COMP9517: Computer Vision

Pattern Recognition

Part 1

Introduction

- **Pattern recognition:** is the scientific discipline whose goal is to automatically recognise patterns and regularities in the data (e.g. images).

- **Examples:**
  - object recognition / object classification
  - Text classification (e.g. spam/non-spam emails)
  - Speech recognition
  - Event detection
  - recommender systems

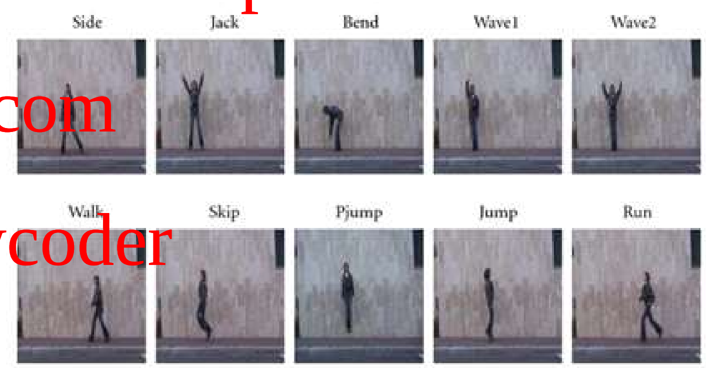
Pattern recognition categories

Based on learning paradigm:

- **Supervised learning:** learning patterns in a set of data together with available labels (ground truth)
- **Unsupervised learning:** finding patterns in a set of data without any labels available
- **Semi-supervised learning:** uses a combination of labelled and unlabelled data to learn patterns
- **Weakly supervised learning**: uses noisy, limited or imprecise labels for the data to learn patterns in a supervised setting

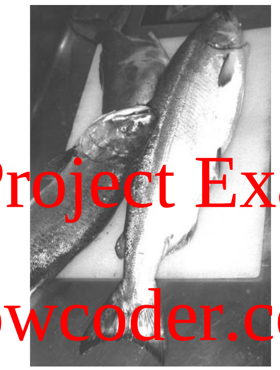
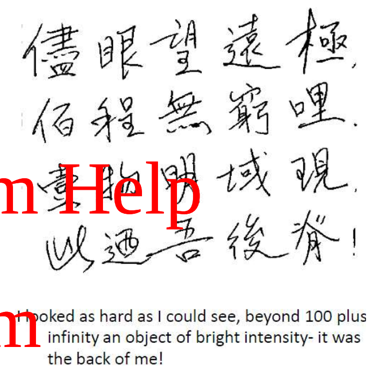
Applications in Computer Vision

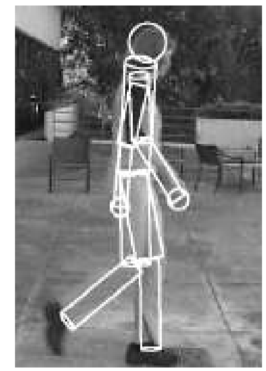
- making decisions about image content
- classifying objects in an image
- recognising activities

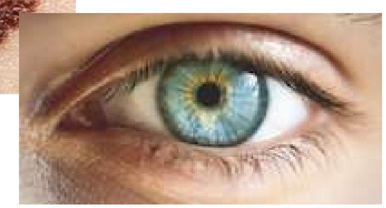


Applications in Computer Vision

- Character recognition
- Human activity recognition
- Face detection/recognition
- Image-based medical diagnosis
- Image Segmentation
- Biometric authentication

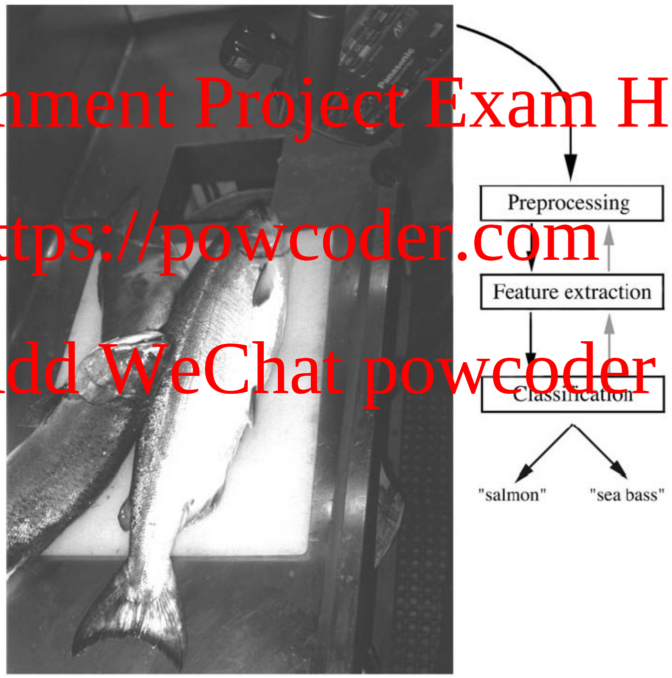





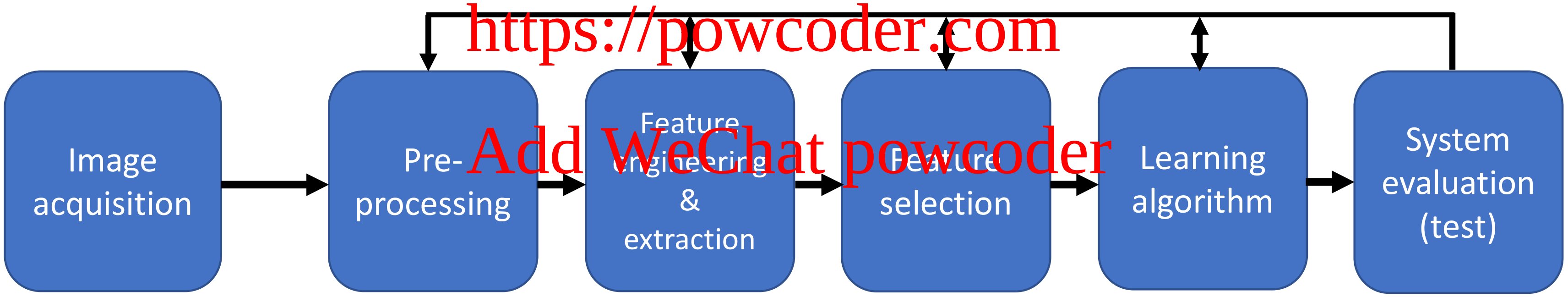
Pattern Recognition Systems

- Prototype System



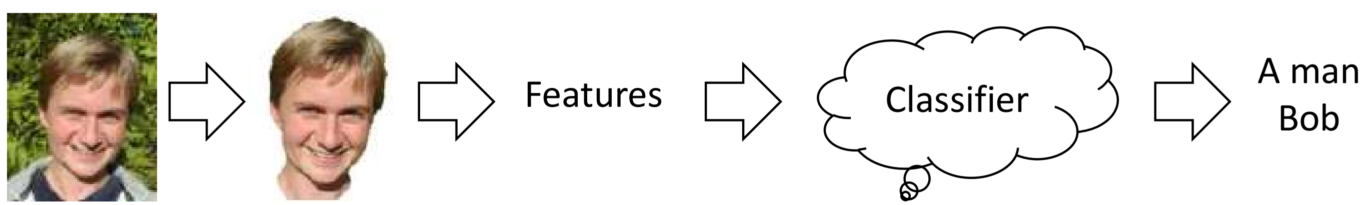
Pattern Recognition Systems

- Basic stages involved in the design of a classification system in computer vision:



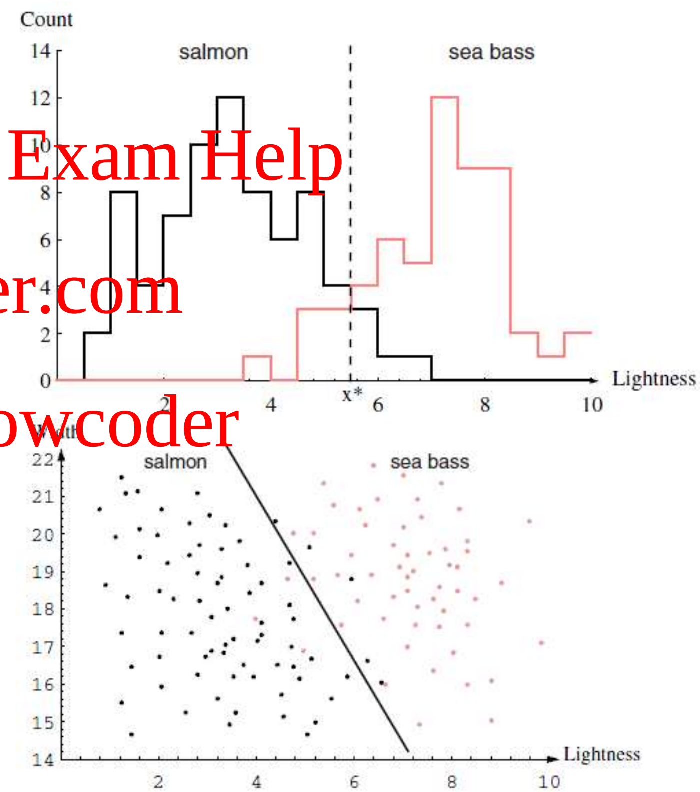
Pattern Recognition Concepts

- ***Object*** is a physical unit (an identifiable portion)
- ***Regions*** (ideally) correspond to objects, obtained after segmentation of an image
- ***Classes:*** the set of objects can be divided into disjoint subsets that may have some common features- such sets are called classes
- Class to which an object belongs is denoted by a **class label**
- ***Classification*** is a process that assigns labels to objects, based on object properties
- ***Classifier:*** the corresponding algorithm/method is called the classifier
- ***Pattern:*** the classifier bases its decision on object features, called the pattern



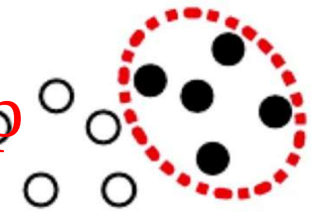
More Concepts

- ***Features:*** description/properties of objects
- ***Model:*** description of classes
- ***Pre-processing*** for noise removal, segmentation
- ***Feature Extraction*** reduces the data by measuring certain “ features” or properties
- ***Training samples***- objects with known ground truth / labels used to build model
- ***Cost***- consequence of making incorrect decision
- ***Decision boundary*** between regions in feature space

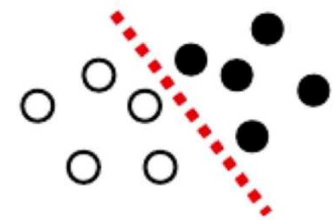


Pattern Recognition Overview-I

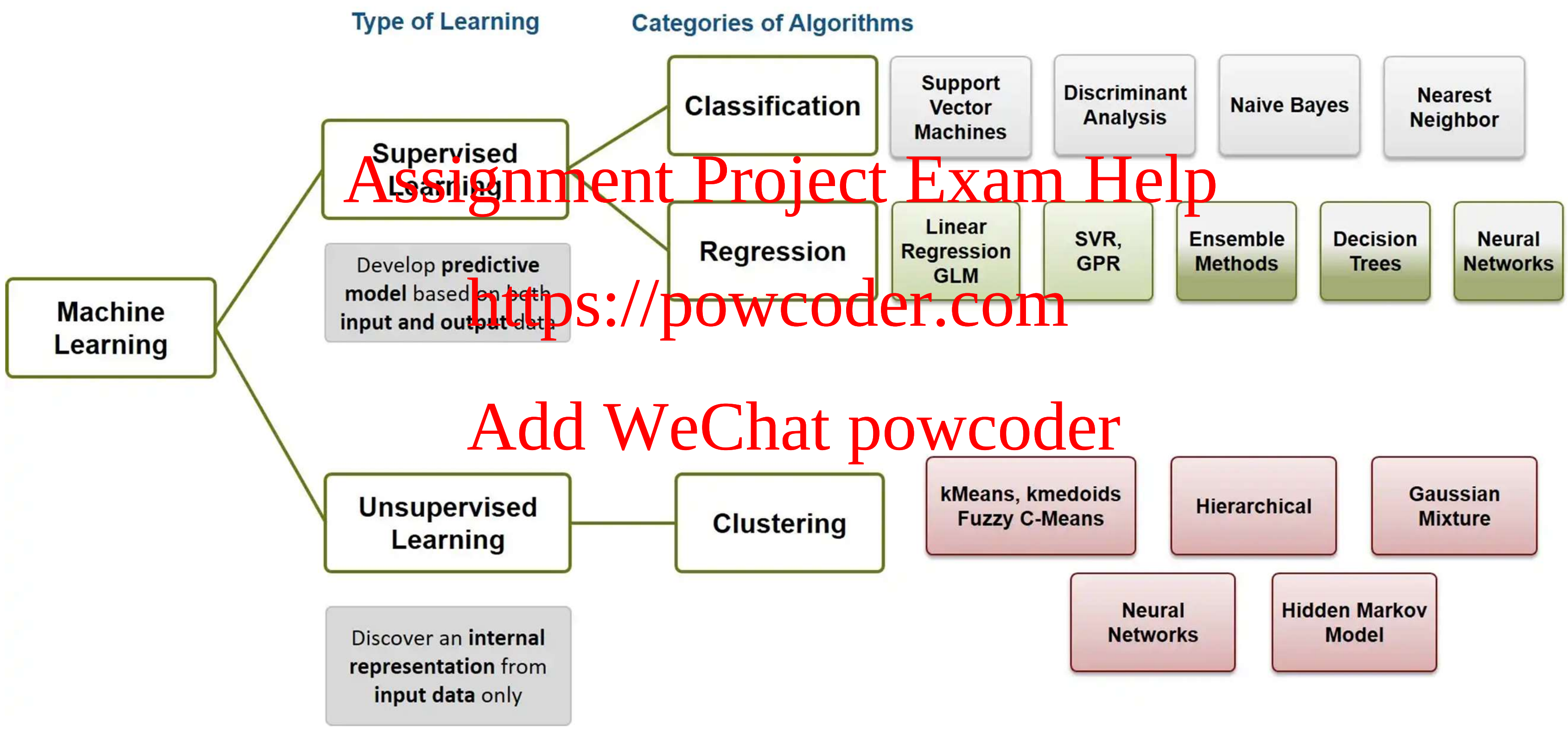
- Model Types
  - **Generative Model:** builds some models for each class
    - probabilistic “model” of each class
      - models the joint probability *p(x,y) (it learns p(x|y) and p(y) and p(x,y) = p(x|y)p(y))*
      - “mechanism” by which the data is generated
    - decision Boundary:
      - one model more likely than others (*p(y|x)*)
    - applicable to unsupervised learning tasks (unlabeled data)



  - **Discriminative Model**
    - focus on the decision boundary
      - models the posterior probability *p(y|x)*
    - suited for supervised learning tasks (labeled data)



Pattern Recognition Overview



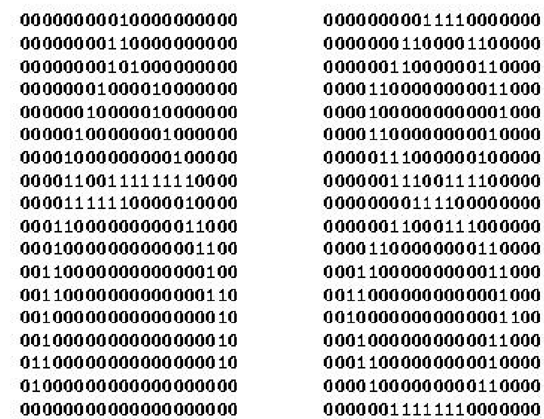
Features and Descriptions

- ***Features***
  - descriptions representing scalar properties of objects are called ***features***
  - used to represent knowledge as part of more complex representation structure
- ***Feature vector***
  - combines many features, e.g. size feature represents area property, compactness feature represents circularity

- Good representation is important to solve a problem

Feature Vector Representation

- $X = [x_1, x_2, \dots, x_d]$, each x_j is a descriptor
  - x_j may be an object measurement
  - x_j may be count of object parts
  - x_j may be colour of the object
  - Features go by other names like predictors/descriptors/covariates/independent variables
- Example:
  - For fish type example: *[length, colour, lightness,…]*
  - *Letter/digit recognition: [holes, moments, SIFT,…]*



Feature Extraction

- Goal of feature extraction is to characterise object by measurements that are
  - similar for objects in the same class/category and
  - different for objects in different classes

- Must find ***distinguishing features*** that are invariant to input transformations
- Design of features often based on prior experience or intuition

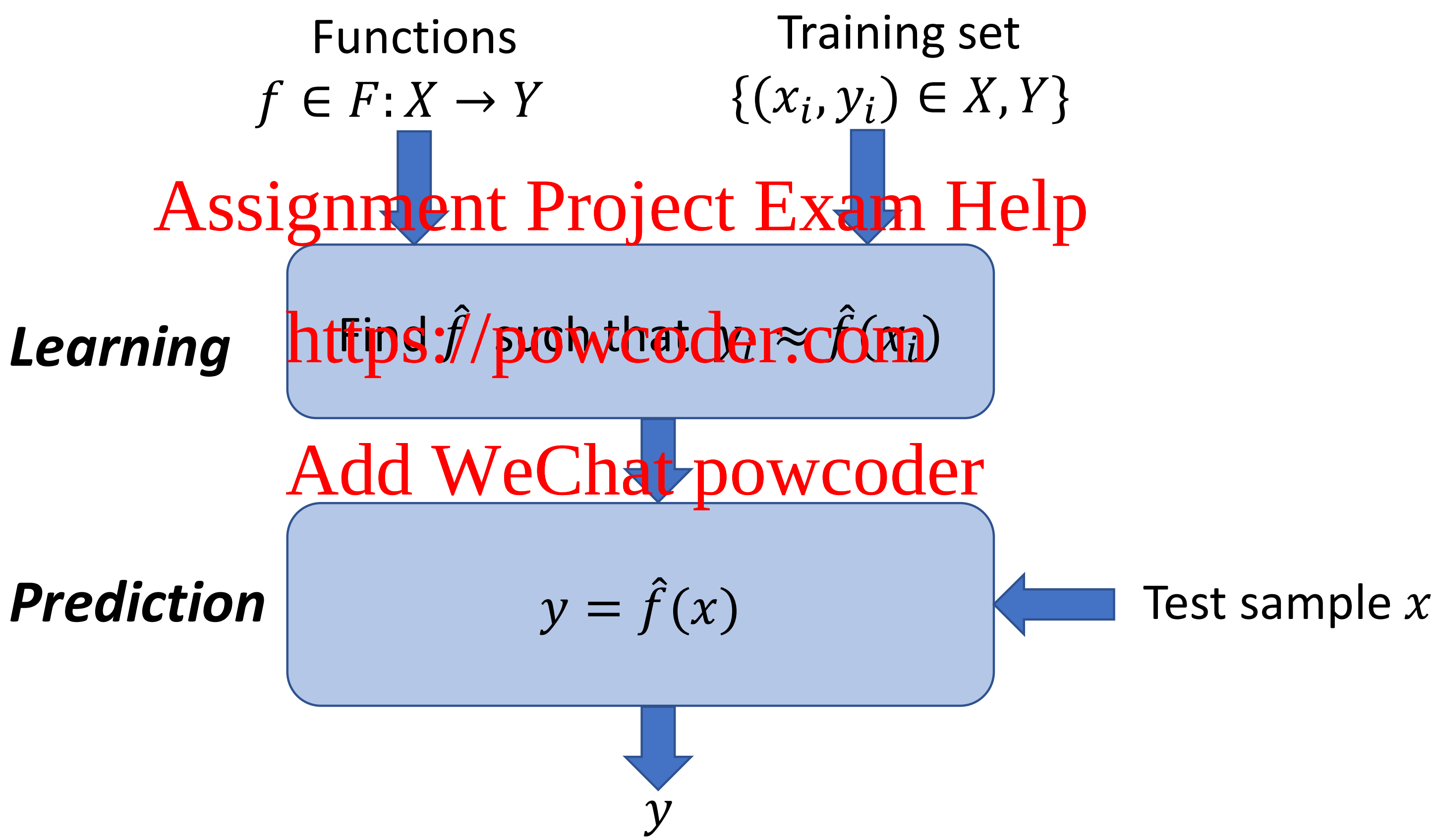
Feature Extraction

- Selecting features that are
  - translation, rotation and scale invariant in images, eg. shape, colour, texture
  - handling ***occlusion***, projective distortion for 3-D objects in images when distance between object and camera changes
  - handling ***non-rigid deformations*** common in 3-D vision, eg fingers around a cup
  - handling variations in illumination, shadow effects

- Feature selection is problem- and domain-dependent, requires domain knowledge

- But classification techniques can help to *make feature values less noise sensitive*, and to *select valuable features* out of a larger set

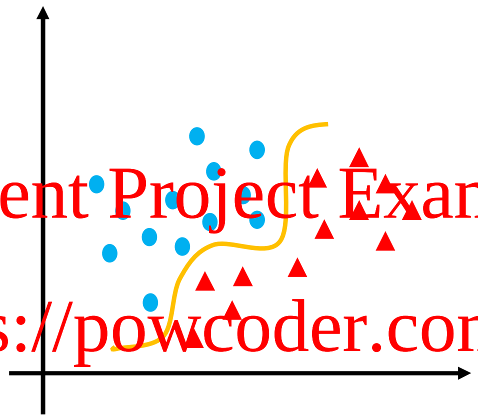
Supervised Learning Overview



Classification

- Classifier performs object recognition by assigning a class label to an object, using the object description in the form of features
- Perfect classification is often impossible, instead determine probability for each possible category
- Variability in feature values for objects in the same class versus objects in different classes causes the difficulty of the classification problem
  - Variability in feature values may arise due to complexity, but also due to ***noise***
  - Noisy features and missing features are major issues- not always possible to determine values of all features for an input

Classification



- If we have training set of N observations:

$$\{(x_i, y_i)\}, x_i \in \mathbb{R}^d, y_i \in \{-1,1\}$$

- Classification problem is to estimate $f(x)$ from this data such that:

$$f(x_i) = y_i$$

Nearest Class Mean Classifier

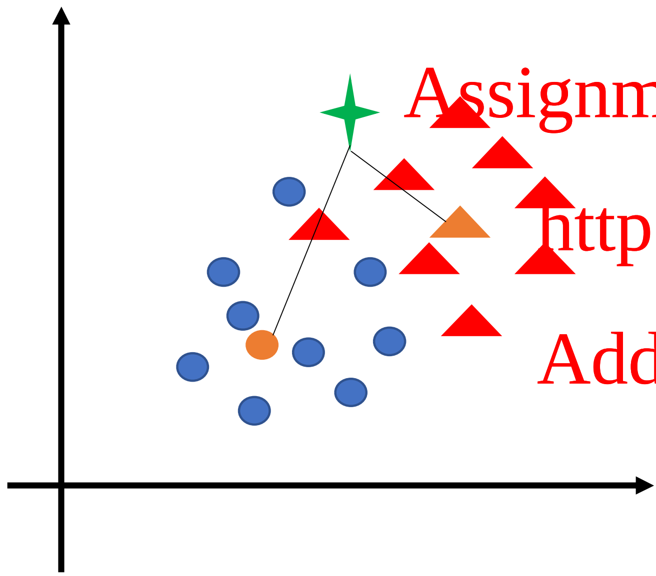
- This is a classifier based on minimum distance principle, where the class exemplars are just the centroids (or means)- model described by the mean

- Training: for training sample pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ where x_i is the feature vector for sample i and y_i is the class label, class centroids are:

$$\mu_k = \frac{1}{|C_k|} \sum_{j \in C_k} x_j$$

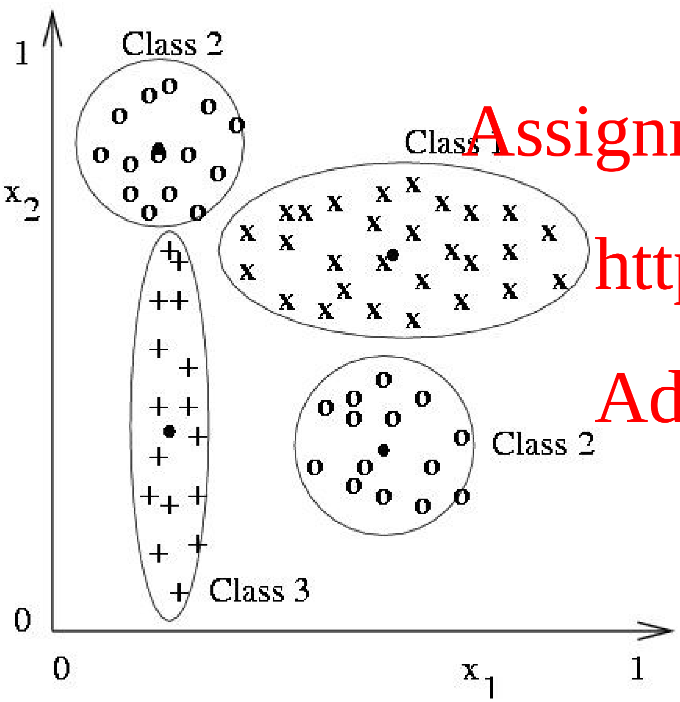
- Test
  - a new unknown object with feature vector x is classified as class k if it is much closer to the mean vector of class k than to any other class mean vector

Nearest Class Mean Classifier



- Compute the Euclidean distance between feature vector X and the mean of each class
- Choose closest class, if close enough (reject otherwise)

Nearest Class Mean Classifier



- Class 2 has two modes; where is its mean?
- But if modes are detected, two subclass mean vectors can be used

Nearest Class Mean Classifier

- Pros:
  - Simple
  - Fast
  - works well when classes are compact and far from each other.
- Cons:
  - For complex classes (eg. Multimodal, non-spherical) may give very poor results
  - Cannot handle outliers and noisy data well
  - Cannot handle missing data

K-nearest Neighbours

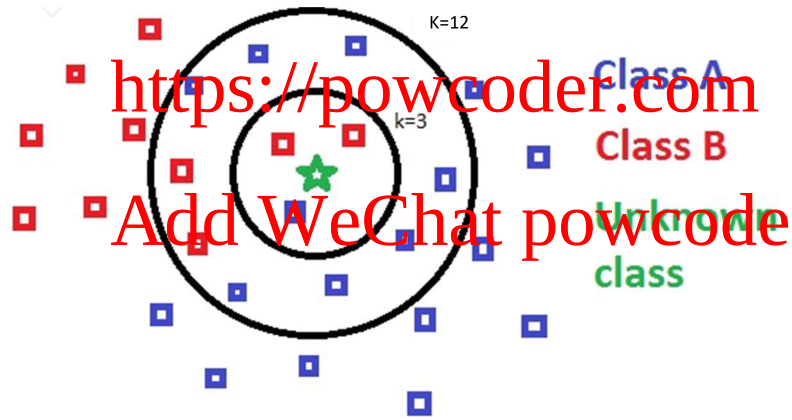
- K-NN is a classifier that decides class label for a sample, based on the K nearest samples
- The sample will be assigned to the class which has majority of members in the neighborhood

- The neighbours are selected from a set of samples for which the class is known

- For every new test sample, the distances between the test sample and all training samples are computed, and the K nearest training samples are used to decide the class label of test sample

K-nearest Neighbours

- Commonly used distance for continuous variables is Euclidean distance
- for discrete variables, Hamming distance.



https://towardsdatascience.com/knn-using-scikit-learn-c6bed765be75

K-nearest Neighbours

- Pros:
  - Very simple and intuitive
  - Easy to implement
  - No a priori assumptions
  - No training step
  - Decision surfaces are non-linear
- Cons:
  - Slow algorithm for big datasets
  - Does not perform well when the number of variables grows (curse of dimensionality)
  - Needs homogeneous feature types and scales
  - Finding the optimal number of neighbours can be challenging

K-nearest Neighbours: An Application

- Automated MS-lesion segmentation by KNN
- Manually labeled image used as training set
- 4 features used: Intensity and voxel locations (x,y,z coordinates)

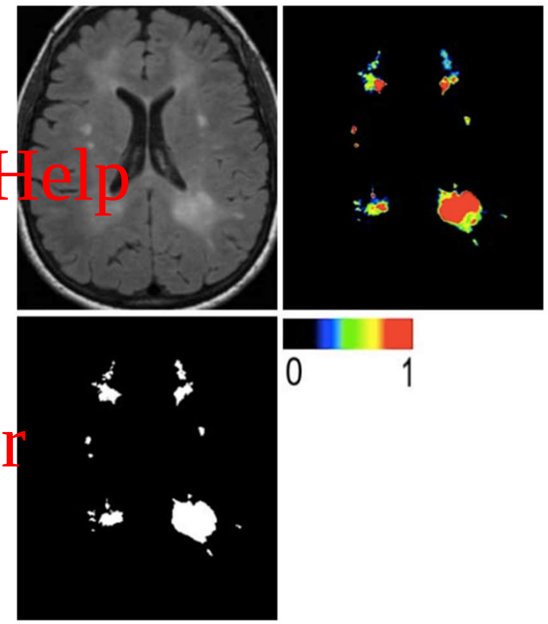
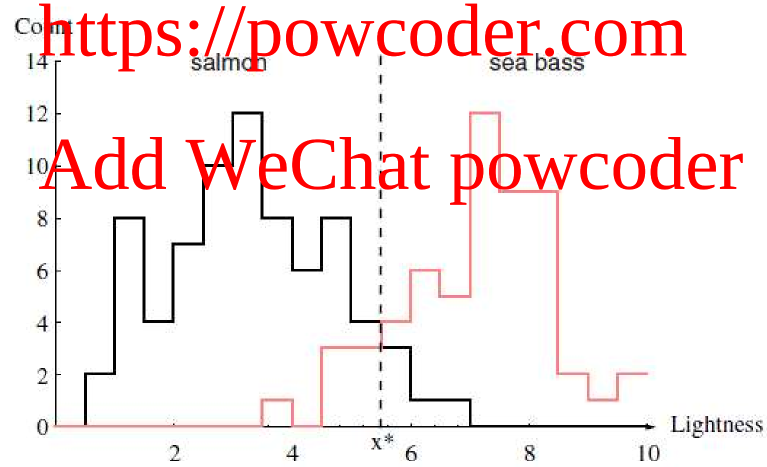


Figure 1 MS-lesion segmentation results. Top left: FLAIR image; top right: probabilistic segmentation, showing probability of lesion per voxel (see color bar); down left: binary segmentation, derived from probabilistic segmentation with threshold 0.4.

- Ref: Anbeek et. Al, “Automated MS-lesion segmentation by K-nearest neighbor classification”, MIDAS journal, 2008

Bayesian Decision Theory

- A classifier's decision may or may not be correct, so setting should be probabilistic (soft decision)
- Probability distributions may be used to make classification decisions with least expected error rate



Bayesian Decision Theory

- ***Bayesian classifier*** classifies an object into the class to which it is most likely to belong, based on observed features
- Assume:
  - *a priori* probability $P(c_i)$ for each class c_i
  - unconditional distribution $P(x)$
  - class conditional distribution $P(x|c_i)$

We shall compute the posterior probability distribution $P(c_i|x)$ as follows:

- If all the classes are disjoint, by Bayes Rule, the *a posteriori* probabilities are given by:

$$P(c_i|x) = \frac{P(x|c_i)P(c_i)}{\sum_j P(x|c_i)P(c_i)}$$

Bayesian Decision Theory

- If we have an observation x for which $P(c_1|x)$ is greater than $P(c_2|x)$, we would naturally prefer to decide that the true state of nature is c_1
- So the decision is: $c^* = \arg max_i(P(c_i|x))$

$P(c_i|x) = \frac{P(x|c_i)P(c_i)}{\sum_j P(x|c_i)P(c_i)}$ and the denominator

$\sum_j P(x|c_i)P(c_i) = P(x)$ is equal for all $P(c_i|x)$, so we can write:

$$c^* = \arg max_i(P(x|c_i)P(c_i))$$

Bayesian Decision Theory

- Whenever we observe a particular $\boldsymbol{x}$, the probability of error is:

$$P(error|x) = \begin{cases} P(c_1|x), if\ we\ decide\ c_2 \\ P(c_2|x), if\ we\ decide\ c_1 \end{cases}$$

- Clearly, for a given x we can minimise the probability of error by deciding c_1 if $P(c_1|x) > P(c_2|x)$

- This is the ***Bayes decision rule***

Bayesian Decision Theory: Example

- We want to classify fish type: Salmon , Sea bass, Other
- From past experience we already know the probability of each class:

P(c_i)	Salmon	Sea bass	Others
Prior	0.3	0.6	0.1

- If we decide only based on prior, we shall always have to choose “sea bass”. This is called “*decision rule based on prior*”
  - This can behave very poorly
  - It never predicts other classes

Bayesian Decision Theory: Example

- Let us use some features to add more information:
    - E.g.: length

- From experience we know the *"class conditionals"* for feature *"length"*

$P(x\|c_i)$	Salmon	Sea bass	other
length > 100 cm	0.5	0.3	0.2
50 cm < length < 100 cm	0.4	0.5	0.2
length < 50 cm	0.1	0.2	0.6

- Now we can estimate the posterior probability for each class

Bayesian Decision Theory: Example

- If we have a fish with length 70cm, what would be our prediction?

$P(c_i = salmon|x = 70cm) \propto P(70cm|salmon) * P(salmon)$ =0.4*0.3=0.12

$P(c_i = sea\ bass|x = 70cm) \propto P(70cm|sea\ bass) * P(sea\ bass)$ =0.5*0.6=0.30

$P(c_i = other|x = 70cm) \propto P(70cm|other) * P(other)$ =0.2*0.1=0.02

- So based on these probabilities, our model predicts the type as “sea bass”
- Question: If the price of salmon is twice that of sea bass, and sea bass is also more expensive than the other types of fish, is the cost of a wrong decision the same for any misclassification?

Bayesian Decision Theory Risk

- If the prices of all types of fish are the same, then we can make the decision by maximizing the posterior probability

- But if the prices are not the same, we have to minimize the loss:
  - Loss is the cost of an action α_i based on our decision: $\lambda(\alpha_i|c_i)$

  - The expected loss associated to action α_i is:

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|c_j)P(c_j|x)$$

  - $R(\alpha_i|x)$ is also called conditional risk
  - An optimal Bayes decision strategy is to minimize the conditional risk

Bayesian Decision Theory Risk: Example

- Continuing with the same example, and assuming we have the loss function $\lambda(\alpha_i|c_i)$:

$\lambda(\alpha_i\|c_i)$	Salmon	Sea bass	Other
If predicted salmon	0	2	3
If predicted sea bass	10	0	4
If predicted other	20	7	0

$$R(sell\ as\ salmon|50 < x < 100) = \lambda(sell\ as\ salmon|salmon)P(salmon|50 < x < 100) + \lambda(sell\ as\ salmon|sea\ bass)\ P(sea\ bass|50 < x < 100) + \lambda(sell\ as\ salmon|other)P(other|50 < x < 100) \propto 0\times P(50 < x < 100|salmon)P(salmon) + 2\times P(50 < x < 100|sea\ bass)P(sea\ bass) + 3 \times P(50 < x < 100|other)P(other) = 0 + 2 \times 0.5 \times 0.6 + 3 \times 0.2 \times 0.1 = 0.66$$

$$R(sell\ as\ sea\ bass|50 < x < 100) \propto 1.28$$

$$R(sell\ as\ others|50 < x < 100) \propto 4.5$$

Bayesian Decision Theory Risk: Example

$$R(sell\ as\ salmon|50 < x < 100) \propto 0.66$$
$$R(sell\ as\ sea\ bass|50 < x < 100) \propto 1.28$$
$$R(sell\ as\ others|50 < x < 100) \propto 4.5$$

- So, if the length of your fish is in the range of $[50,100]$, the loss would be minimized if you predict it as "*salmon*"

Bayesian Decision Theory

Parametric Models for Distributions

- To compute $P(x|c_i)$ and $P(c_i)$, we can use an empirical method based on given samples

- Or if we know that the distribution of x follows a parametric model, then we may estimate the parameters using the samples

- **An Example:**

  - Assume that the patterns in the i^{th} class can be described by a normal distribution, whose covariance matrix Σ_i is known but the mean μ_i is unknown
  - Then, an estimate of the mean may be the average of the labelled samples available in the training set:

$$\tilde{\mu} = \bar{x}$$

Bayesian Decision Theory

- Pros:
  - Considers uncertainties
  - Permits combining new information with the current knowledge
  - Simple & intuitive

- Cons
  - Computationally expensive
  - Choice of priors can be subjective

Decision Trees Introduction

- Most practical pattern recognition methods address problems where feature vectors are real-valued and there exists some notion of a metric

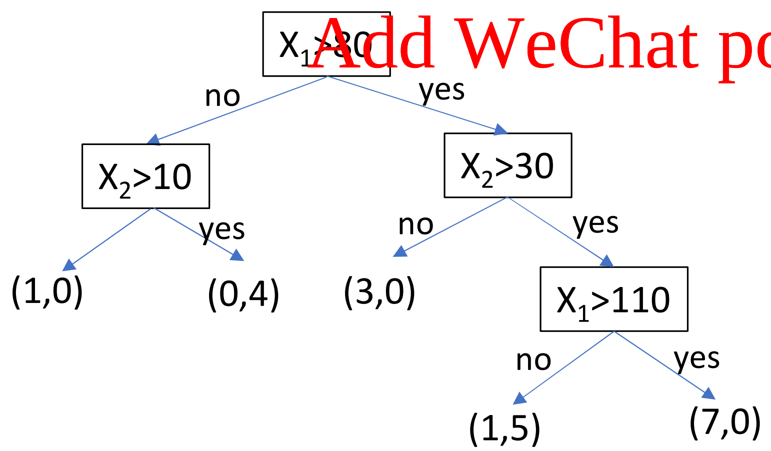
- There are classification problems involving ***nominal data*** where instance descriptions are discrete and without any natural notion of similarity or even ordering

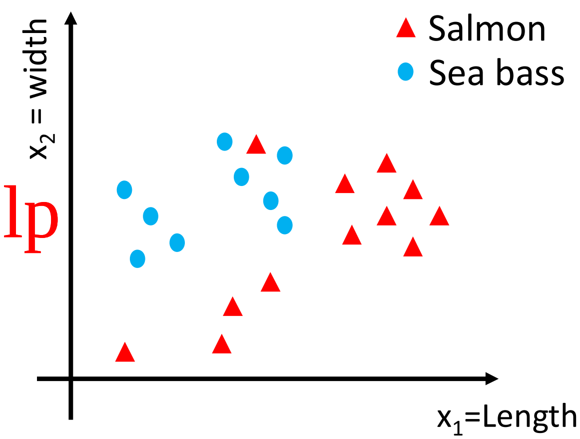
  - For example: {high, medium, low}, {red, green, blue}

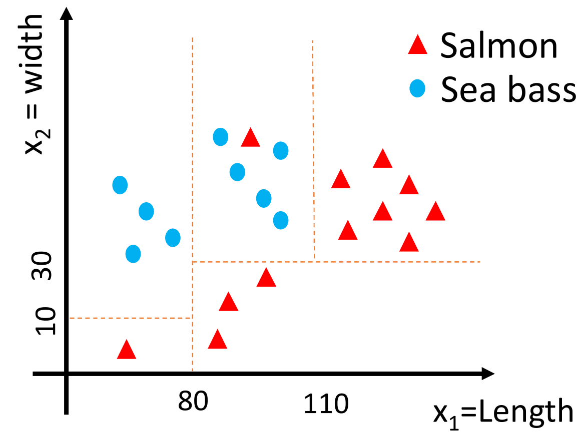
  - ***Nominal data*** are also called as ***categorical data***
- How can we use such nominal data for classification?
  - Use rule-based methods

Decision Trees: Example

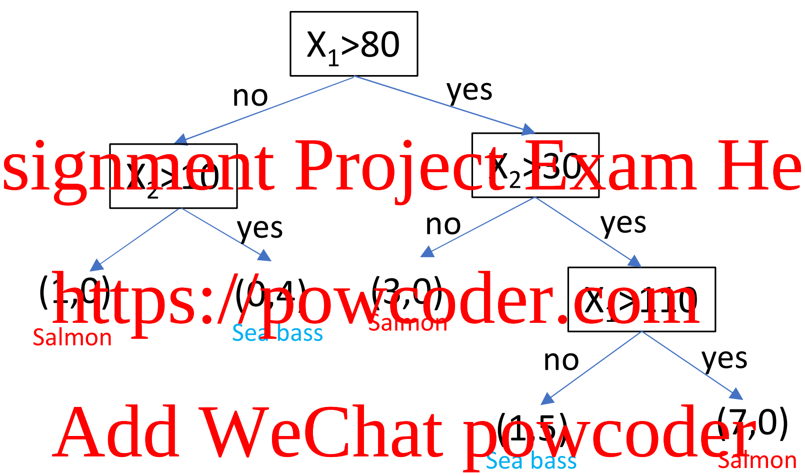
- Let's go back to the fish example with two types of fish, i.e. "*salmon*" and "*sea bass*", and assume we have two features:
    - Length
    - Width
- We want to classify fishes based on these two features







Decision Trees: Example



- For any new sample, we start from the top of the tree, answer the questions, follow the appropriate path to the bottom, and then decide the label

Decision Trees Overview

- Approach
  - classify a pattern through a sequence of questions, the next question asked depends on answer to current question
  - sequence of questions displayed in a directed ***decision tree*** or simply ***tree***

- Structure
  - nodes in the tree represent features
  - leaf nodes contain the class labels
  - one feature (or a few) at a time to split search space of patterns
  - each branching node has one child for each possible value of the parent feature

- Classification
  - begins at the root node, follows the appropriate link to a leaf node
  - assigns the class label of the leaf node to the test pattern

Decision Trees Construction

- Binary decision tree is a binary tree structure that has a decision function associated with each node
- Simple case: numeric feature values, decision function compares value of a feature to a threshold. The decision function selects left/right branch if the value of the feature is less / greater than the threshold

- Advantages: at each node, only one feature is used, and its threshold value need to be stored
- For any given set of training examples, there may be more than one possible decision tree to classify them, depending on feature order
- We must select features that give the `best' tree based on some criterion
- Smallest tree preferred

Decision Trees Construction

- Algorithm
  1. Select a feature to place at the node (the first one is the root)
  2. Make one branch for each possible value
  3. For each branch node, repeat step 1 and 2 using only those instances that actually reach the branch
  4. When all instances at a node have the same classification (or label), stop growing that part of the tree

- How to determine which feature to split on?
  – One way is to use measures from information theory
  – ***Entropy*** and ***Information Gain***

Constructing Optimal Decision Tree

- Entropy
  - To construct optimal decision trees from training data, we need a definition of optimality
  - One simple criterion is ***entropy***, based on information theory
    The entropy of a set of events x = {x_1, x_2, ..., x_n} is:

$$H(x) = \sum_{i=1}^{n} -P(x_i) \log_2 P(x_i)$$

where $P(x_i)$ is the probability of event x_i

  - Entropy may be viewed as the average uncertainty of the information source. If the information source is homogeneous and has no uncertainty, then entropy is 0 and if the source information is uncertain then entropy is higher than zero.

Decision Trees: Entropy

Example of entropy computations:

Let us look at the fish example

P(class)	Salmon	Sea bass	Others
Prior	0.3	0.6	0.1

The entropy or uncertainty of information (type of fish) is:

$$H(x) = -[0.3 \times \log(0.3) + 0.6 \times \log(0.6) + 0.1 \times \log(0.1)] = 1.29$$

Decision Trees: Information Gain

Information gain is an entropy-based measure to evaluate features and produce optimal decision trees

If S is a set of training samples and f is one feature of the samples, then Information Gain w.r.t. feature f:

$$IG(S, f) = H(S) - H(S|f)$$

$$IG(S, f) = Entropy(S) - \sum_{f_a \in values(f)} \frac{|S_{f_a}|}{|S|} Entropy(S_{f_a})$$

Use the feature with highest information gain to split on:

- Prior probabilities can be estimated using frequency of associated events in the training data

Decision Trees: Information Gain Example

- Let us look at the fish example with two features of "*length*" and "*width*" again, but for the sake of simplicity, we assume three categories for each feature:

$$x_1 \in \{Small, Medium, Large\}$$
$$x_2 \in \{Small, Medium, Large\}$$

- This table is the list of our 15 observations/samples from two classes of "*salmon*" and "*sea bass*"

x_1	x_2	Type
S	S	Salmon
M	S	Salmon
M	S	Salmon
S	M	Sea bass
S	L	Sea bass
S	M	Sea bass
M	M	Sea bass
M	L	Sea bass
L	M	Salmon
L	M	Salmon
L	L	Salmon
S	L	Sea bass
M	L	Sea bass
M	M	Sea bass
M	L	Sea bass

Decision Trees: Information Gain Example

- Before selecting the first feature we need to know the base entropy. There are 6 salmon and 9 sea bass in the sample so:

$$P(salmon) = \frac{6}{15} = 0.4, P(Sea\ bass) = \frac{9}{15} = 0.6$$

$$H_{base} = -0.6\log(0.6) - 0.4\log(0.4) = 0.97$$

- To estimate $IG(S, x_1)$ we need to use frequency table for x_1 to compute $H(S|x_1)$

		Type		
		Salmon	Sea bass	
x_1	S	1	4	5
	M	2	5	7
	L	3	0	3
				15

x_1	x_2	Type
S	S	Salmon
M	S	Salmon
M	S	Salmon
S	M	Sea bass
S	L	Sea bass
S	M	Sea bass
M	M	Sea bass
M	L	Sea bass
L	M	Salmon
L	M	Salmon
L	L	Salmon
S	L	Sea bass
M	L	Sea bass
M	M	Sea bass
M	L	Sea bass

Decision Trees: Information Gain Example

$$H(S|x_1) = \frac{5}{15}Ent.(1,4) + \frac{7}{15}Ent.(2,5) + \frac{3}{15}ENt.(3,0) = 0.64$$

$$IG(S, x_1) = H_{base} - H(S|x_1) = 0.97 - 0.64 = 0.33$$

- Similarly for $H(S|x_2)$:

$$H(S|x_2) = \frac{6}{15}Ent.(2,4) + \frac{6}{15}Ent.(1,5) + \frac{3}{15}ENt.(3,0) = 0.62$$

$$IG(S, x_2) = H_{base} - H(S|x_2) = 0.97 - 0.62 = 0.35$$

		Type		
		Salmon	Sea bass	
x_1	S	1	4	5
	M	2	5	7
	L	3	0	3
				15

- $IG(S, x_2) > IG(S, x_1)$, so the decision tree starts with spliting x_2 and will repeat the same procedure at every node
- divide the dataset by its branches and repeat the same process on every branch.
- A branch with entropy more than 0 needs further splitting.

Decision Trees Summary

- Pros:
  - Easy to interpret
  - Can handle both numerical and categorical data
  - It can handle outliers and missing values
  - Gives information on importance of features (feature selection)

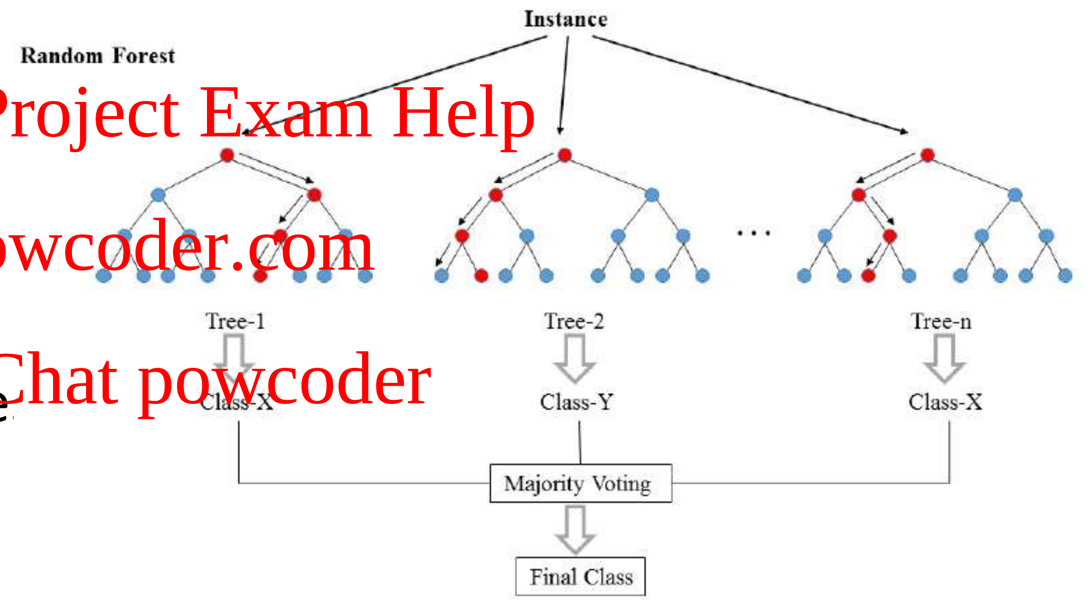
- Cons:
  - Tends to overfit
  - Only axis-aligned splits
  - Greedy algorithm (may not find the best tree)

Ensemble Learning

- Ensemble learning combines multiple models to improve predictive performance, compared to those obtained from any of the constituent models
- Multiple models can be created using
  - different classifiers/learning algorithms
  - different parameters for the same algorithm
  - different training examples
  - different feature sets

Random Forests

- Random forests is an ensemble learning method
- constructs an ensemble of decision trees by training
- output is the mode of all classes output by individual trees
- They overcome the decision trees' habit of overfitting



https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

Random Forests: Breiman's algorithm

Training

- Let N be number of training instances, M the number of features
- Sample N instances at random with replacement from the original data (bootstrap aggregating or bagging)
- At each node, m<<M features are selected at random out of the M, and the best split on these m is used to split the node (the value of m is held constant during the forest growing)
- Each tree is grown to the largest extent possible (no pruning)

Testing

- A new sample is pushed down a tree, which is assigned the label of the training samples in the terminal node it ends up in
- Iterate over all trees in the ensemble, report the mode vote of all trees as the random forest prediction

Random Forests

- The forest error rate depends on two factors:
  - correlation between any two trees in the forest
    - Increased correlation increases the forest error rate

  - strength of each individual tree in the forest
    - stronger tree has low error rate

    - increasing the strength of individual trees decreases the forest error rate

- Parameter m
  - reducing m reduces both the correlation and the strength, increasing it increases both
  - somewhat in between is an "optimal" range of values for m

Random Forests

- Pros:
  - unexcelled in accuracy among current classical ML algorithms for many problems
  - works efficiently on large datasets
  - handles thousands of input features without feature selection
  - handles missing values effectively
- Cons:
  - less interpretable than an individual decision tree
  - More complex and more time-consuming to construct than decision trees

Random Forests

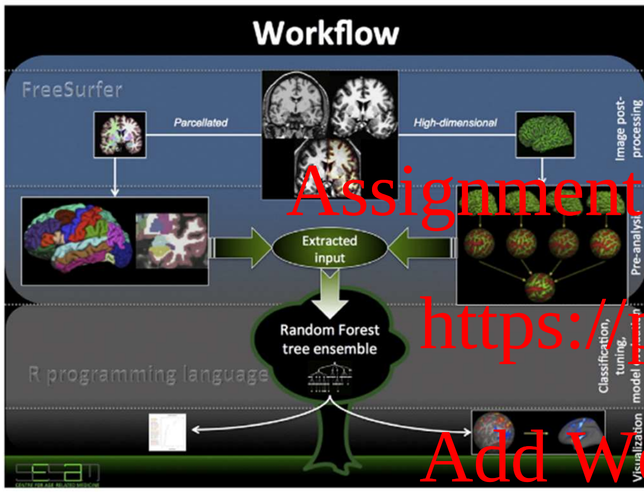


Fig. A1. Workflow diagram. The diagram illustrates main steps of image post-processing and analysis. It starts and proceeds in two directions aimed at extraction of parcelled and high-dimensional measurements using Freesurfer software (blue box). After this part had completed, the extracted measures underwent steps for outlier detection, and the resulted output was used in further Random Forest classification runs (in R programming language – gray box). We additionally tuned our models using recursive feature elimination and m_{try}-parameter adjustment (which defines the number of predictors randomly sampled at each node of the classifier). Finally, feature importance vectors from the best models were either mapped into the brain space (for the high-dimensional data) or plotted (for the parcelled input).

Fig. 3. Cortical pattern of relevance for Alzheimer's disease detection: high-dimensional morphometric data. The figure illustrates regions, which were the most relevant for Alzheimer's disease detection based on the mean decrease of the Gini index (see Methods). In all three high-dimensional modalities, the pattern was AD-specific and included changes predominantly in temporal lobes (with maximum relevance of entorhinal region).

- RF is used to predict Alzheimer's disease
- Features: cortical thickness, Jacobian maps, sulcal depth

- Ref: Lebedev et. Al, "Random Forest ensemble for detection and prediction of Alzheimer's disease with a god between-cohort robustness", Neuroimage, 2014

References and Acknowledgements

- Shapiro and Stockman, Chapter 4
- Duda, Hart and Stork, Chapters 1, 2.1
- Hastie, Tibshirani and Friedman, "The elements of statistical learning", Chapters 2 and 12

- More references
  - Sergios Theodoridis, Konstantinos Koutroumbas, *Pattern Recognition*, 2009
  - Ian H. Witten, Eibe Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2005

- Some diagrams are extracted from the above resources