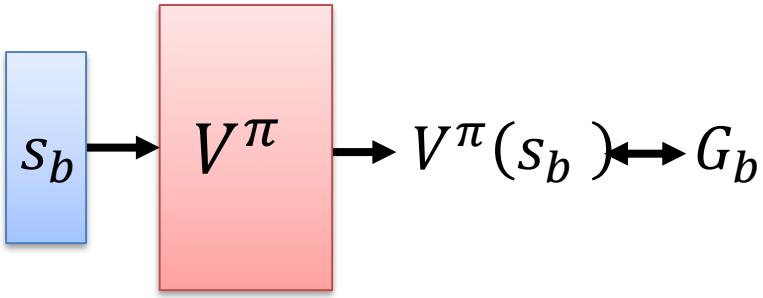
After seeing s_a,

Until the end of the episode, the cumulated reward is G_a



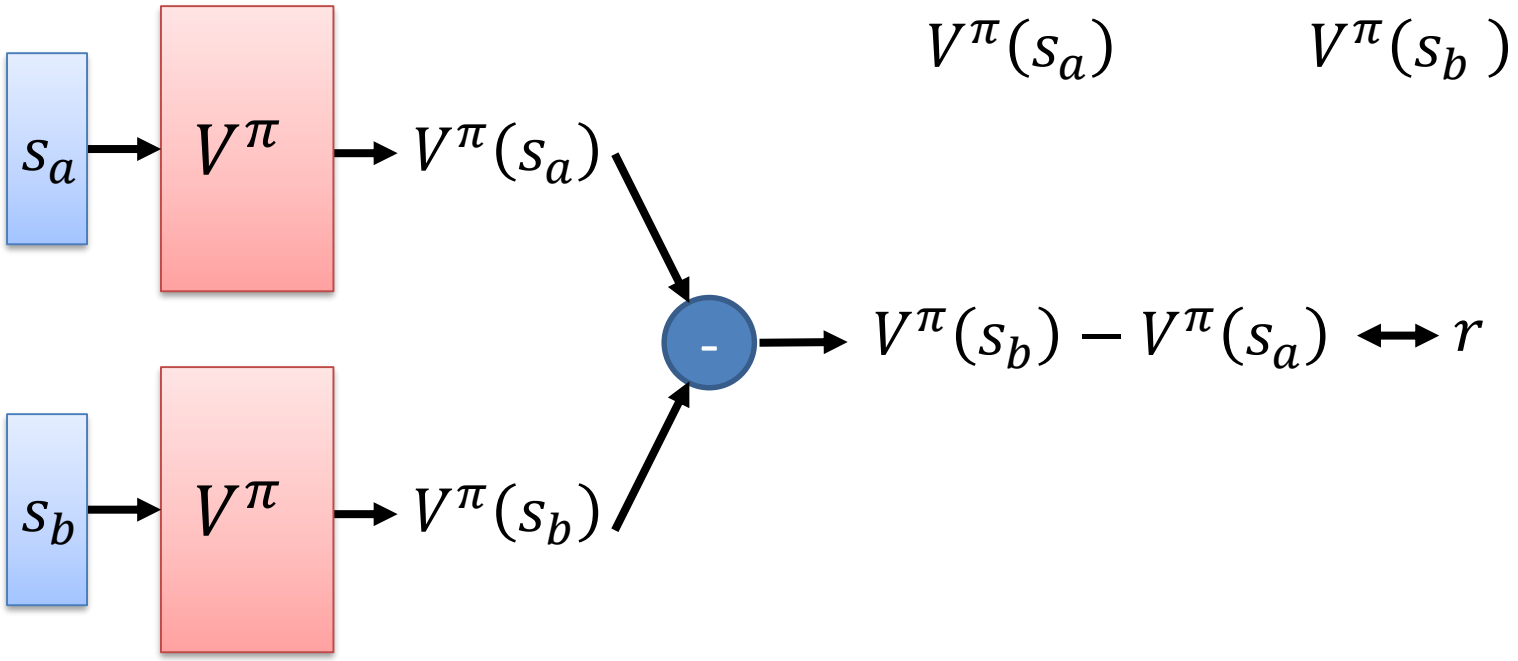
After seeing s_b,

Until the end of the episode, the cumulated reward is G_b



How to estimate $V^{\pi}(s)$

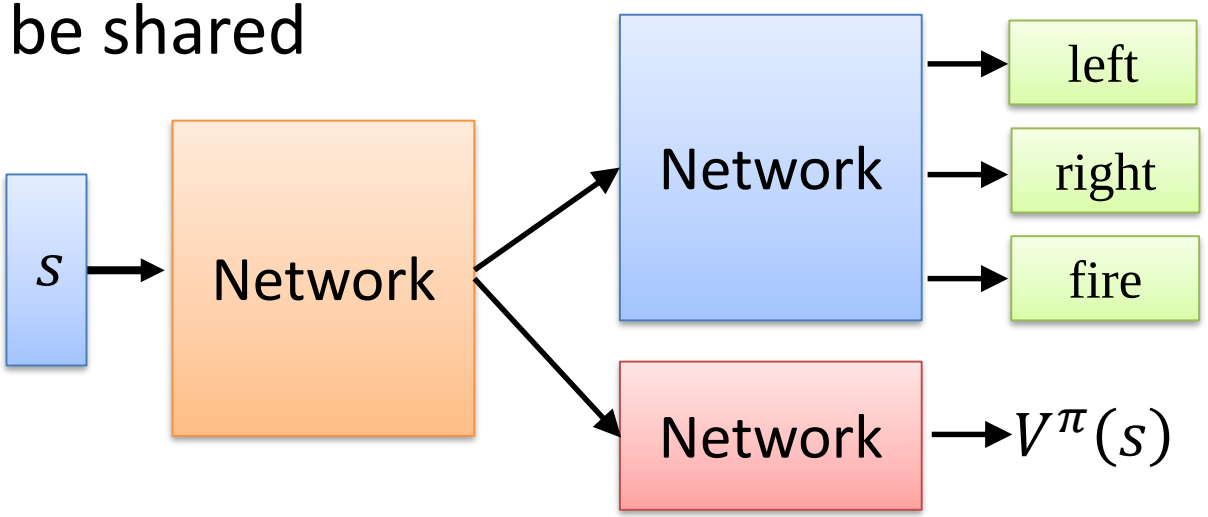
- Temporal-difference approach $\ldots s_a, a, r, s_b \ldots$



Some applications have very long episodes, so that delaying all learning until an episode's end is too slow.

Actor-Critic

- Tips
  - The parameters of actor $\pi(s)$ and critic $V^{\pi}(s)$ can be shared



  - Use output entropy as regularization for $\pi(s)$
    - Larger entropy is preferred → exploration

University of Nottingham
UK | CHINA | MALAYSIA

COMP3055 Machine Learning

Topic 18 – Others

Zheng Lu

2024 Autumn

Pre-training and Fine-tuning

- Practical situation
  - You have some specific task that you want to apply machine learning techniques, for example, classifying medical dialogues into certain medical categories.
  - You only have a tiny labelled dataset to train with.
  - There is a general dataset with large number of labelled data.

slide credit: M. Gormly, J. Hewitt, G. Durrett, A. Goldie

Pre-training and Fine-tuning

- Pre-training
  - Train the large network models on a large amount of general data
  - Save the network parameters
  - Learn general things

Pre-training and Fine-tuning

- Fine-tuning
  - Initialize the large network model using parameters trained during pre-training.
  - Continue training with additional data (usually small number and task-specific).
  - Adapt to the task.