CG3002

Embedded Systems Design Project

L4: System Architecture & Machine Learning Basics

LECTURER:

Wang Ye

wangye@comp.nus.edu.sg

Office: AS6-04-08

Some slides are adopted and modified from Min Yen Kan, John Canny, Nathan Bastian, etc.

What is Machine Learning?

Addresses the question of how to build computer programs or algorithms that improve their performance at some task through experience.

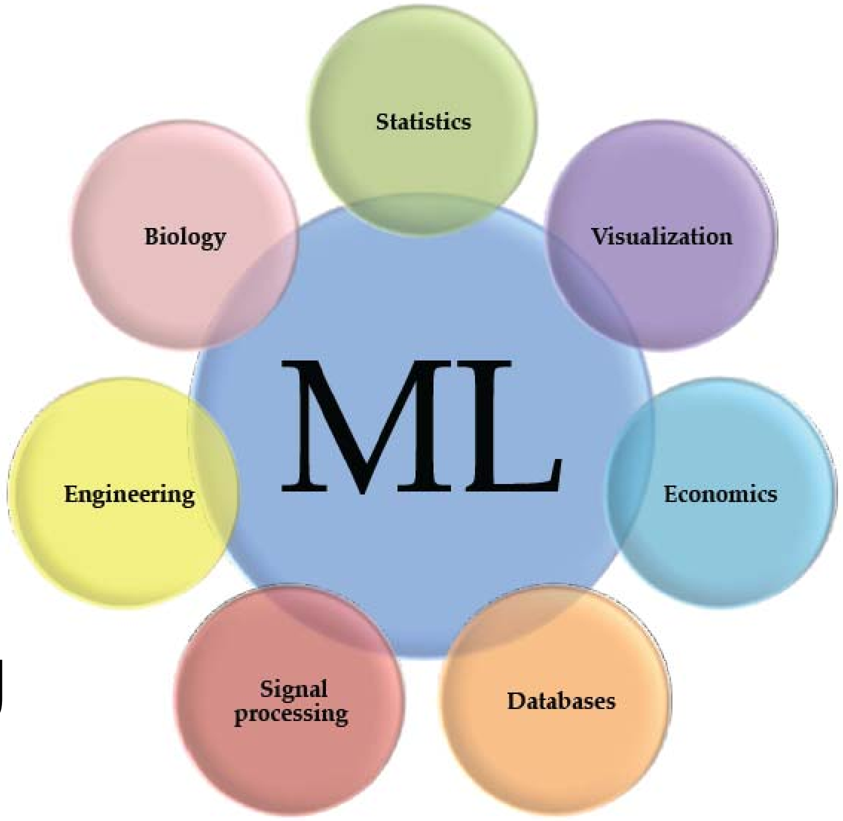
Or more simply, we try to make computers learn to solve problems.

(e.g., Computer Vision-Based Sorting of Atlantic Salmon)

Examples of ML Problems

- Spam Detection
- Face Detection
- OCR
- Voice Recognition
- Medical Diagnosis
- Weather Forecasting
- *Activity Recognition*

And Many More…



How can Machines Learn to Solve Problems?

Learn from nature: humans or animals

Humans learn from experience to improve ourselves…

Can we also let machines learn from experience?

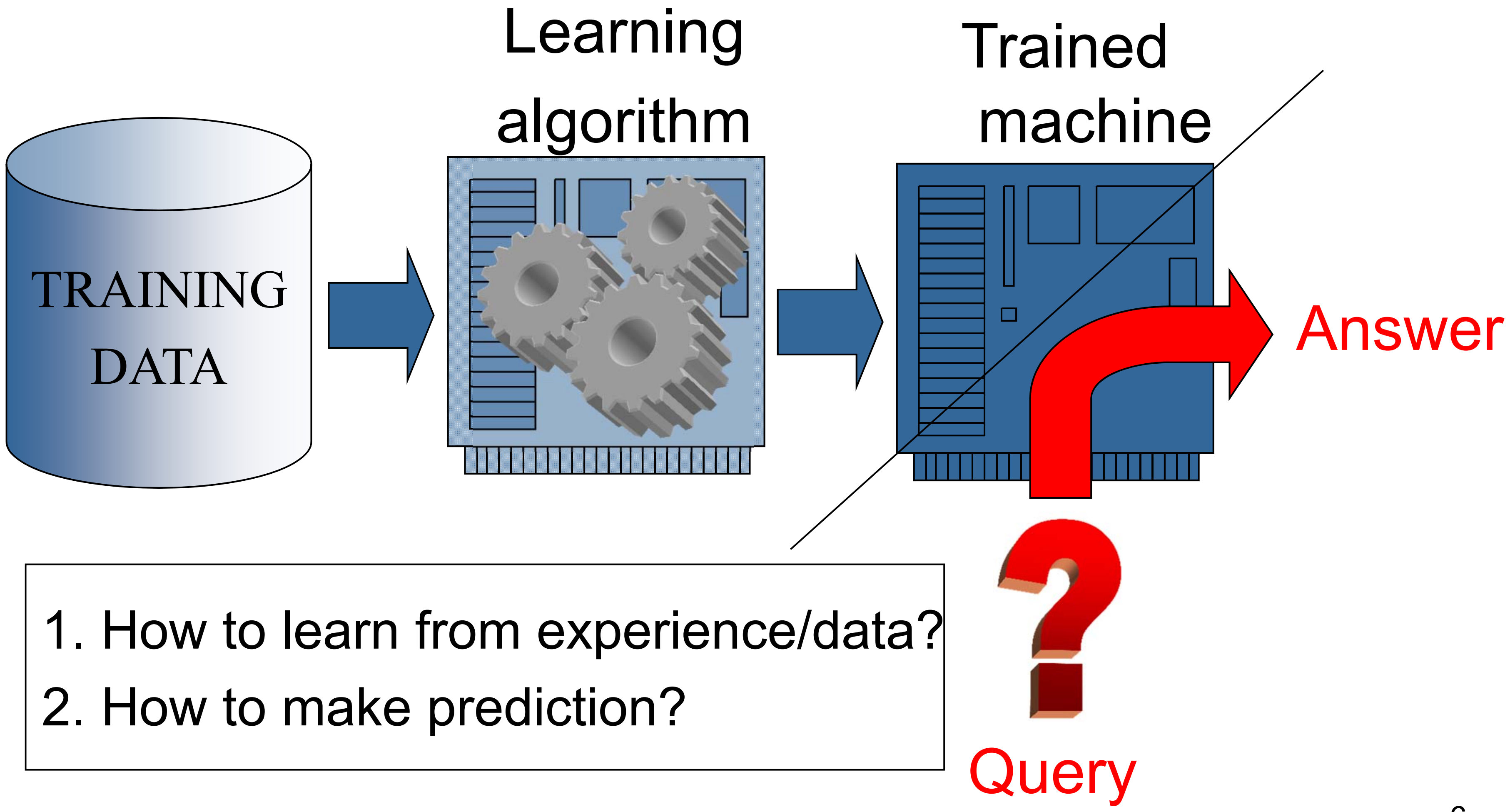
The essence of learning problems

1. A pattern exists

2. Difficult to pin down formally/analytically/mathematically

3. We have data for it (e.g., in our memory)

A Machine Learning Framework
(general)



1. How to learn from experience/data?
2. How to make prediction?

Problem Formalism

Formalization

- Input: $\boldsymbol{x}$ *(sensor data)*
- Output: y *(dance move?)*
- Target Function: $f : \mathcal{X} \to \mathcal{Y}$ *(ideal formula)*
- Data: $\mathcal{D} = (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)$ *(annotated training data)*

- Hypothesis: $g : \mathcal{X} \to \mathcal{Y}$ *(model to be used)*

$\mathcal{X},\ \mathcal{Y}$ and $\mathcal{D}$ are given by the learning problem;
The target f is unknown (to be learned).

We learn the function f from the data $\mathcal{D}$.

Solution Components

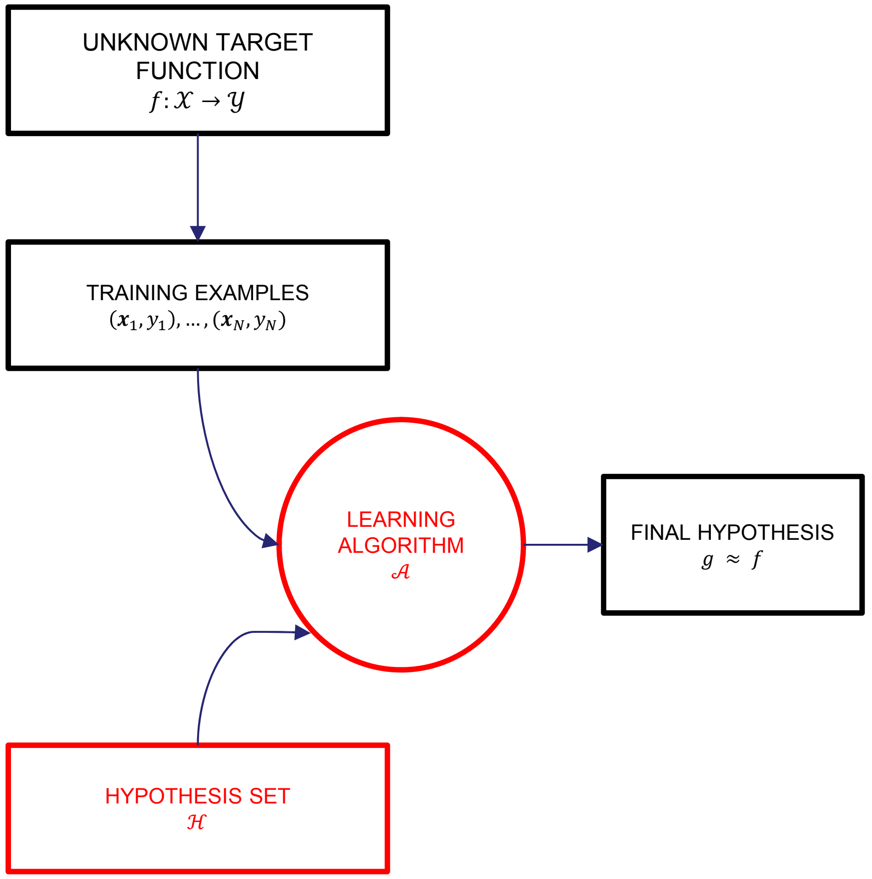
There are 2 components of the learning algorithm:

1. The ***Hypothesis set***
   $\mathcal{H} = \{h_1, h_2, \ldots, h_N\}$

2. The ***Learning algorithm*** $\mathcal{A}$
   Selects $g \in \mathcal{H}$

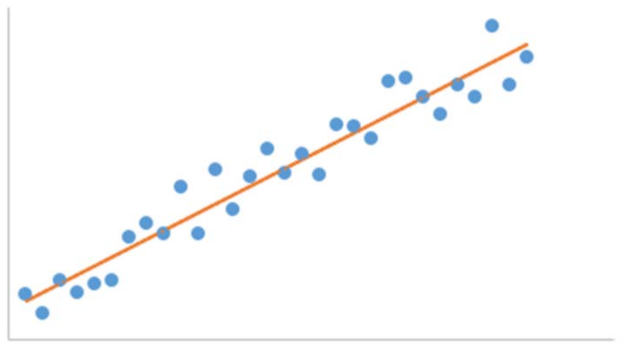
Together they are referred to as the *learning model*.

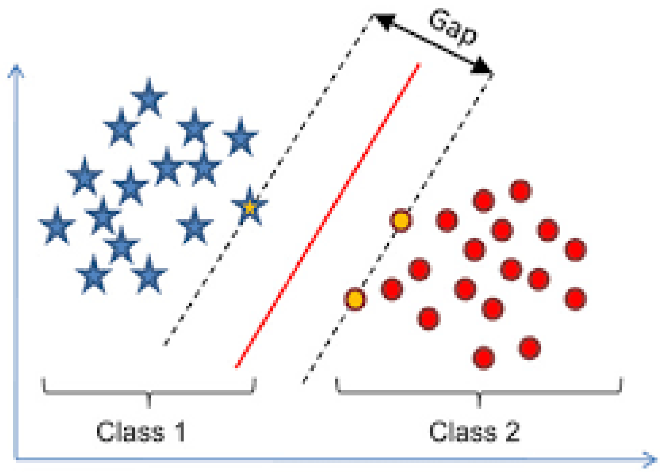
We chose $\mathcal{H}$ and the learning algorithm $\mathcal{A}$ that chooses g


UNKNOWN TARGET FUNCTION $f: \mathcal{X} \to \mathcal{Y}$
TRAINING EXAMPLES $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$
LEARNING ALGORITHM $\mathcal{A}$
FINAL HYPOTHESIS $g \approx f$
HYPOTHESIS SET $\mathcal{H}$

Terminology

Statistical modeling: In simple terms, statistical modeling is a simplified, mathematically-formalized way to approximate reality (i.e., what generates your data) and optionally to make predictions from this approximation. The statistical model is the mathematical equation that is used.


Regression (curve fitting)



Classification

Terminology

Pattern recognition: is the process of classifying input data into objects or classes based on key *features*. There are two classification methods in pattern recognition: supervised and unsupervised classification.

Classification: In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.

Regression analysis is a statistical process for estimating the relationships among variables. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors').

Q: Is a human activity recognition a classification or regression problem?

Types of Learning

Supervised Learning

Unsupervised Learning

Semi-supervised Learning

Reinforcement Learning

We will concentrate on

Supervised Learning

Activity Recognition

- Enable new forms of human-computer interaction
- Detect social interactions
- Assist diagnosis and monitoring of patients

Or… track people dancing?

Approaches

- Can use hand coded methods such as thresholding but…

- Lets make a computer learn the methods for us: A machine learning approach

Overview

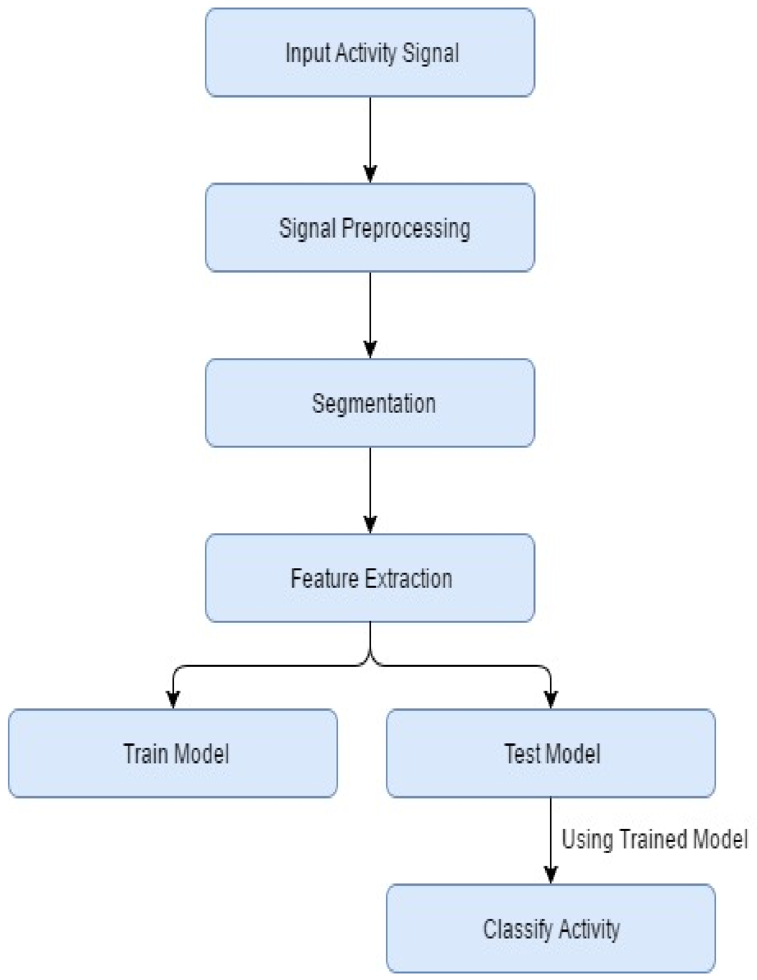
Sensor Inputs

Signal Pre-processing

Segmentation

Feature Extraction

Classification of Activity



Dataset

- Need to collect a dataset for supervised training
- Dataset will consist of activity raw data and label
- Will need many different examples of each activity for generalisation

Sensor Inputs

Each sensor provides raw inputs to the system.

Sensors can be of different types e.g. Accelerometers, Gyroscopes, Proximity Sensors, Vision Based Sensors e.g. Camera etc.

Each sensor may have more than one axis.

One or multiple sensors can be used to capture an activity or event.

Preprocessing

Different preprocessing techniques can be applied to the raw data to smooth the signal or remove the noise.

The signal can be normalized or scaled before being processed.

Different types of filters can be used:

- Low Pass Filters: To remove noise from the raw input signals
- High Pass Filters: To remove the effect of gravity from the raw input signals

Segmentation

Identifying the start and end of the actual activity from the pre processed signal.

Important for recognizing the activity accurately.

Different segmentation approaches can be used.

Another technique is to identify when the movement starts. It can be marked by a sudden change in accelerometer values.

Feature Selection

Different features can be selected to classify the activities.

Selecting good features which distinguish well between the activities and increase the accuracy of the algorithm.

Features can be extracted for each axis e.g. x, y and z.

Features are classified into two types:

- Time Domain Features
- Frequency Domain Features

Feature Selection

Time Domain Features:

- Mean
- Variance
- Median
- Mean Absolute Deviation

Feature Selection

Frequency Domain Features:

- Spectral Energy
- Entropy
- Peak Frequency
- Sub-band power

Potential Models

Support Vector Machine

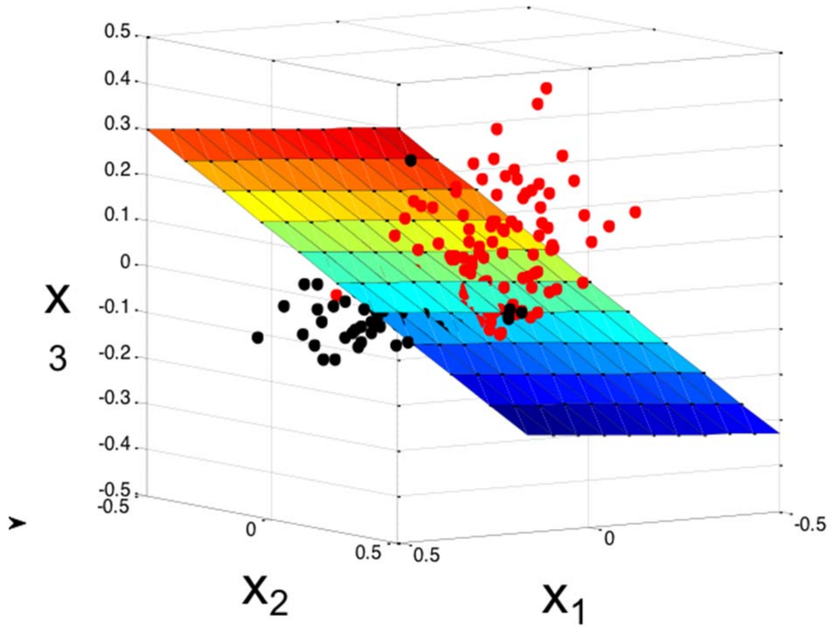
K-Nearest Neighbours

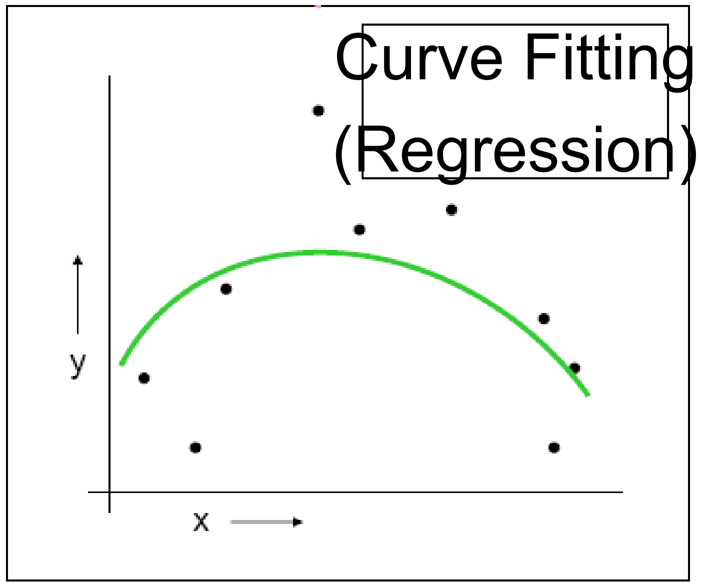
Feedforward Neural Network

Supervised Learning

- **Prediction** Problem: Given a training data set (Xi,Yi) i=1,2,…,N, learn a function that predicts Y given X with minimal probability of error.
- Regression (Y continuous)
- Classification (Y discrete)

Classification and Regression





Classification

Regression

Support Vector Machine (SVM)

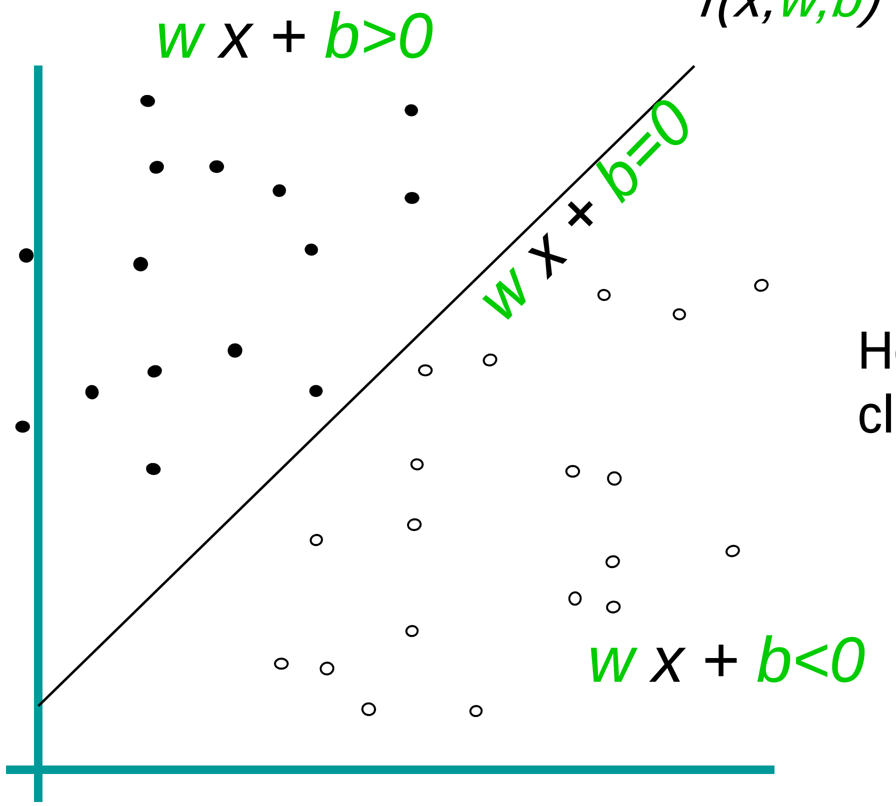
- Core idea: Maximize Margin
  - Hard Margin (Linear)
  - Kernel (Nonlinear)

Linear Classifiers

$$\begin{tikzcd} & \alpha \arrow[d] & \\ x \arrow[r] & \boxed{f} \arrow[r] & y^{est} \end{tikzcd}$$

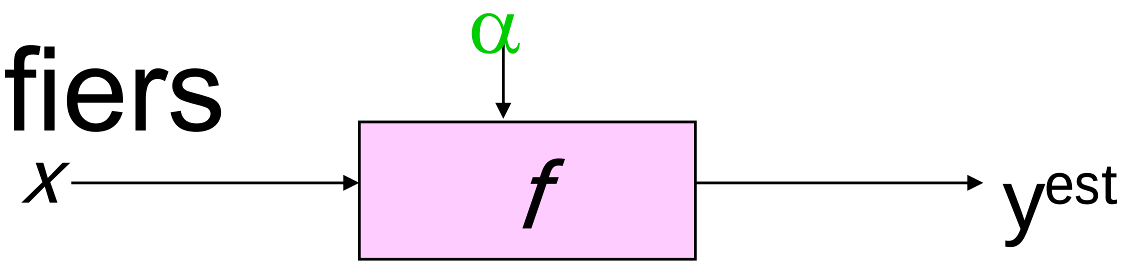
$f(x,w,b) = sign(w\,x + b)$

- ● denotes +1
- ○ denotes -1



How would you classify this data?

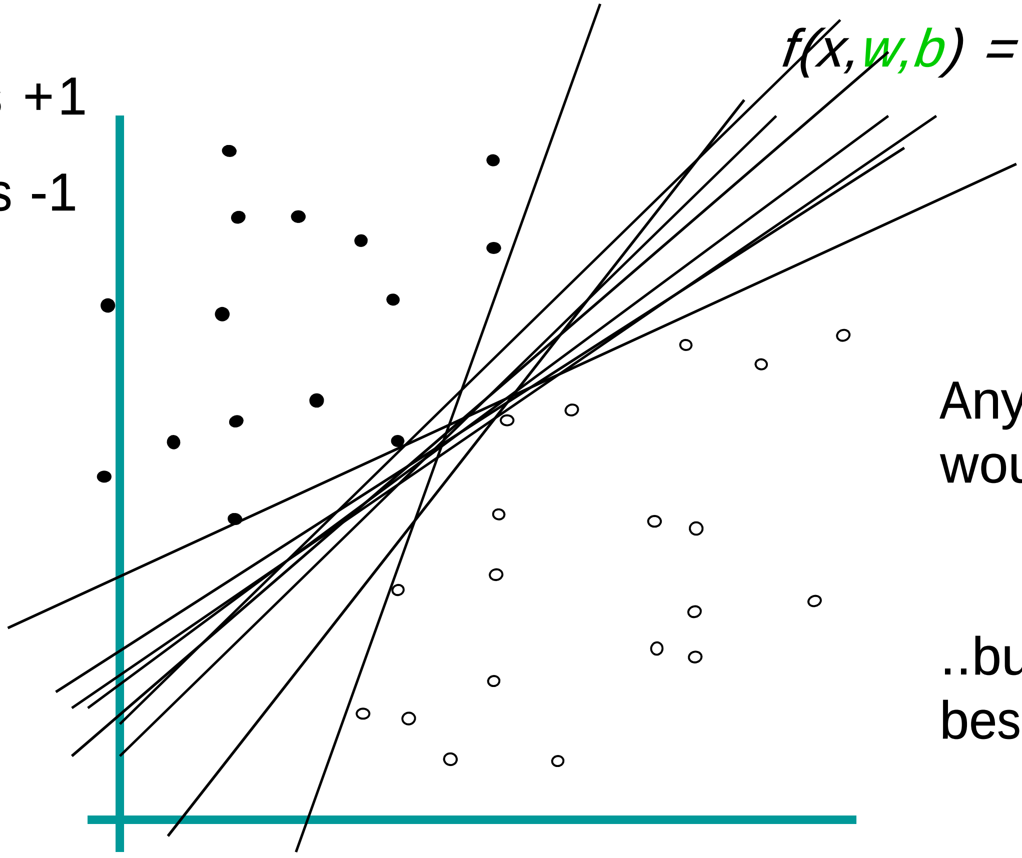
Linear Classifiers



$$x \xrightarrow{} \boxed{f} \xrightarrow{} y^{est}$$

$f(x, w, b) = \text{sign}(w\, x + b)$

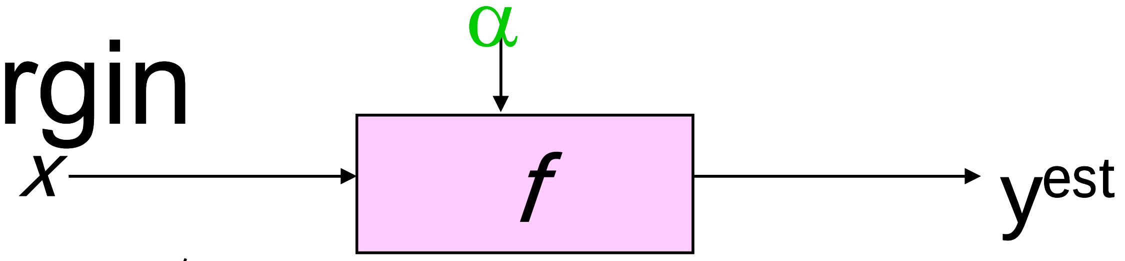
- ● denotes +1
- ○ denotes -1



Any of these would be fine..

..but which is best?

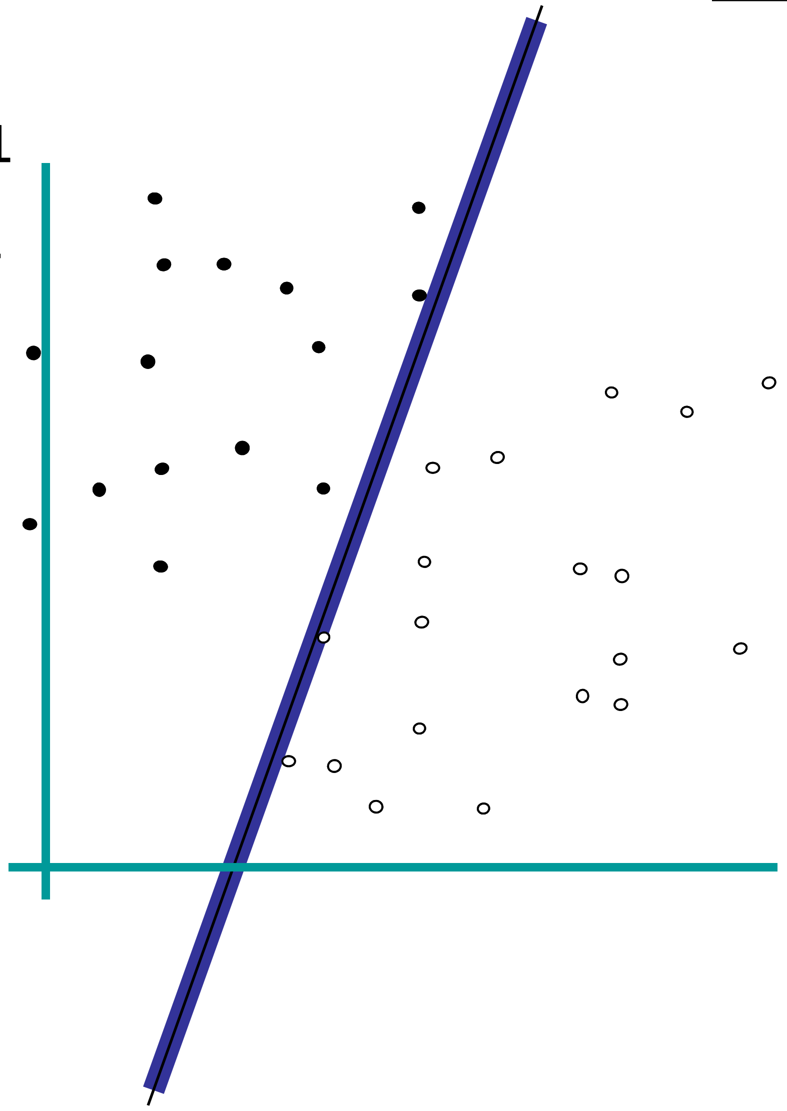
Classifier Margin



$f(x,w,b) = sign(w\,x + b)$

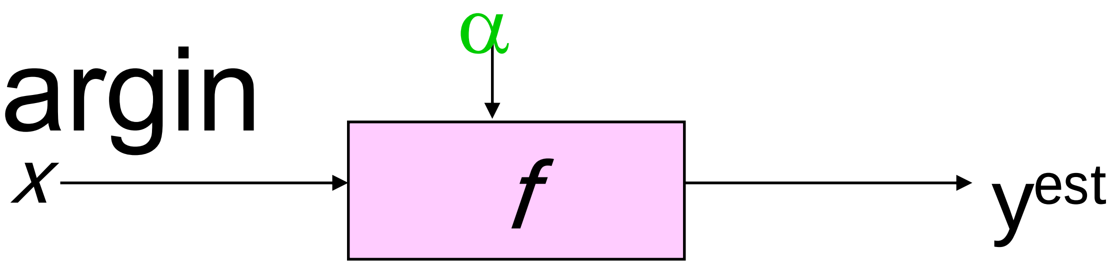
● denotes +1

○ denotes -1

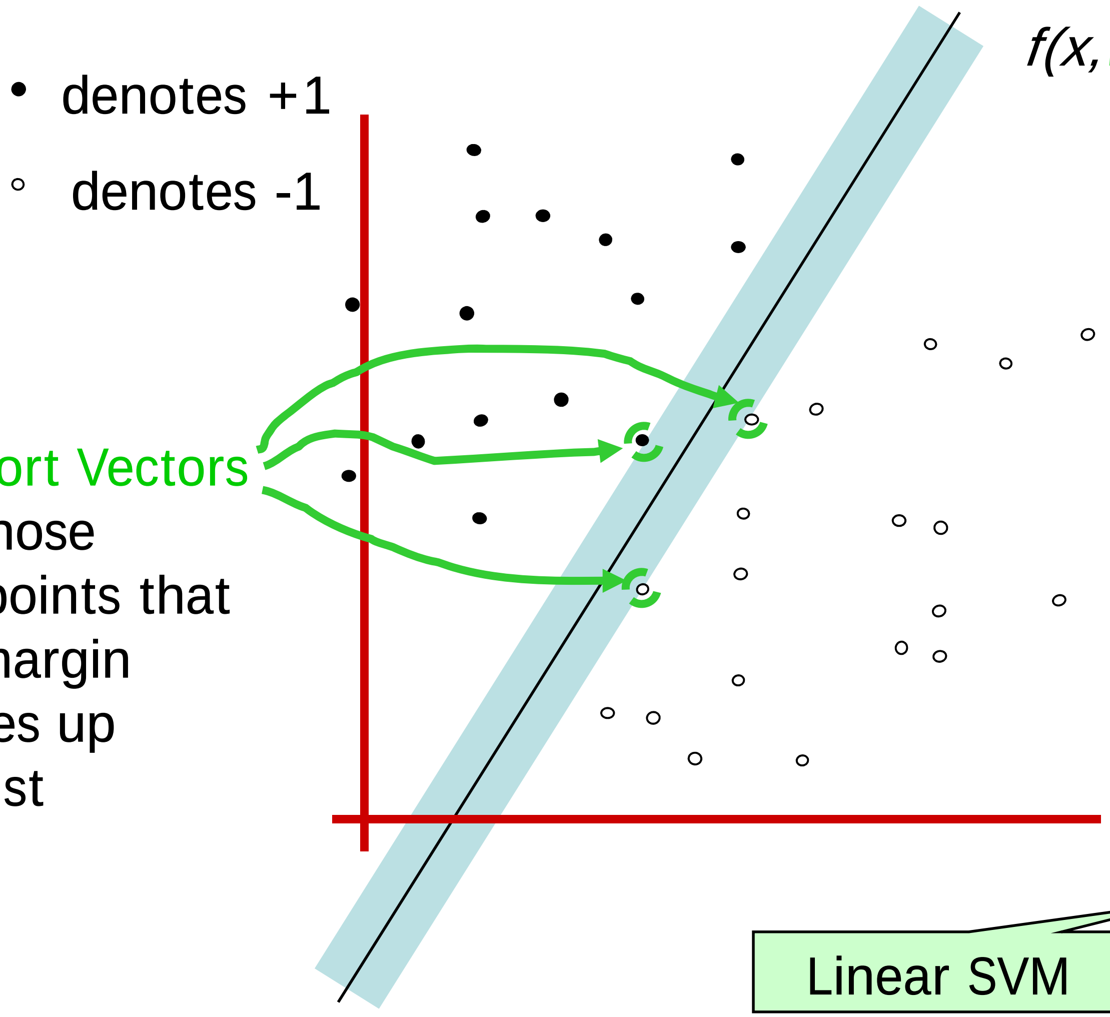


Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Maximum Margin



$f(x,w,b) = sign(w\,x + b)$



Support Vectors are those datapoints that the margin pushes up against

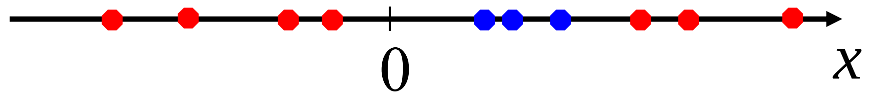
The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

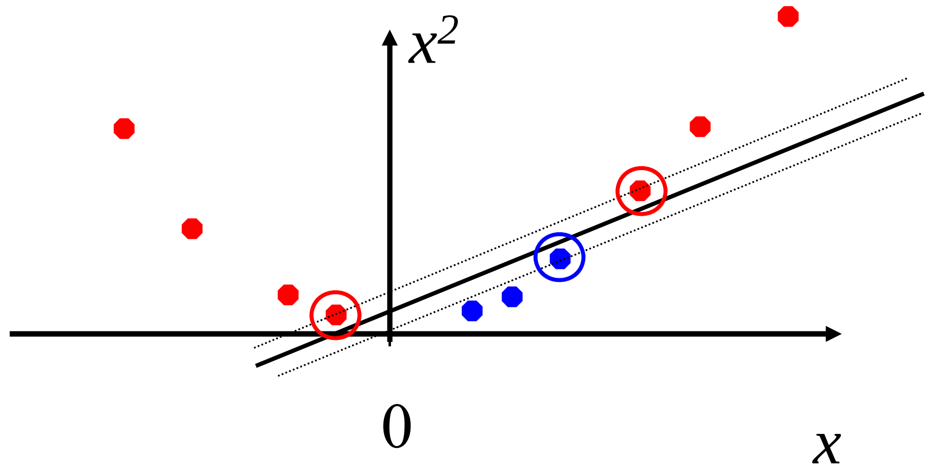
Linear SVM

Non-linear SVMs

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

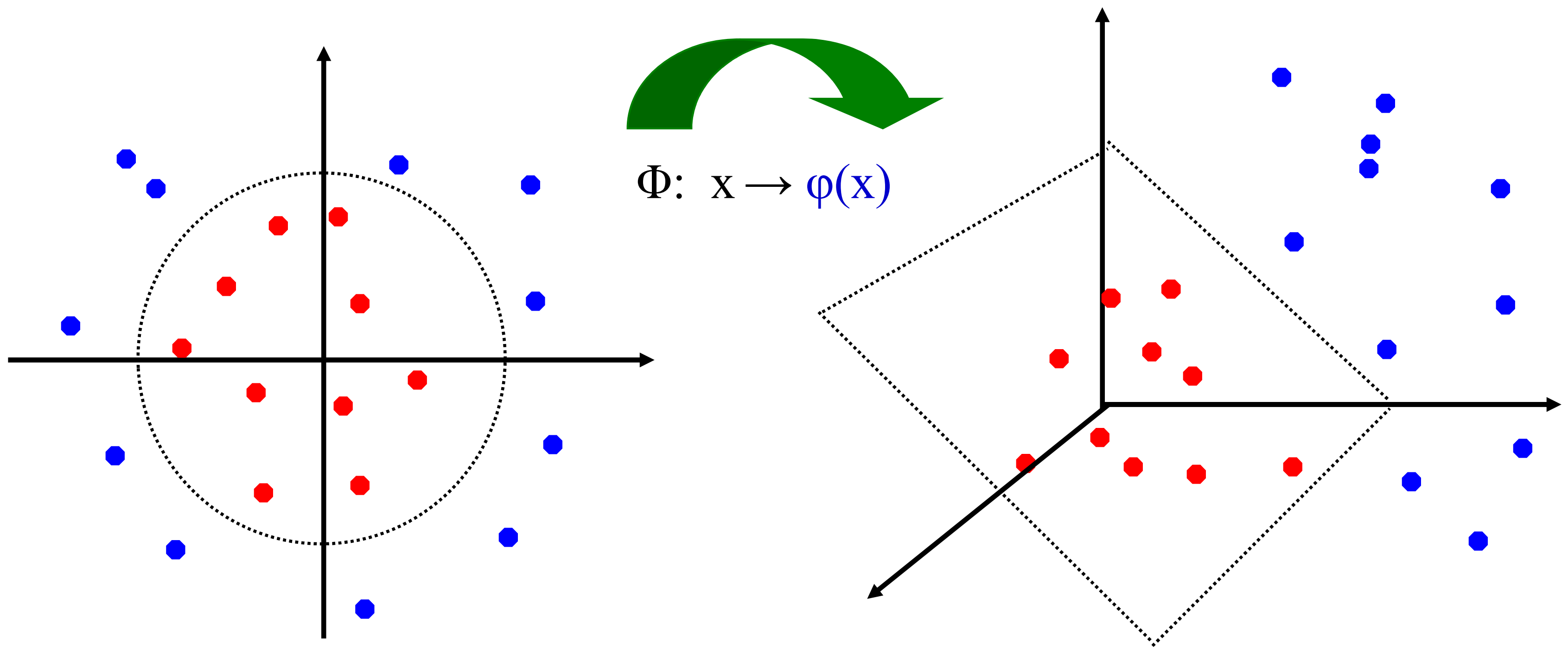


- How about… mapping data to a higher-dimensional space:



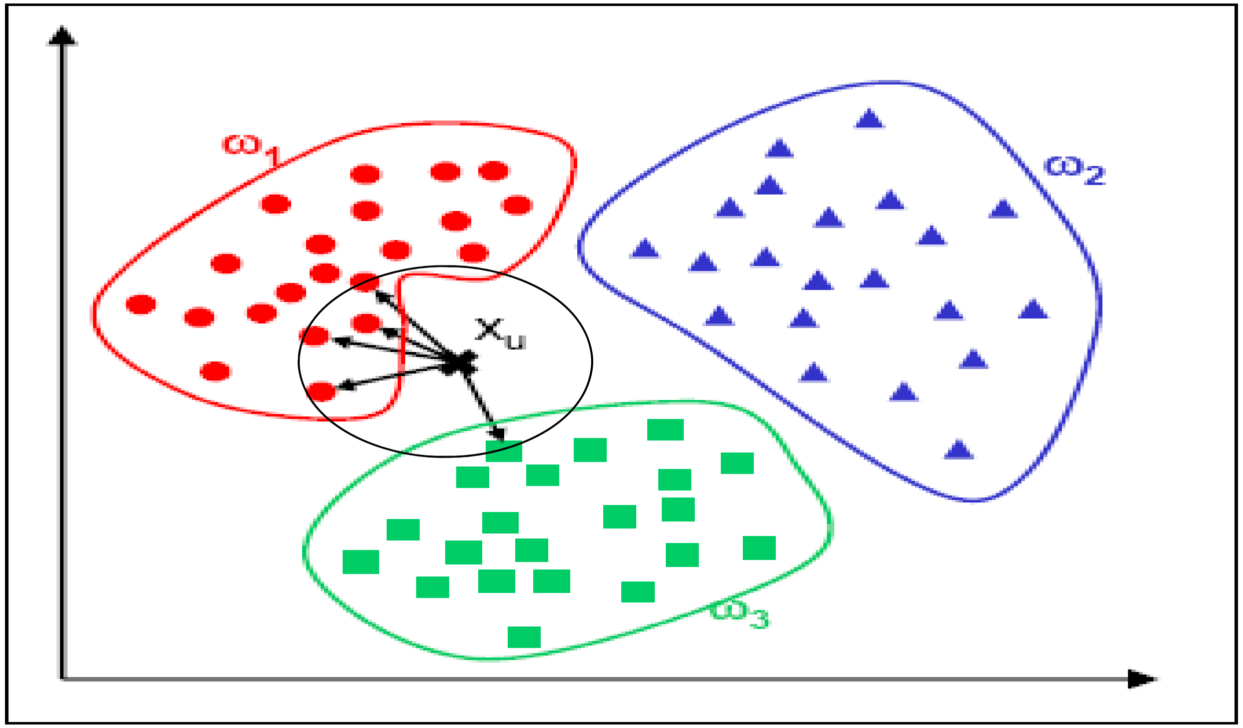
Non-linear SVMs: Feature space

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



K-Nearest Neighbor Classifier

- **Classify unlabeled data according to the dominate class of the k nearest neighbors in the training data set.**

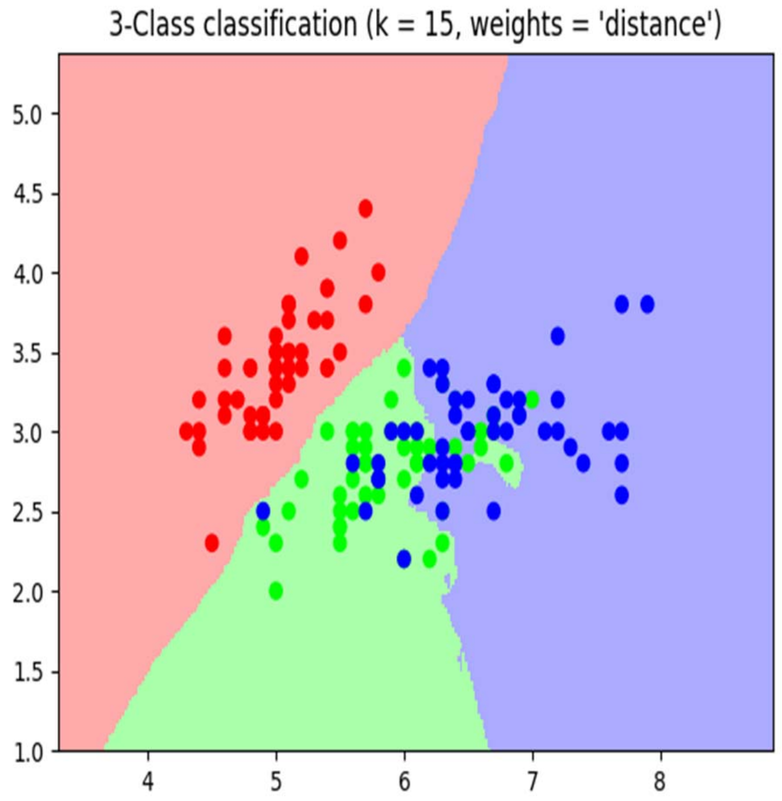


K Nearest Neighbours

- KNN is a classification and regression algorithm.
- Classifies datasets based on similarity with its closest K neighbours.
- Euclidean distance can be used as a similarity measure.
- Training consists of storing the feature vectors and class labels of the training samples.
- Testing consists of assigning a label to test object which is most frequent among the k training samples nearest to that point.

K Nearest Neighbours

3 class classification (K = 15)



k-Nearest Neighbors

Given a query item:
Find k closest matches
in a labeled dataset ↓

k-Nearest Neighbors

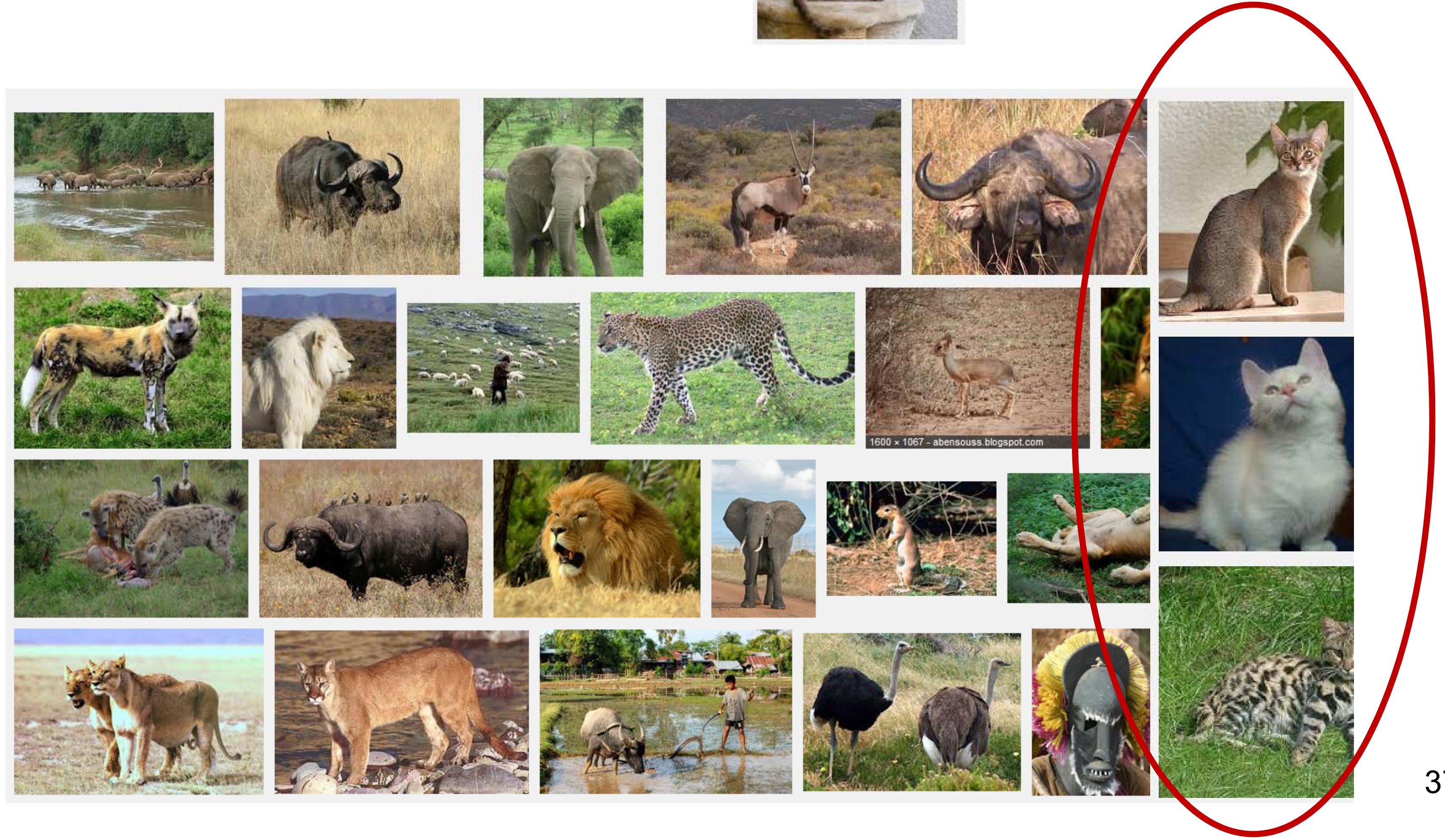
Given a query item:
Find k closest matches
in a labeled dataset ↓

Return the most
frequent label

k-Nearest Neighbors

k = 3 votes for "cat"

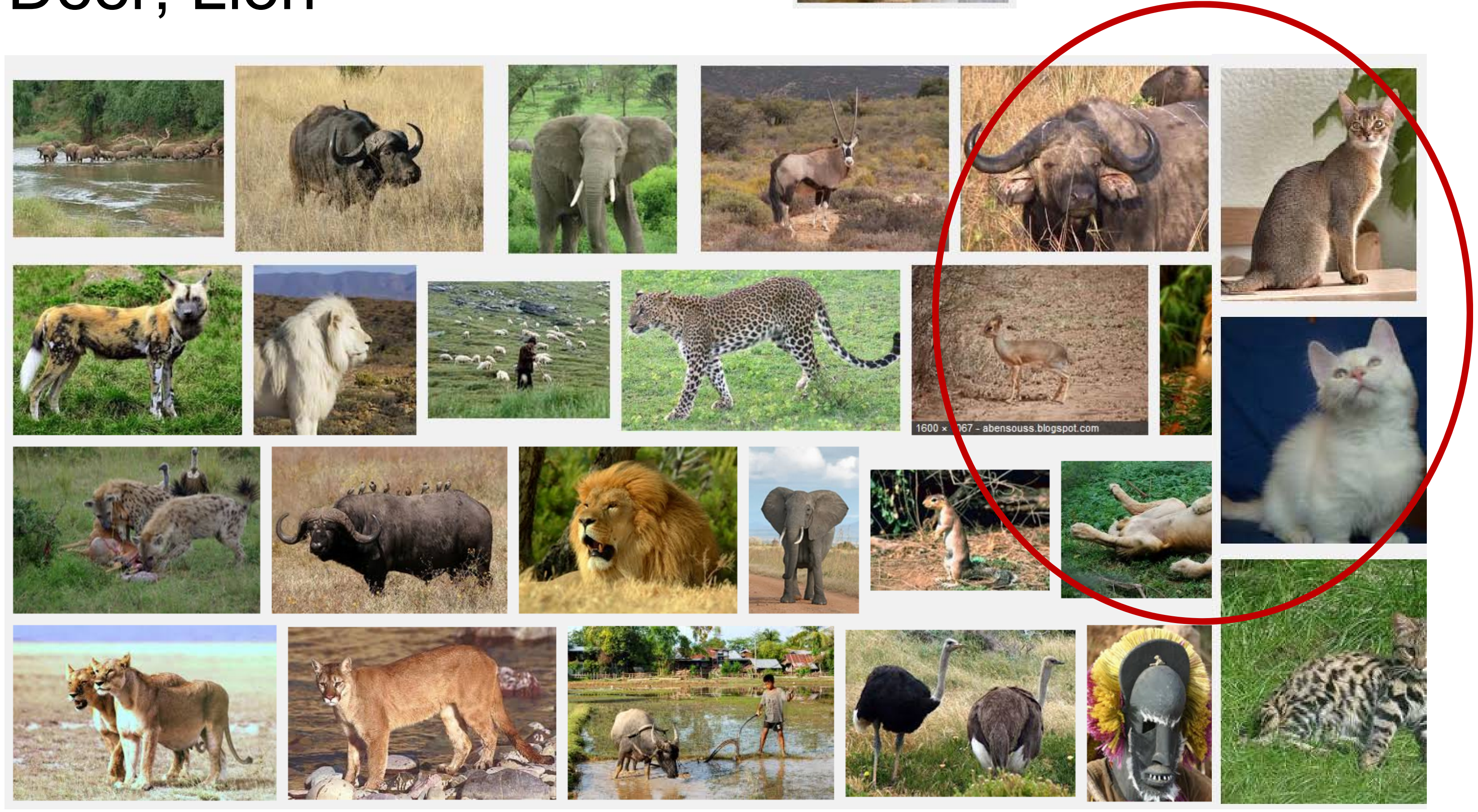
Cat wins…



k-Nearest Neighbors

2 votes for cat,
1 each for Buffalo,
Deer, Lion

Cat wins…



KNN: Disadvantages

- Large storage requirements
- Computationally intensive recall
- Neighborhood size k is manually assigned

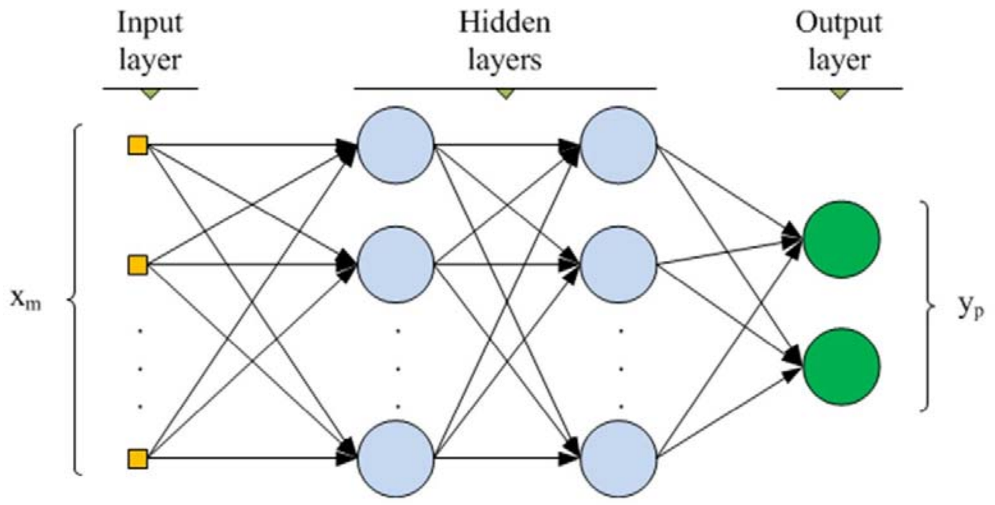
Feedforward Neural Network

Used in many applications such as spam detection, fingerprint recognition systems, finance prediction, activity recognition etc

A neural network is a multi-layered structure with the first layer as an input and the last layer as outputs.

Feedforward Neural Network

Each layer consists of multiple neurons.



The middle layers do not connect with the outer world, hence are called hidden layers.

Feedforward Neural Network

In a fully connected FNN, each neuron in one layer is connected with each neuron in the next layer.

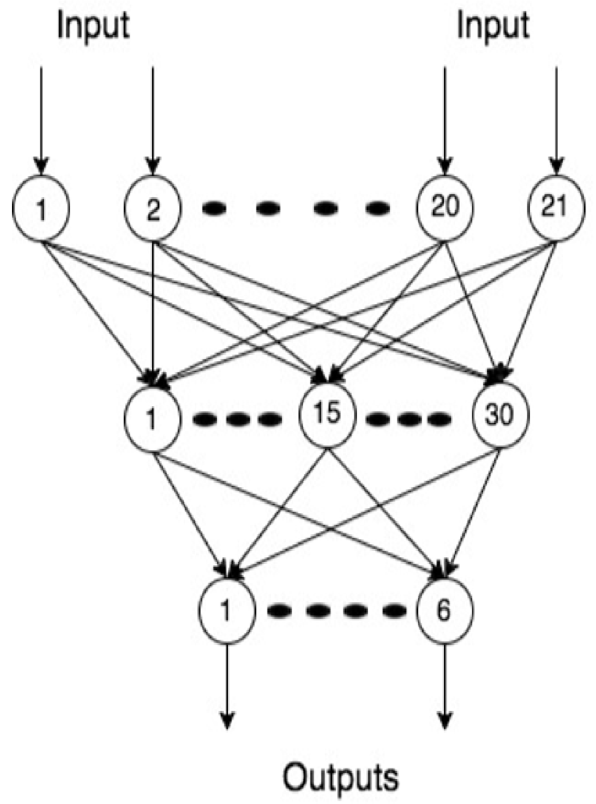
Each connection between neurons has a weight associated with it.

Learnt knowledge of the model is encoded into the weights of the neurons.

Feedforward Neural Network

Input of the neural network can be the activity signal input or the features extracted from the signal.

Output of the neural network consists of the labels/classes of the activity.



Feedforward Neural Network with 21 inputs, 3 hidden layers and 6 outputs

Comparing two signals…

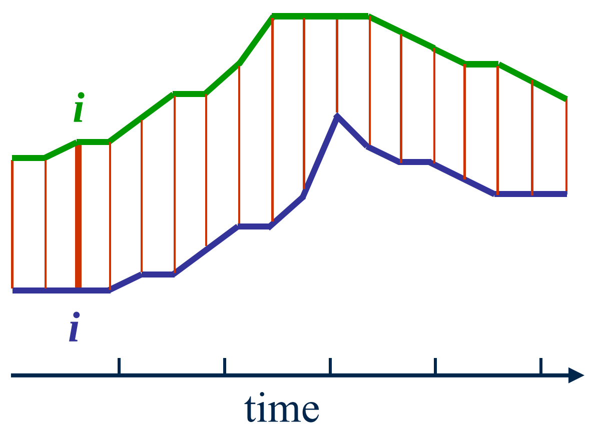
- Suppose we would like to compare two signals
- We need a certain flexibility by calculating the matching points of the two (maybe slightly different speeds)

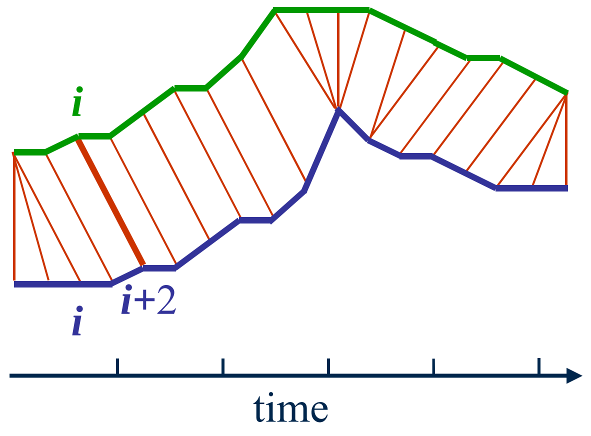
A common technique for this task is known as Dynamic Time Warping (DTW)

Dynamic Time Warping

- Used for the comparison of time series data.
- The incoming signal is compared with a reference template of the signal.
- One is interested in finding the optimal matching between two time series sequences (signals).

Dynamic Time Warping

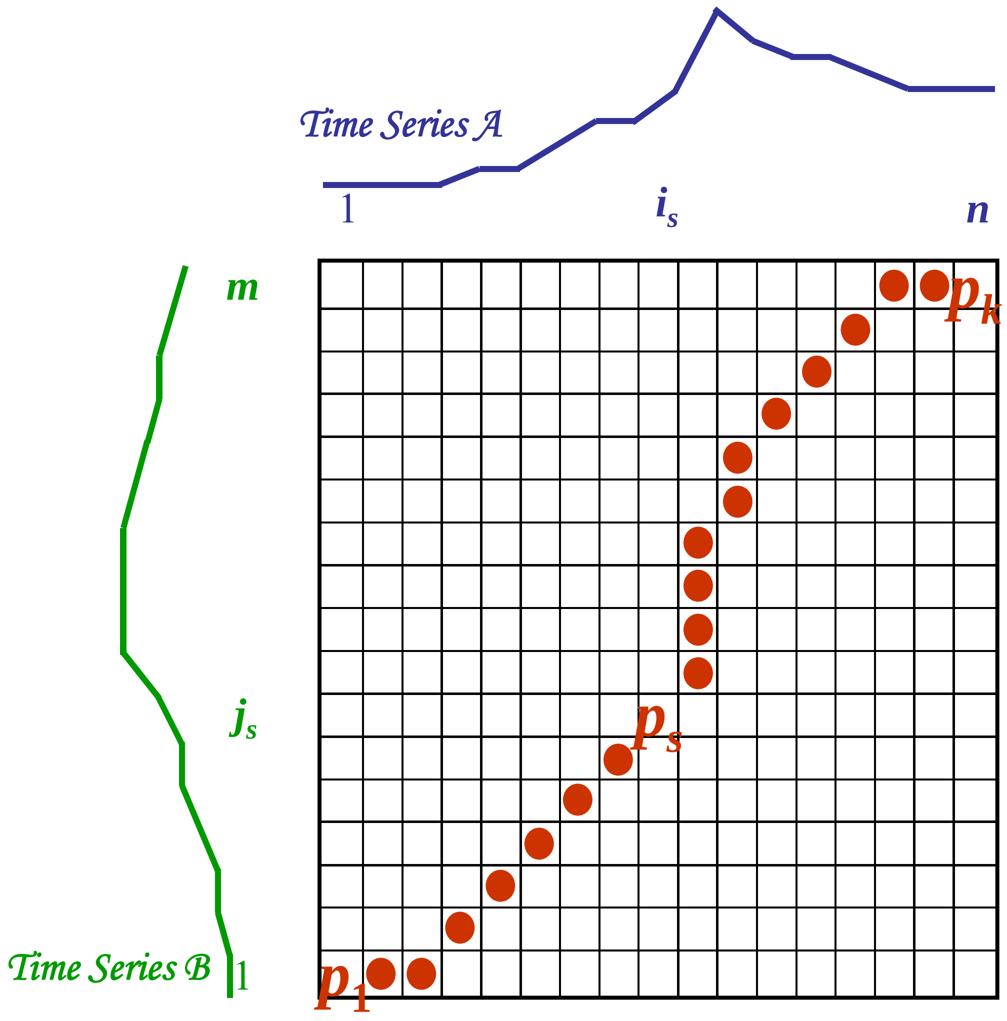




Any distance (Euclidean, Manhattan, …) which aligns the *i*-th point on one time series with the *i*-th point on the other will produce a poor similarity score.

A non-linear (elastic) alignment produces a more intuitive similarity measure, allowing similar shapes to match even if they are out of phase in the time axis.

Warping Function



To find the *best alignment* between ***A*** and ***B*** one needs to find the path through the grid

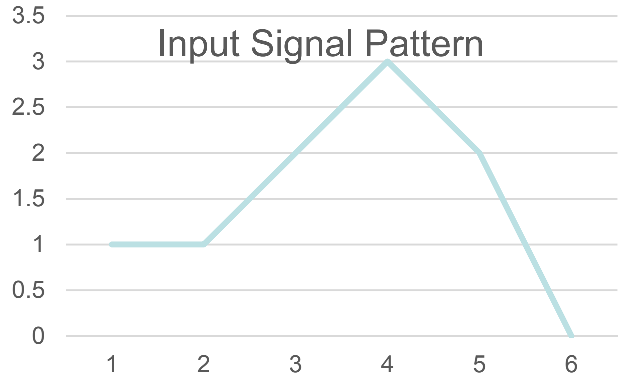
$P = p_1, \dots, p_s, \dots, p_k$

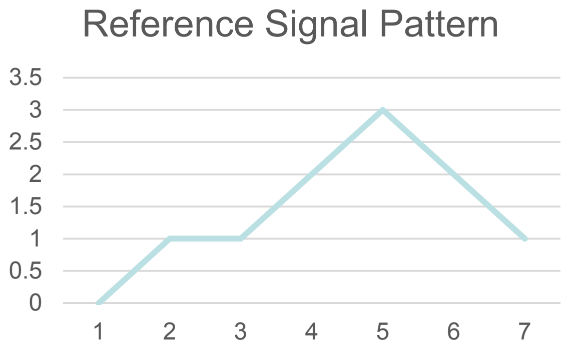
$p_s = (i_s, j_s)$

which *minimizes* the total distance between them.

P is called a <u>*warping function*</u>.

DTW Example





(Input Pattern)	x(t) = [1 1 2 3 2 0]
(Reference Pattern)	y(t) = [0 1 1 2 3 2 1]

The sample by sample difference for the last sample will be undefined since the number of samples in both signals are not the same.

DTW Example

- Both signals are similar in that they have only one peak.
- The reference sample is slightly longer than the input signal, and the peak occurs later.
- The two signals are not synced in time.
- Consider a matrix of distance between every sample of x(t) and y(t) as below:

Y(t)

1	0	0	1	2	1	-1
2	-1	-1	0	1	0	-2
3	-2	-2	-1	0	-1	-3
2	-1	-1	0	1	0	-2
1	0	0	1	2	1	-1
1	0	0	1	2	1	-1
0	1	1	2	3	2	0
	1	**1**	**2**	**3**	**2**	**0**

X(t)

DTW Example

The values in red indicate which values of x(t) are close to y(t).

The distance measure can also be written as (x(t) – y(t))^2 with the distance matrix as below.

Y(t)

1	0	0	1	4	1	1
2	1	1	0	1	0	4
3	4	4	1	0	1	9
2	1	1	0	1	0	4
1	0	0	1	4	1	1
1	0	0	1	4	1	1
0	1	1	4	9	4	0
	1	**1**	**2**	**3**	**2**	**0**

X(t)

DTW can be used as a classifier for activity recognition after finding the distance matrix between the new input and the already stored activity samples.

Classification libraries

There are many libraries/apis for ML, that support many different types of models

- Scikit-learn (python)
- Tensorflow (standalone, python, etc)
- Keras (python)
- Weka (java, standalone, python)

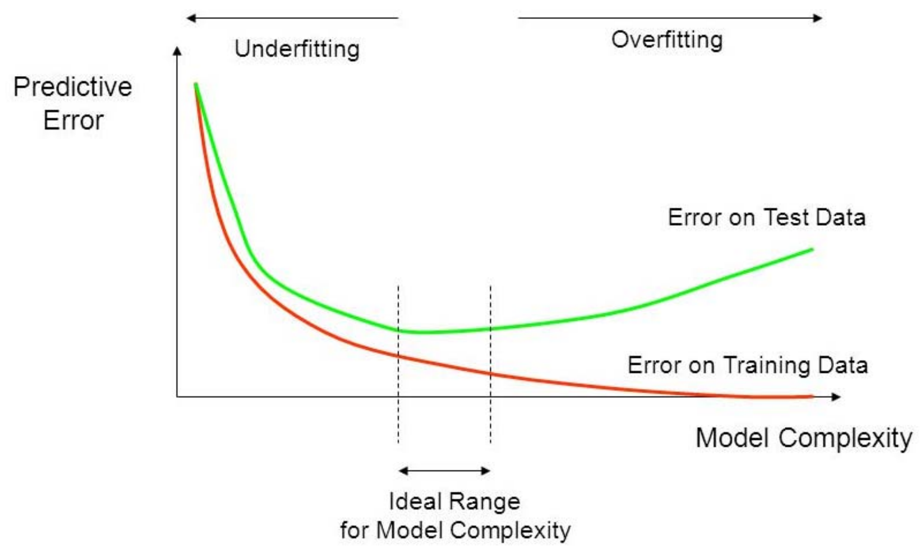
Model Assessment

- The *generalization* performance of a machine learning method relates to its prediction capability on independent test sets.
- Assessment of this performance is extremely important in practice, since it guides the choice of the machine learning method or model.
- Further, this gives us a measure of the quality of our chosen model.

Model Assessment (cont.)

- *Test Error*
  - The average error that results from using a machine learning method to predict the response on a <u>new</u> observation.
  - The prediction error over an independent test sample.
- *Training Error*
  - The average loss over the training sample: $\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$
- **Note:** The training error rate can dramatically *underestimate* the test error rate

Model Assessment (cont.)



- As the model becomes more and more complex, it uses the training data more and is able to adapt to more complicated underlying structures.

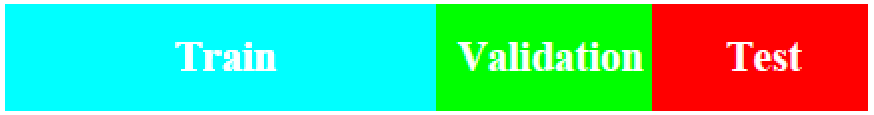
- Training error consistently decreases with model complexity but performs worse on test data (higher test error)

Model Assessment (cont.)

- If we are in a data-rich situation, the best approach for both *model selection* and *model assessment* is to randomly divide the dataset into three parts: training set, validation set, and test set.

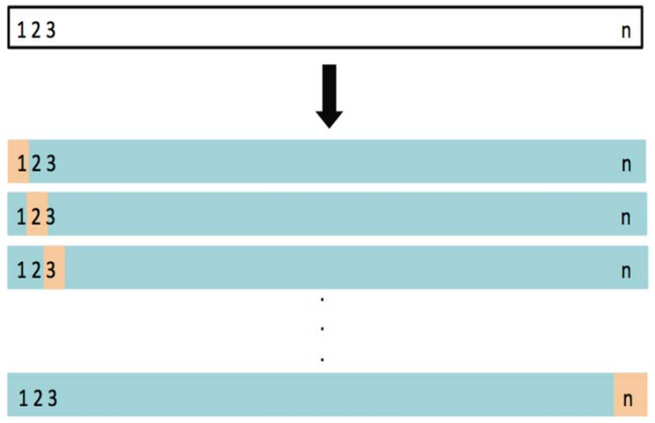
- The *training set* is used to fit the models. The *validation set* is used to estimate prediction error for model selection. The *test set* is used for assessment of the prediction error of the final chosen model.

- A typical split might by 50% for training, and 25% each for validation and testing.



Leave-One-Out Cross-Validation

- Instead of creating two subsets of comparable size, a single observation is used for the validation set and the remaining observations (n – 1) make up the training set.



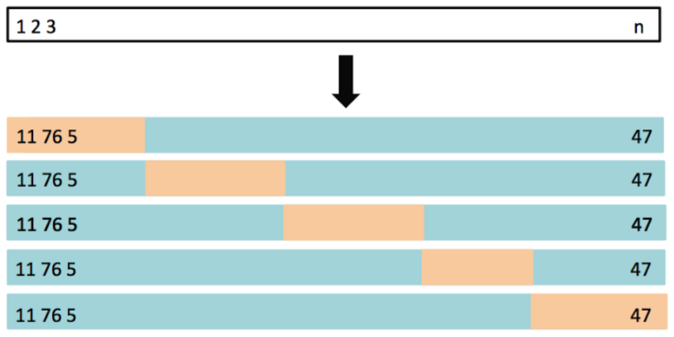
$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{MSE}_i$$

- **LOOCV Algorithm:**
  – Split the entire data set of size n into:
    - Blue = training data set
    - Beige = validation data set
  – Fit the model using the training data set
  – Evaluate the model using validation set and compute the corresponding MSE.
  – Repeat this process n times, producing n squared errors. The average of these n squared errors estimates the test MSE.

K-Fold Cross-Validation

- Probably the simplest and most widely used method for estimating prediction error.
- This method directly estimates the average prediction error when the machine learning method is applied to an independent test sample.
- Ideally, if we had enough data, we would set aside a validation set (as previously described) and use it to assess the performance of our prediction model.
- To finesse the problem, *K*-fold cross-validation uses part of the available data to fit the model, and a different part to test it.

K-Fold Cross-Validation (cont.)



- We use this method because LOOCV is computationally intensive.
- We randomly divide the data set of into *K* folds (typically *K* = 5 or 10).

- The first fold is treated as a validation set, and the method is fit on the remaining *K* – 1 folds. The MSE is computed on the observations in the *held-out* fold. The process is repeated *K* times, taking out a different part each time.
- By averaging the *K* estimates of the test error, we get an estimated validation (test) error rate for new observations.

10-fold Cross-validation vs leave-one-out cross-validation

- LOOCV is better when you have a small set of training data. In this case, you can't really make 10 folds to make predictions on using the rest of your data to train the model.

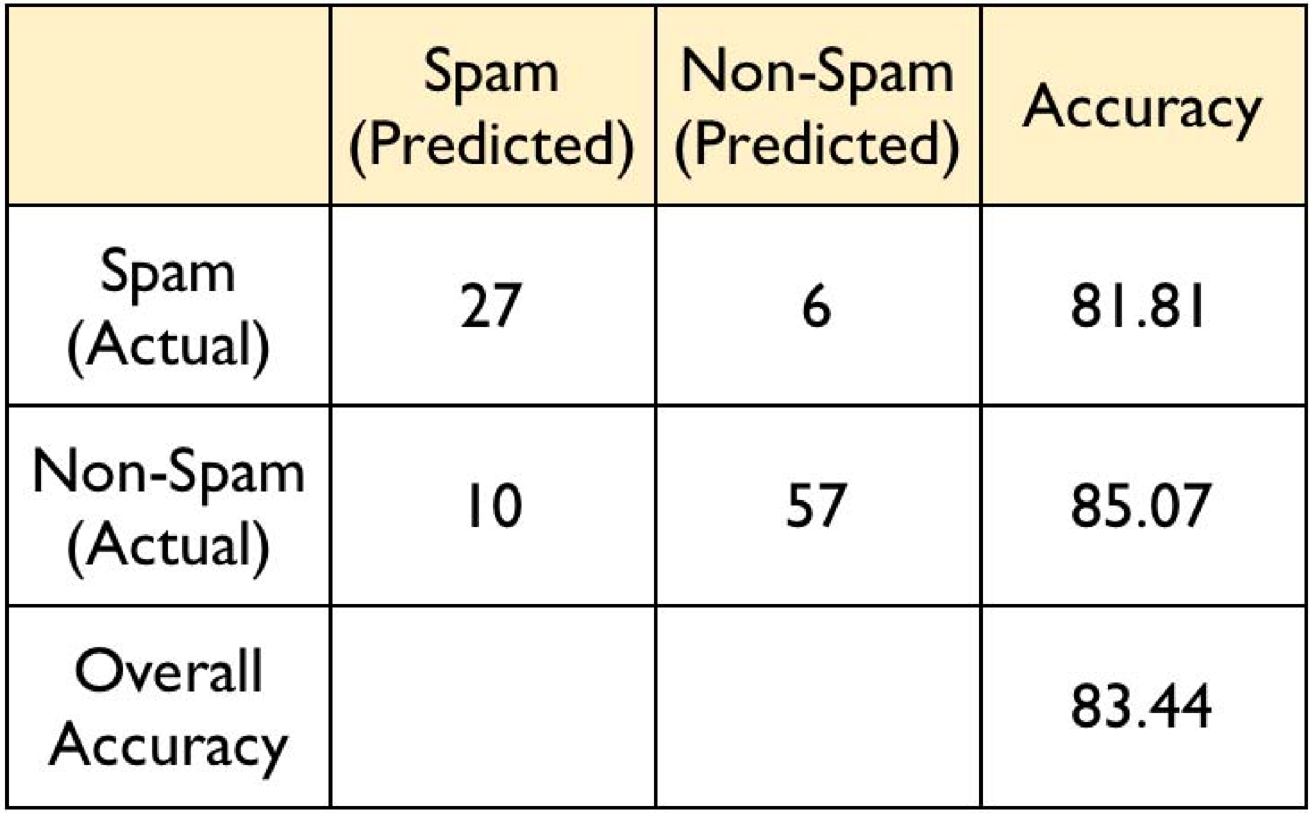
- If you have a large amount of training data on the other hand, 10-fold cross validation would be a better bet, because there will be too many iterations for leave one out cross-validation.

Evaluation Metrics

		Actual Label	
		Positive	Negative
Predicted Label	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of predictions that are correct
Precision	TP / (TP + FP)	The percentage of positive predictions that are correct
Sensitivity (Recall)	TP / (TP + FN)	The percentage of positive cases that were predicted as positive
Specificity	TN / (TN + FP)	The percentage of negative cases that were predicted as negative

Confusion Matrix



	Spam (Predicted)	Non-Spam (Predicted)	Accuracy
Spam (Actual)	27	6	81.81
Non-Spam (Actual)	10	57	85.07
Overall Accuracy			83.44

Machine Learning Examples with scikit-learn

XOR Example with Multi Layer Perceptron (MLP)

We would like to use Machine Learning to learn the XOR operator.

MLPClassifier implements a multi-layer perceptron (MLP) algorithm that trains using Backpropagation.

```
from sklearn.neural_network import MLPClassifier
```

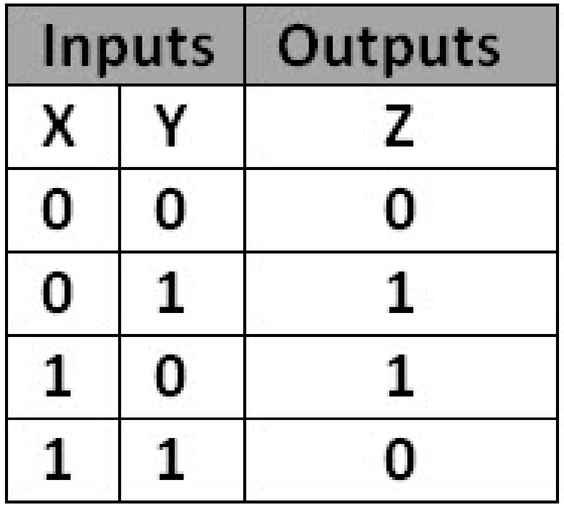
Trains on two arrays:

Input Array: holds the training samples represented as floating point feature vectors

```
XY = [[0., 0.], [0., 1.], [1.,0.], [1.,1.]]
```

Output Array: holds the target values (class labels) for the training samples

```
Z = [0., 1.,1.,0.]
```



Inputs		Outputs
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Example MLP.py

Training and Prediction

Use MPL Classifier and fit the model according to the training data.

```
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(15,), random_state=1)
clf.fit(X, y)
```

After fitting (training), the model can predict labels for new samples

```
print(clf.predict([[0., 0.]]))
print(clf.predict([[0., 1.]]))
print(clf.predict([[1., 0.]]))
print(clf.predict([[1., 1.]]))
```

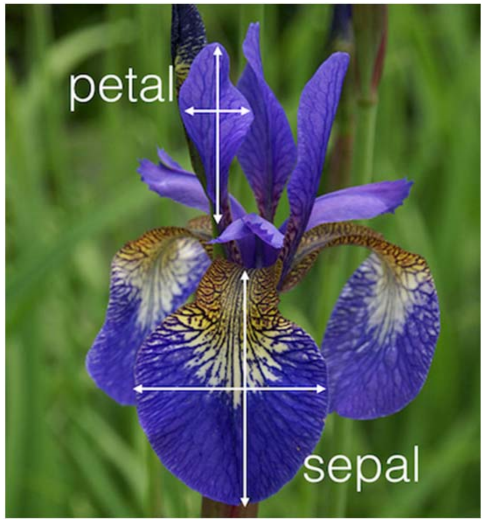
The output looks like:

```
[ 0.]
[ 1.]
[ 1.]
[ 0.]
```

Another Classification Problem

Now we are going to use an online available dataset (IRIS dataset) to classify three flowers:
Iris Setosa
Iris Versicolour
Iris Virginica

Can we classify the above species based on their features?



Import the required libraries

```
from sklearn import neighbors, datasets
```

Load the IRIS data:

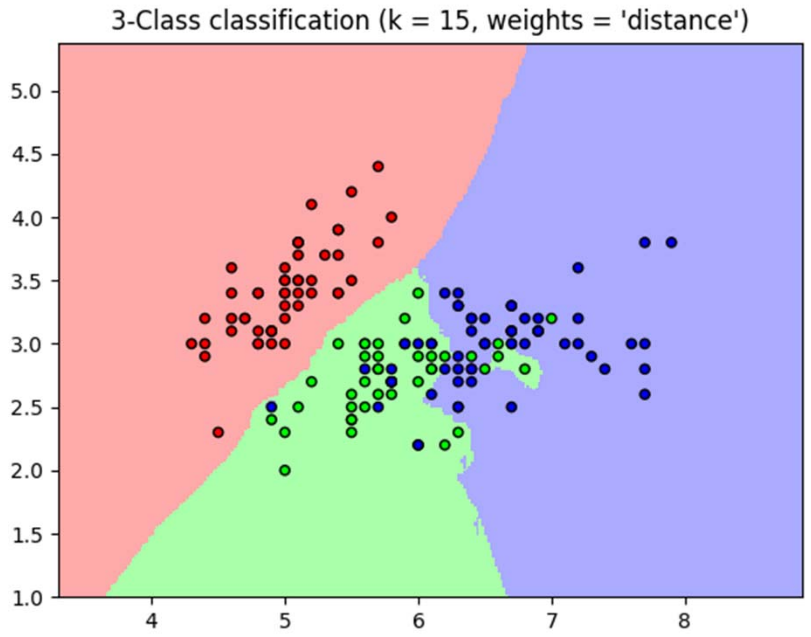
```
iris = datasets.load_iris ()
```

Get the first two features of the data and the target outputs:

```
X = iris.data[:, :2]
y = iris.target
```

Use the KNN classifier to train the data

```
clf = neighbors.KNeighborsClassifier(n_neighbors, weights=weights)
clf.fit(X, y)
```



http://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html#sphx-glr-auto-examples-neighbors-plot-classification-py

Links

Look into classification examples from scikit-learn:

http://scikit-learn.org/stable/supervised_learning.html#supervised-learning

If interested, should also have a look into tensorflow library, which provides more intuition into deep learning models.

https://www.tensorflow.org/api_docs/python/tf/contrib/learn/DNNClassifier

Conclusion

Sensors such as accelerometers serves as a useful tool in identifying activities based on user’s movements.

The classification model determines the accuracy of the system and needs to be analysed carefully in any classification problem.

For an accurate and reliable recognition system, a number of features, segmentation techniques and models should be analysed.