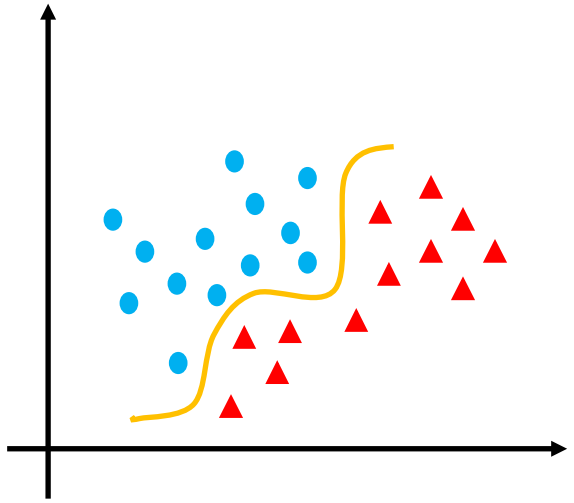


COMP9517: Computer Vision

Pattern Recognition I

Introduction

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

- **Examples**
  - Object recognition (e.g. image classification)
  - Text classification (e.g. spam/non-spam emails)
  - Speech recognition (e.g. automatic subtitling)
  - Event detection (e.g. in surveillance)
  - Recommender systems (e.g. in webshops)

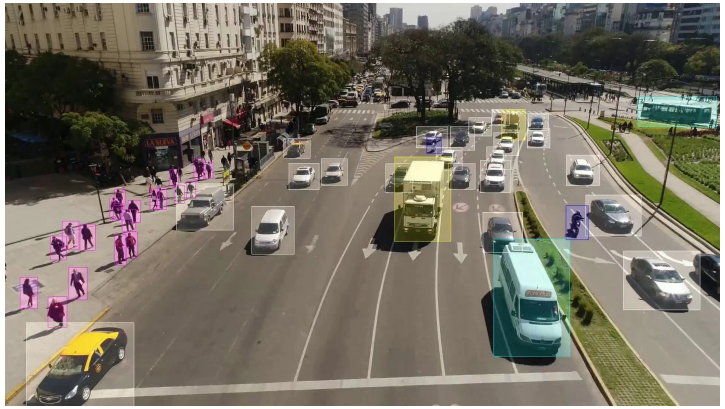
Pattern Recognition Categories

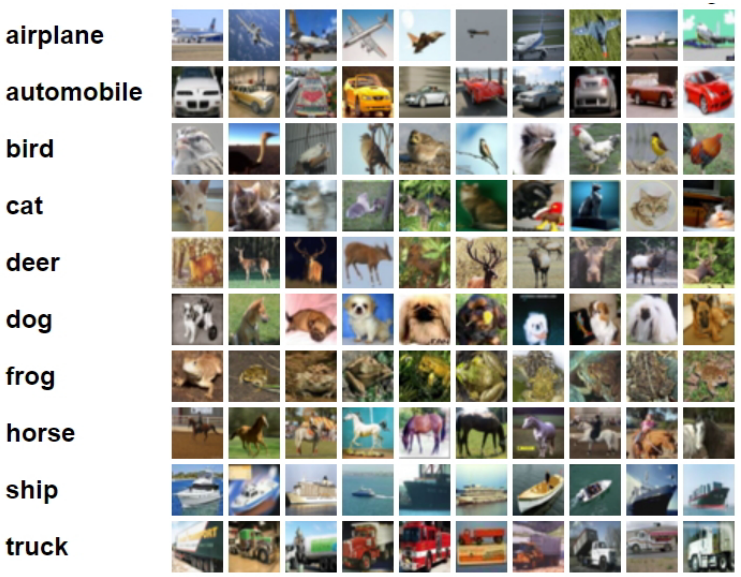
Based on different learning paradigms

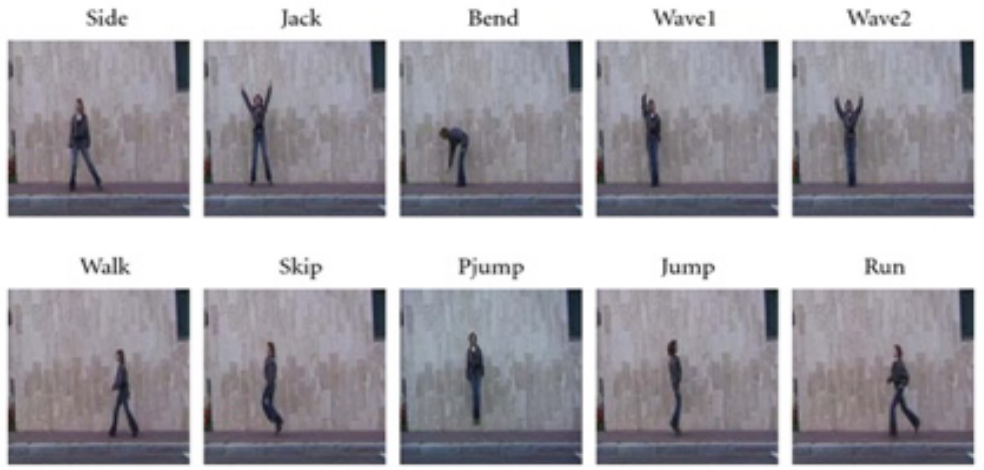
- **Supervised learning**
  Learning patterns in a set of data with available labels (ground truth)
- **Unsupervised learning**
  Finding patterns in a set of data without any labels available
- **Semi-supervised learning**
  Using a combination of labelled and unlabelled data to learn patterns
- **Weakly supervised learning**
  Using noisy / limited / imprecise supervision signals in learning of patterns

Applications in Computer Vision

- Making decisions about image content
- Classifying objects in an image
- Recognising activities

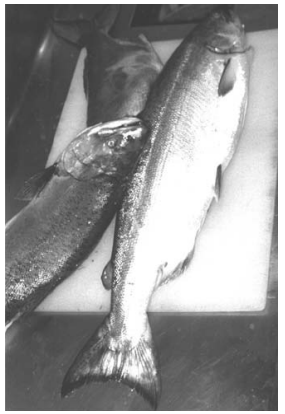


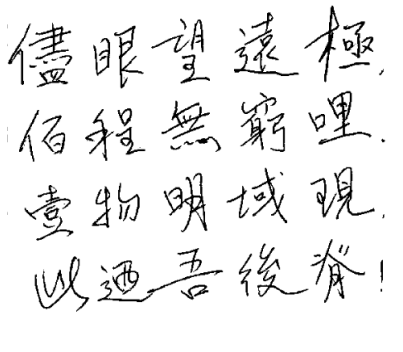


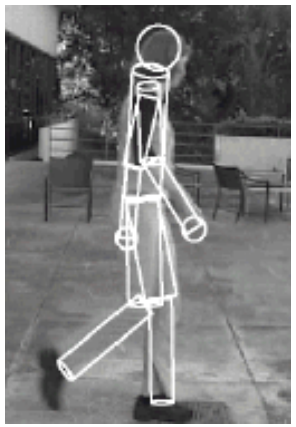


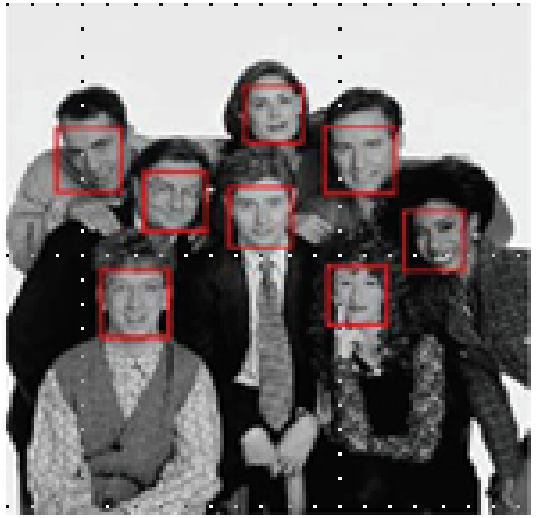
Applications in Computer Vision

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

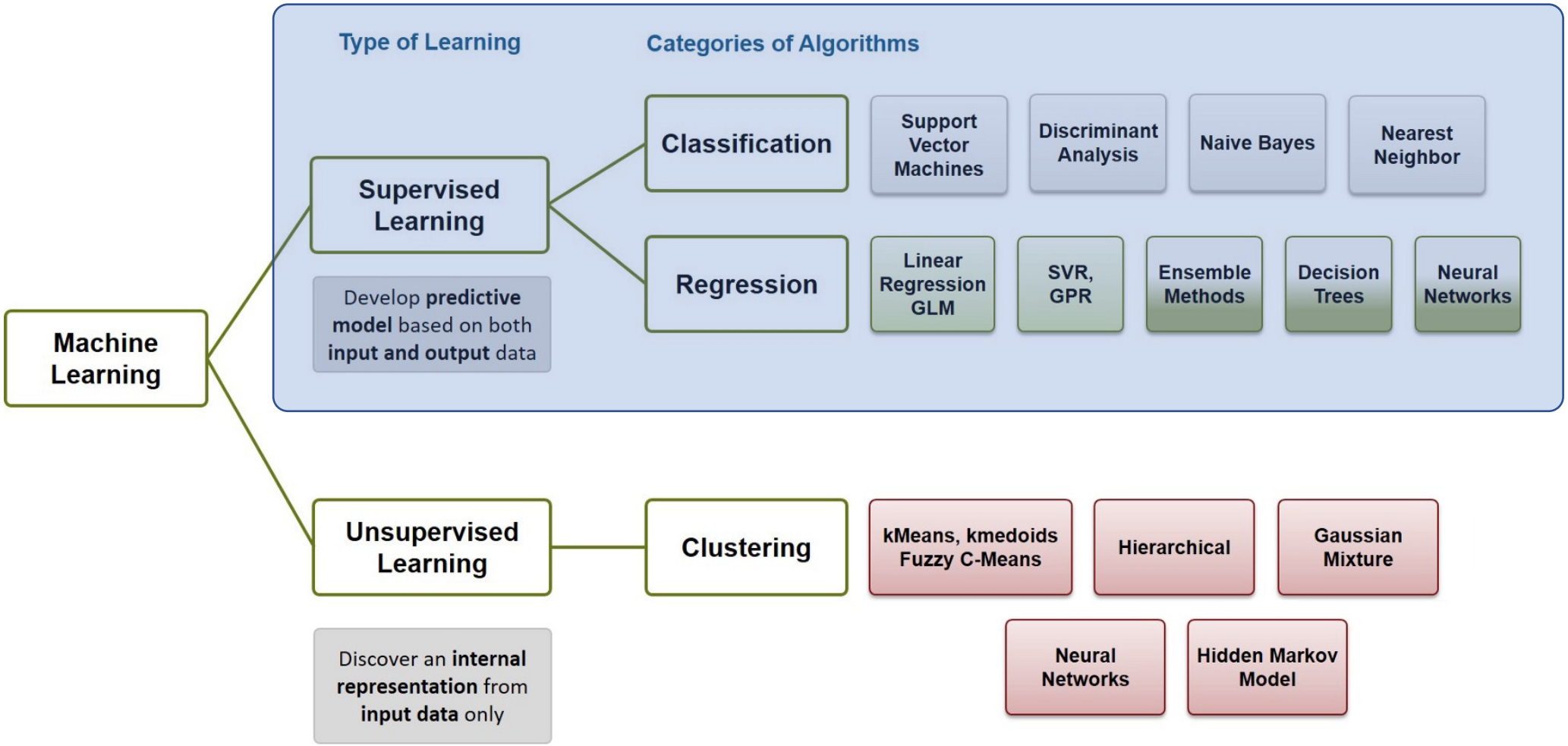








Pattern Recognition Overview



Pattern Recognition (First Lecture)

- Pattern recognition concepts
  - Definition and description of basic terminology
  - Recap of feature extraction and representation

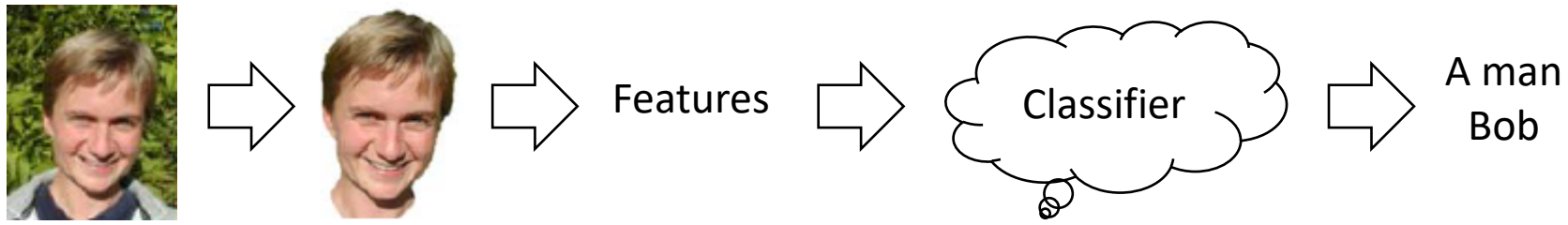
- Supervised learning for classification
  - Nearest class mean classification
  - K-nearest neighbours classification
  - Bayesian decision theory and classification
  - Decision trees for classification
  - Ensemble learning and random forests

Pattern Recognition (Second Lecture)

- Supervised learning for classification
  - Linear classification
  - Support vector machines
  - Multiclass classification
  - Classification performance evaluation
- Supervised learning for regression
  - Linear regression
  - Least-squares regression
  - Regression performance evaluation

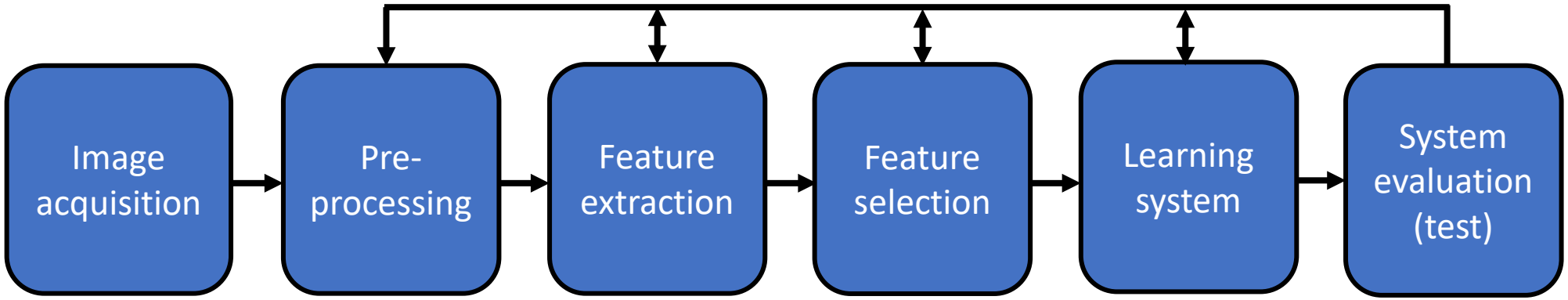
Pattern Recognition Concepts

- **Objects** are (identifiable) physical entities of which images are taken
- **Regions** (ideally) correspond to objects after image segmentation
- **Classes** are disjoint subsets of objects sharing common features
- **Labels** are associated with objects and indicate to which class they belong
- **Classification** is the process of assigning labels to objects based on features
- **Classifiers** are algorithms/methods performing the classification task
- **Patterns** are regularities in object features and are used by classifiers



Pattern Recognition Systems

- Basic stages involved in the design of a classification system

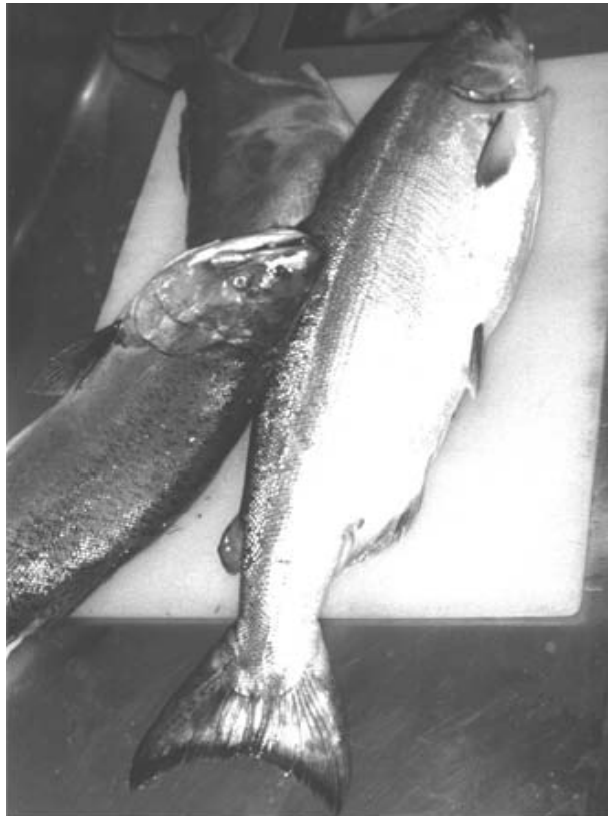


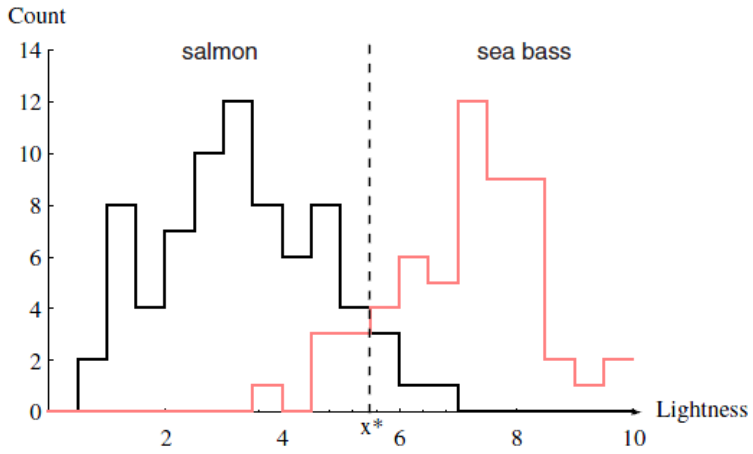
More Pattern Recognition Concepts

- **Pre-processing** aims to enhance images for further processing
- **Feature extraction** reduces the data by measuring certain properties
- **Feature descriptors** represent scalar properties of objects
- **Feature vectors** capture all the properties measured from the data
- **Feature selection** aims to keep only the most descriptive features
- **Models** are (mathematical or statistical) descriptions of classes
- **Training samples** are objects with known labels used to build models
- **Cost** is the consequence of making an incorrect decision/assignment
- **Decision boundary** is the demarcation between regions in feature space

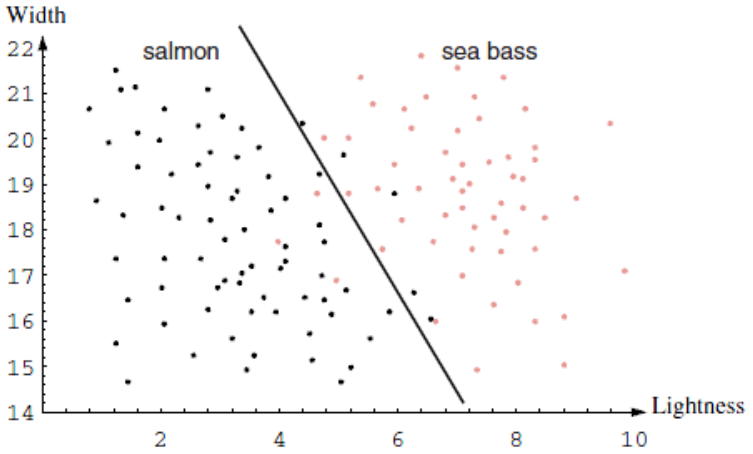
Pattern Recognition Example

salmon or sea bass?





1D feature space



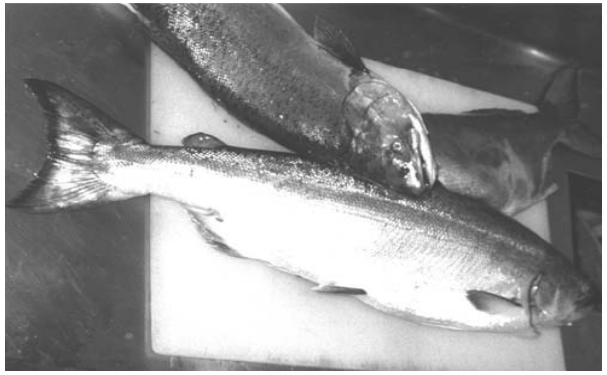
2D feature space

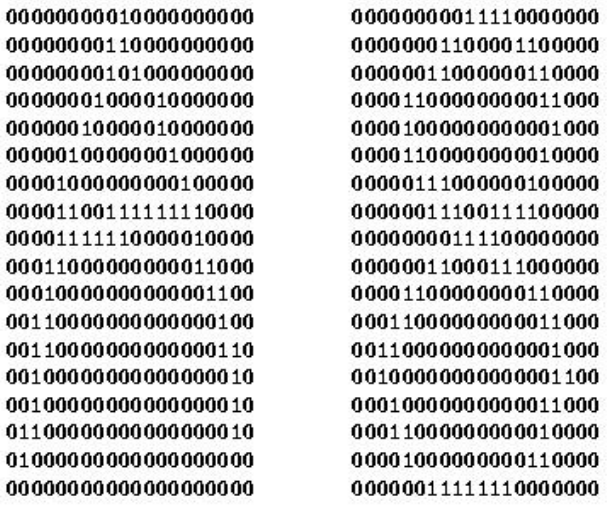
Feature Vector Representation

- Vector $x = [x_1, x_2, \ldots, x_d]$ where each x_j is a feature
  - Object measurement
  - Count of object parts
  - Colour of the object
  - …

  Features represent knowledge about the object and go by other names such as predictors, descriptors, covariates, independent variables…

- **Feature vector examples**
  - For fish recognition: [length, colour, lightness, …]
  - For letter/digit recognition: [holes, moments, SIFT, …]





Feature Extraction

- Characterise objects by **measurements** that are
  - Similar for objects in the same class/category
  - Different for objects in different classes
- Use **distinguishing features**
  - Invariant to object position (translation)
  - Invariant to object orientation (rotation)
  - Invariant to … (depends on the application)
  - Good examples are shape, colour, texture
- Design of features **often based on prior experience** or intuition

Feature Extraction

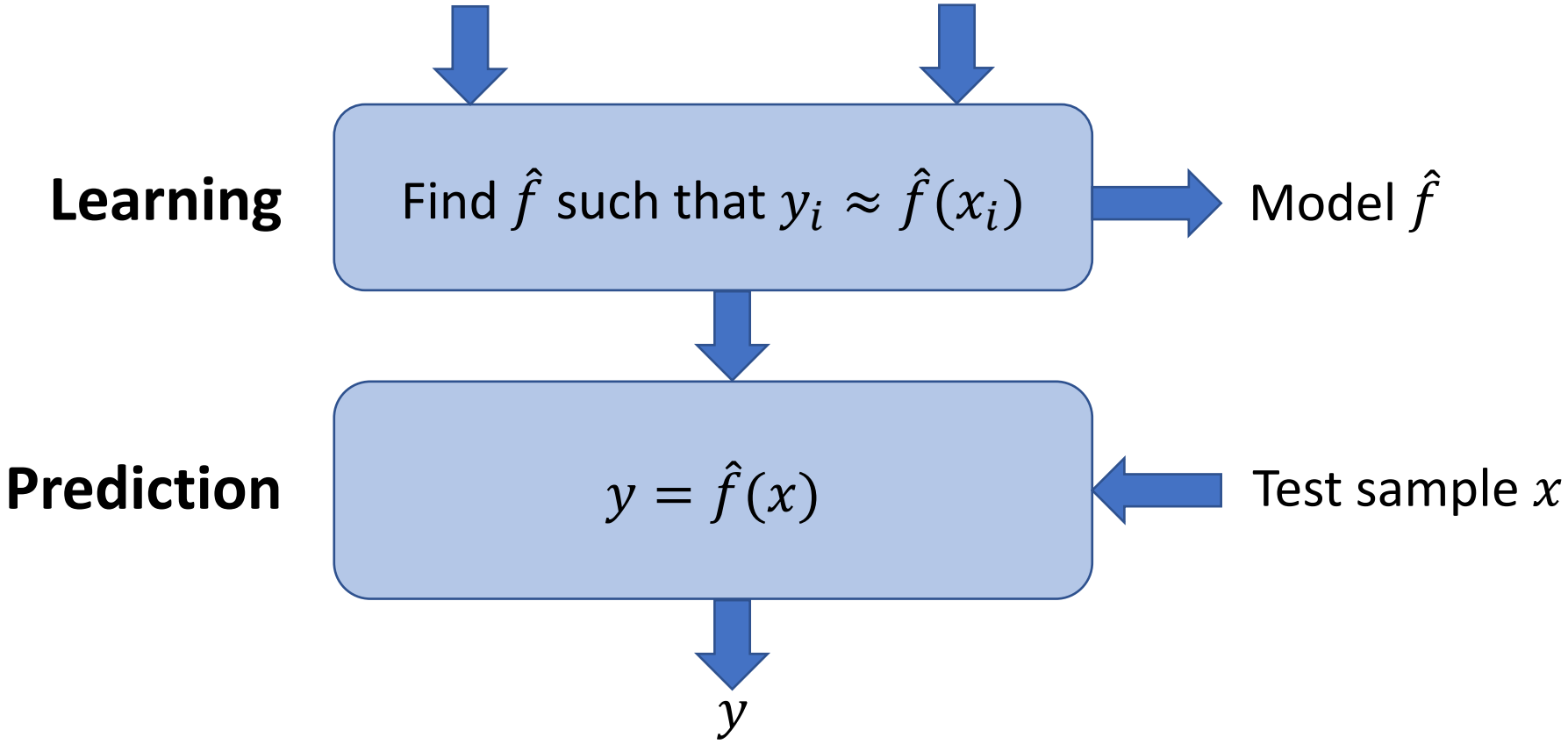
- Select features that are **robust to**
  - Rigid transformations (translation and rotation)
  - Occlusions and other 3D-to-2D projective distortions
  - Non-rigid/articulated object deformations (e.g. fingers around a cup)
  - Variations in illumination and shadows
- Feature selection is problem- and domain-dependent and **requires domain knowledge**
- Classification techniques can help to **make feature values less noise sensitive** and to **select valuable features** out of a larger set

Supervised Learning Overview

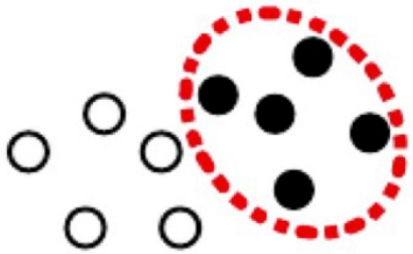
Feature space X
Label space Y

Functions
$f \in F: X \to Y$

Training set
$\{(x_i, y_i) \in X, Y\}$

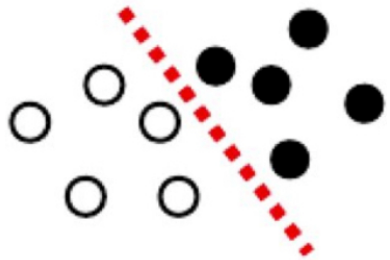


Pattern Recognition Models

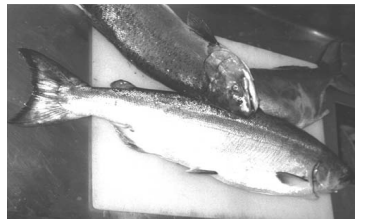


- **Generative models**
  - Model the "mechanism" by which the data was generated
  - Represent each class by a probabilistic "model" $p(x|y)$ and $p(y)$
  - Obtain the joint probability of the data as $p(x, y) = p(x|y)p(y)$
  - Find the decision boundary implicitly via the most likely $p(y|x)$
  - Applicable to unsupervised learning tasks (unlabelled data)

- **Discriminative models**
  - Focus on explicit modelling of the decision boundary
  - Applicable to supervised learning tasks (labelled data)



Classification

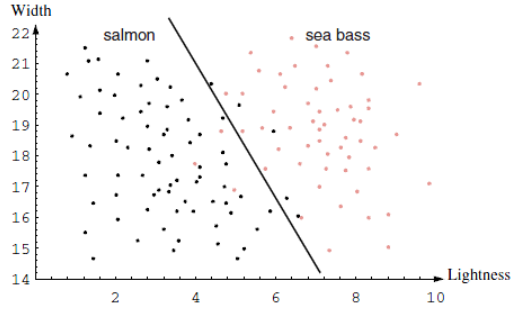


$x \rightarrow y = \text{"salmon"}$

- 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 the probability for each possible class**

$$x \rightarrow \begin{cases} p_{\text{salmon}} = 0.7 \\ p_{\text{bass}} = 0.3 \end{cases}$$

- Difficulties are caused by **variability in feature values for objects in the same class versus objects in different classes**
  - Variability 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 object



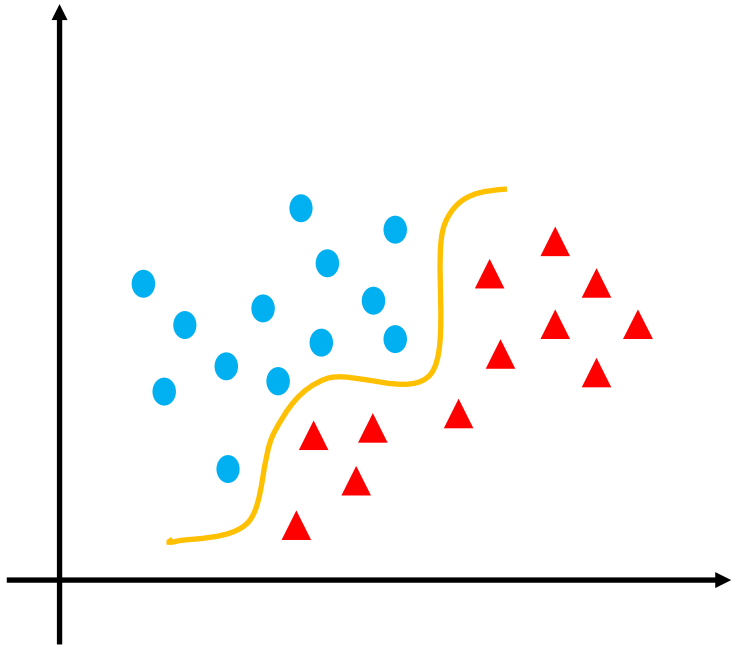
Binary Classification

- Given a training set of N observations:

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

- Classification is the problem of estimating:

$$y_i = f(x_i)$$



Nearest Class Mean Classifier

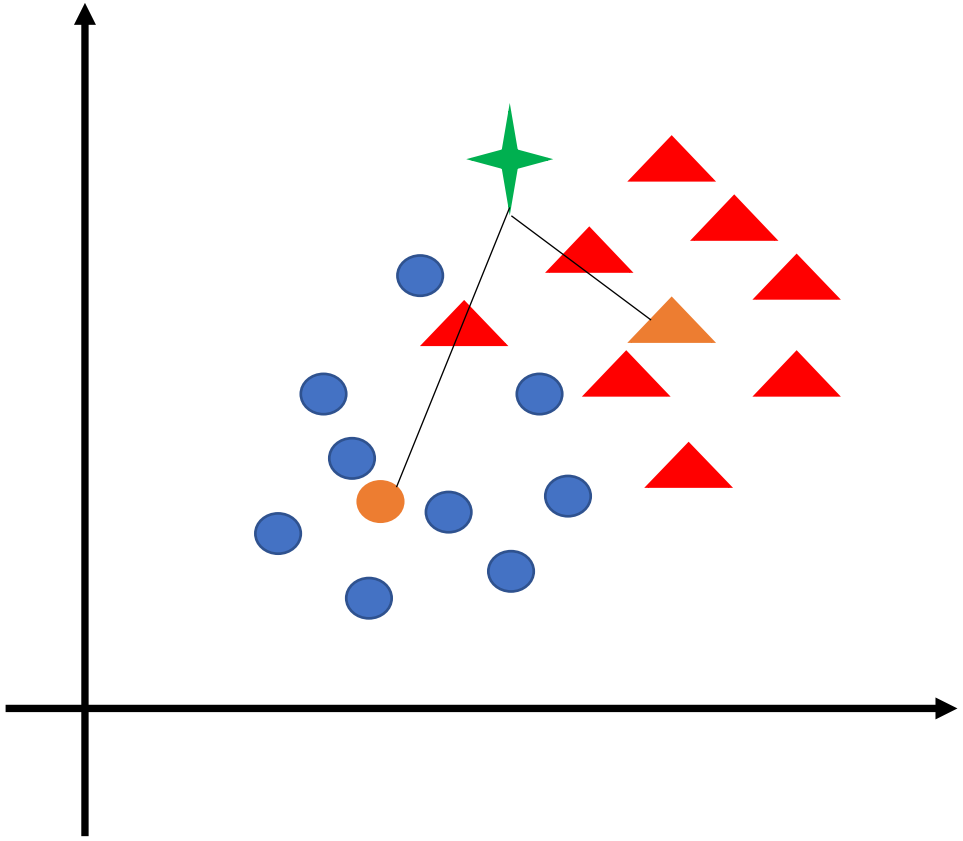
- Based on the **minimum distance principle**, where the class exemplars are just the class centroids (or means)

- **Training**: Given training sample pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, the centroid for each class k is obtained as

$$\mu_k = \frac{1}{|c_k|} \sum_{x_i \in c_k} x_i$$

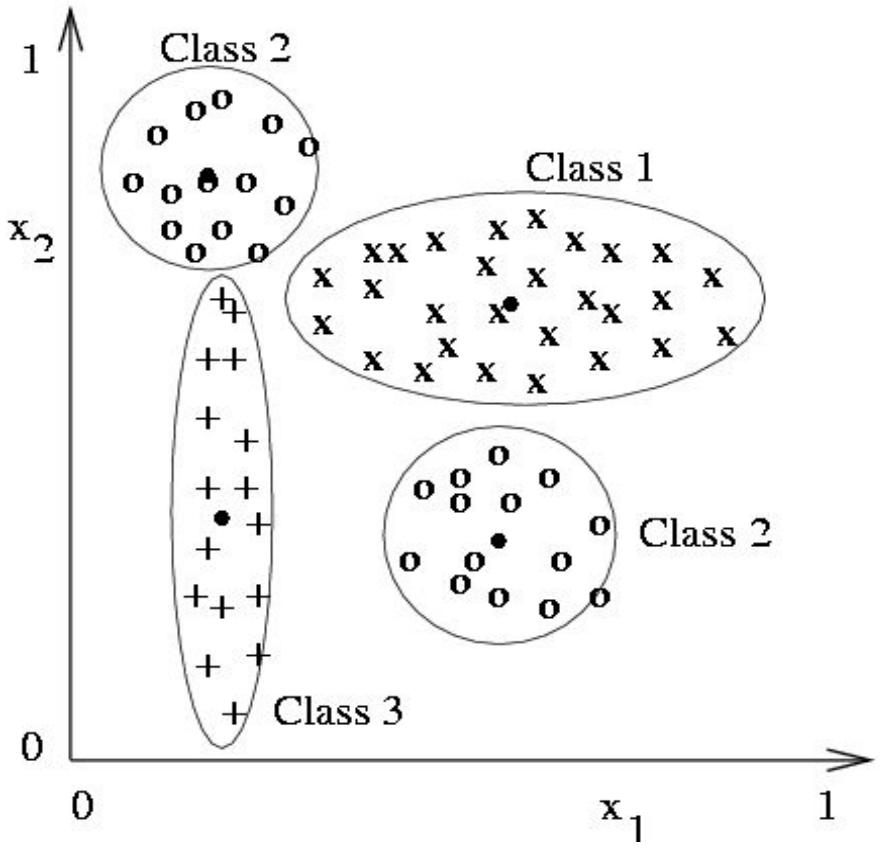
- **Testing**: Each unknown object with feature vector x is classified as class k if x is closer to the centroid of class k than to any other class centroid

Nearest Class Mean Classifier



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

Nearest Class Mean Classifier



- Class 2 has two modes… where is its centroid?
- If modes are detected, two subclass centroids can be used

Nearest Class Mean Classifier

- **Pros**
  - ✓ Simple
  - ✓ Fast
  - ✓ Works well when classes are compact and far from each other

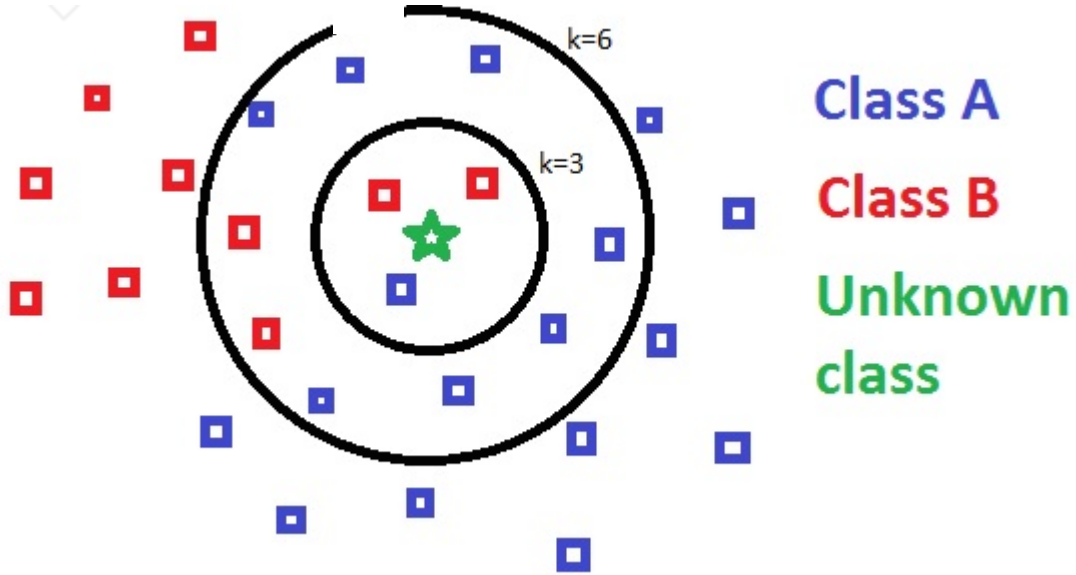
- **Cons**
  - × Poor results for complex classes (multimodal, non-spherical)
  - × Cannot handle outliers and noisy data well
  - × Cannot handle missing data

K-Nearest Neighbours Classifier

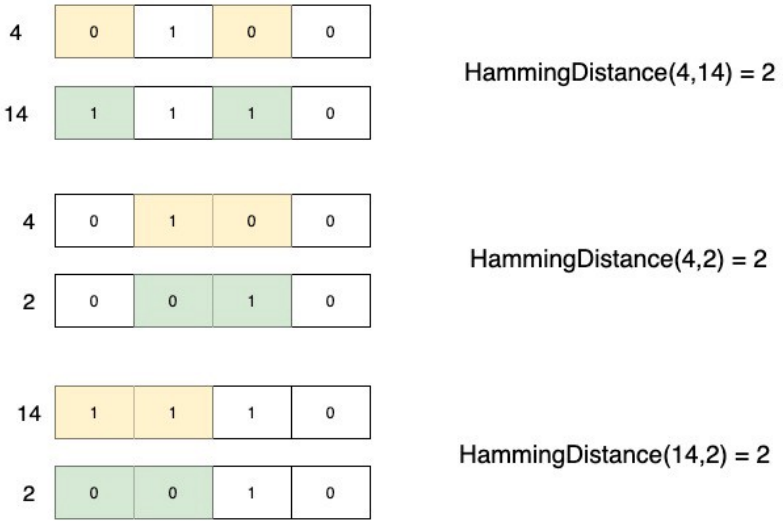
- K-NN is a classifier that decides the class label for a sample **based on the K nearest samples** in the data set
- For every new test sample, **the distances between the test sample and all training samples are computed**, and **the K nearest training samples are used** to decide the class label of test sample
- The sample will be assigned to the class which has the **majority of members in the neighbourhood**
- The neighbours are selected from a set of **training samples for which the class is known**

K-Nearest Neighbours Classifier

- **Euclidean distance** is commonly used **for continuous variables**
- **Hamming distance** is commonly used **for discrete variables**



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



Hamming distance

Image credit: https://medium.com/geekculture/total-hamming-distance-problem-1b74decd71c9

K-Nearest Neighbours Classifier

- **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 data sets
  - × Needs homogeneous (similar nature) feature types and scales
  - × Does not perform well when the number of variables grows (curse of dimensionality)
  - × Finding the optimal K (number of neighbours) to use can be challenging

K-Nearest Neighbours: An Application

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

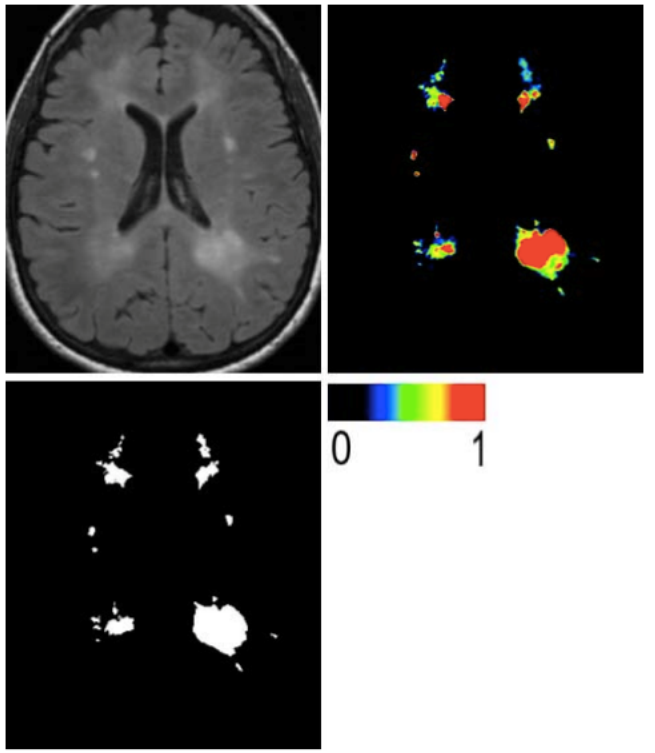
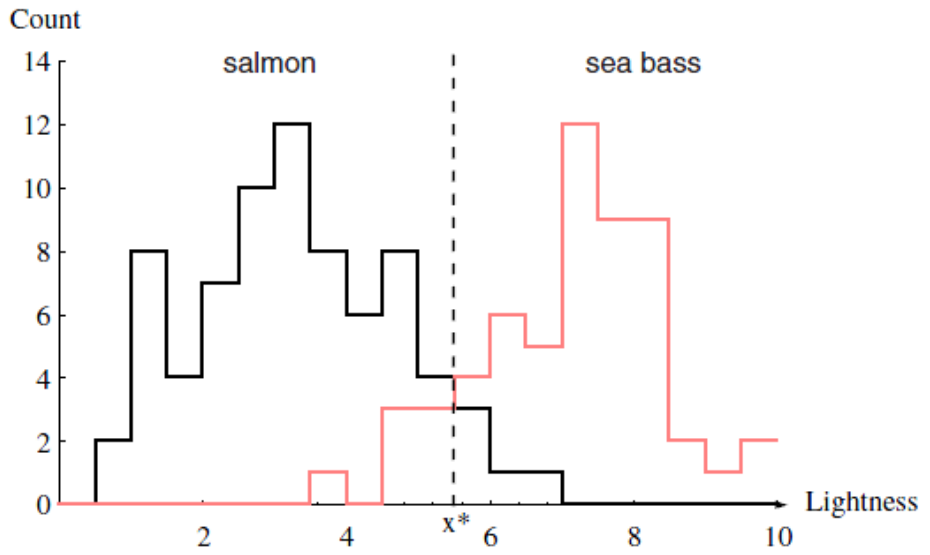


Figure 1 MS-lesion segmentation results. Top left: FLAIR image; top right: probabilistic segmentation, showing probability of lesion per voxel (see color bar); down left: binary segmentation, derived from probabilistic segmentation with threshold 0.4.

https://www.midasjournal.org/browse/publication/610

Bayesian Decision Theory

- Classifier decisions may or may not be correct, so they should be probabilistic (soft decisions rather than hard decisions)
- Probability distributions may be used to make classification decisions with the least expected error rate
- Introducing prior knowledge



Bayesian Decision Theory

- **Bayesian classification** assigns an object into the class to which it most likely belongs based on observed features
- Assume the following to be known:
  - Prior probability $p(c_i)$ for each class c_i
  - Class conditional distribution $p(x|c_i)$
- Compute the posterior probability $p(c_i|x)$ as follows:
  If all the classes are disjoint, by Bayes Rule, the posterior probabilities are given by

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

$$p(x, c_i) = p(x|c_i)p(c_i) = p(c_i|x)p(x)$$
$$p(x) = \sum_j p(x, c_j) = \sum_j p(x|c_j)p(c_j)$$

Bayesian Decision Theory

- Given an observation x, for which $p(c_1|x)$ is greater than $p(c_2|x)$, we would naturally prefer to decide that it belongs to c_1
- So the decision rule is: $c = \arg\max_i(p(c_i|x))$
- This is equivalent to: $c = \arg\max_i(p(x|c_i)p(c_i))$

$$p(c_i|x) = \frac{p(x|c_i)p(c_i)}{\sum_j p(x|c_j)p(c_j)} = \frac{p(x|c_i)p(c_i)}{p(x)} \propto p(x|c_i)p(c_i)$$

Bayesian Decision Theory

- For a given x, the probability of error is

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

- So, clearly, we can minimise the probability of error by deciding c_1 if $p(c_1|x) > p(c_2|x)$, and c_2 if $p(c_2|x) > p(c_1|x)$
- This is called the **Bayes decision rule**

Bayesian Decision Rule: Example

- Suppose 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	Other
Prior	0.3	0.6	0.1

- If we were to decide based only on the prior, our best bet would be to always classify as sea bass
  - This is called **decision rule based on prior**
  - It never predicts other classes
  - This can behave very poorly

Bayesian Decision Rule: Example

- Let us use some feature, e.g. **length**, to add more information
- From experience we know the **class conditionals** for 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

$$p(c_i|x) \propto p(x|c_i)p(c_i)$$

Bayesian Decision Rule: Example

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

$$p(c_i = \text{salmon}|x = 70\text{ cm}) \propto p(70\text{ cm}|\text{salmon}) * p(\text{salmon}) = 0.4 * 0.3 = 0.12$$

$$p(c_i = \text{sea bass}|x = 70\text{ cm}) \propto p(70\text{ cm}|\text{sea bass}) * p(\text{sea bass}) = 0.5 * 0.6 = 0.30$$

$$p(c_i = \text{other}|x = 70\text{ cm}) \propto p(70\text{ cm}|\text{other}) * p(\text{other}) = 0.2 * 0.1 = 0.02$$

- Based on these probabilities, we predict the type as **sea bass**
- **Question**: What if $p(70\ cm|salmon)$= $p(70\ cm|sea\ bass)$=0.5? What is the effect of prior $\boldsymbol{p(c_i)}$?

Bayesian Decision Rule: Example

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

$$p(c_i = \text{salmon}|x = 70\text{ cm}) \propto p(70\text{ cm}|\text{salmon}) * p(\text{salmon}) = 0.4 * 0.3 = 0.12$$

$$p(c_i = \text{sea bass}|x = 70\text{ cm}) \propto p(70\text{ cm}|\text{sea bass}) * p(\text{sea bass}) = 0.5 * 0.6 = 0.30$$

$$p(c_i = \text{other}|x = 70\text{ cm}) \propto p(70\text{ cm}|\text{other}) * p(\text{other}) = 0.2 * 0.1 = 0.02$$

- Based on these probabilities, we predict 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 Risk

- If we only care about the classification accuracy (the prices of all types of fish are the same), then we can make the decision by **maximizing the posterior probability**
- If the prices are not the same, we **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 with action α_i is:

$$R(\alpha_i|x) = \sum_j \lambda(\alpha_i|c_j)p(c_j|x)$$

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

Bayesian Decision Risk: Example

- Suppose we have the following 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

$$
\begin{aligned}
R(\text{sell as salmon}|50 < x < 100) = {} & \lambda(\text{sell as salmon}|\text{salmon}) * p(\text{salmon}|50 < x < 100) + \\
& \lambda(\text{sell as salmon}|\text{sea bass}) * p(\text{sea bass}|50 < x < 100) + \\
& \lambda(\text{sell as salmon}|\text{other}) * p(\text{other}|50 < x < 100) \\
\propto {} & 0 * p(50 < x < 100|\text{salmon}) * p(\text{salmon}) + \\
& 2 * p(50 < x < 100|\text{sea bass}) * p(\text{sea bass}) + \\
& 3 * p(50 < x < 100|\text{other}) * p(\text{other}) \\
= {} & 0 * 0.4 * 0.3 + 2 * 0.5 * 0.6 + 3 * 0.2 * 0.1 = 0.66
\end{aligned}
$$

Bayesian Decision Risk: Example

$$R(\text{sell as salmon}|50 < x < 100) \propto 0.66$$
$$R(\text{sell as sea bass}|50 < x < 100) \propto 1.28$$
$$R(\text{sell as other}|50 < x < 100) \propto 4.5$$

- So, if the length of the fish was in the range of $[50,100]$, the loss would be minimized if the type was predicted as salmon
- Is the loss function in the example good? How to define the loss function?

Bayesian Decision Theory

- **Estimating the probabilities**
  - Values of $p(x|c_i)$ and $p(c_i)$ can be computed empirically from the samples
  - If we know that the distributions follow a parametric model (defining the model), we may estimate the model parameters using the samples (learning the parameters)

- **Example**
  - Suppose class i can be described by a normal distribution whose covariance matrix Σ_i is known but the mean μ_i is unknown
  - An estimate of the mean may then be the average of the labelled samples available in the training set: $\hat{\mu}_i = \bar{x}$

Bayesian Decision Rule Classifier

- **Pros**
  - ✓ Simple and intuitive
  - ✓ Considers uncertainties
  - ✓ Permits combining new information with current knowledge

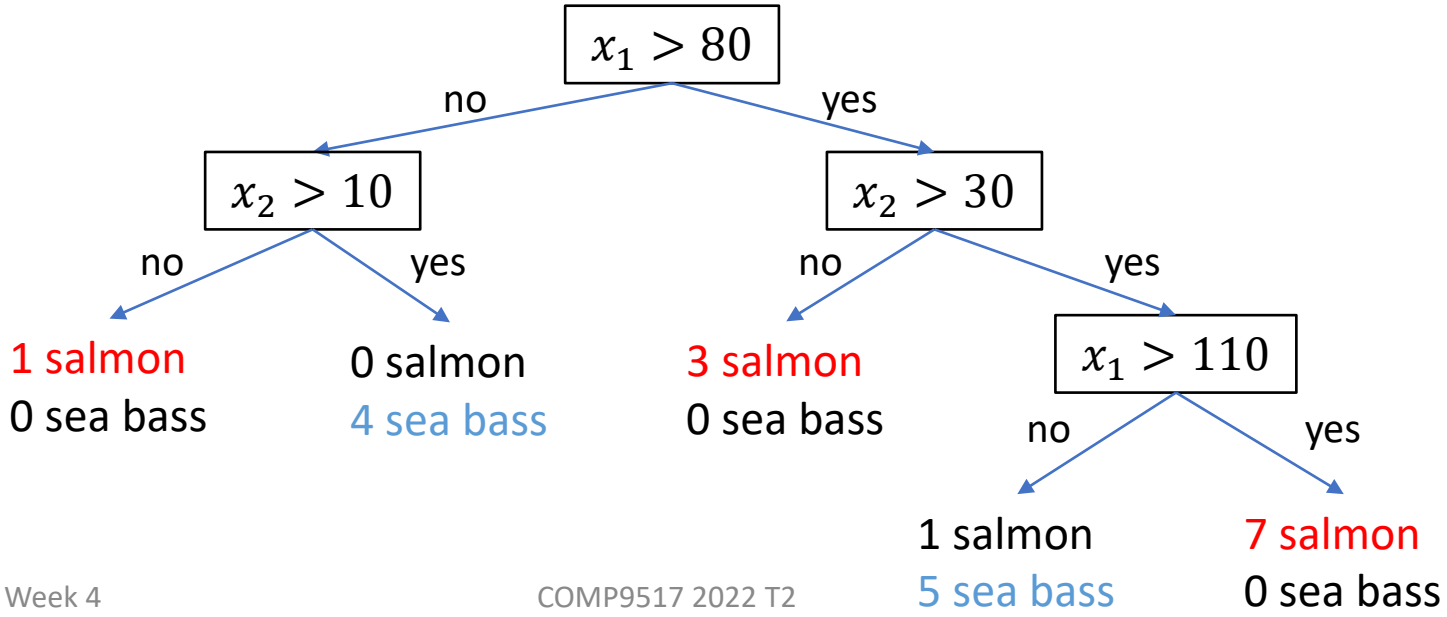
- **Cons**
  - × Computationally expensive
  - × Choice of priors can be subjective

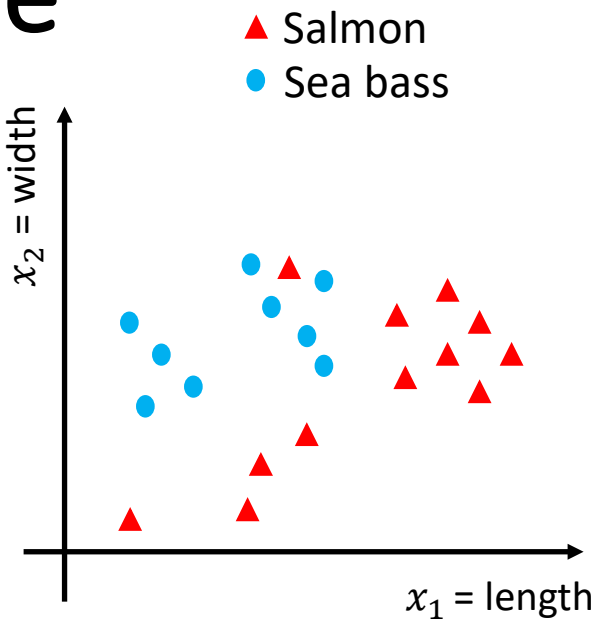
Decision Trees: Introduction

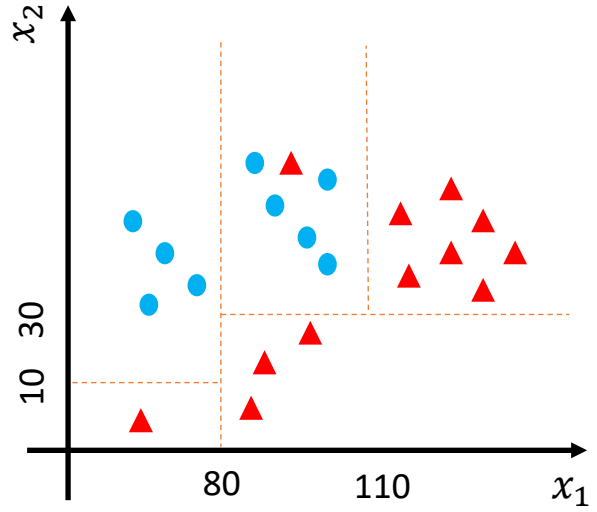
- Most pattern recognition methods address problems where **feature vectors are real-valued** and there exists some notion of a metric
- Some classification problems involve **nominal data**, **with discrete descriptors** and without a natural notion of similarity or ordering
  - Example: {high, medium, low}, {red, green, blue}
  - **Nominal data** are also called as **categorical data**
- Nominal data can be classified using **rule-based method**
- Continuous values can also be handled with rule-based method

Decision Trees: Example

- Suppose we have only two types of fish, **salmon** and **sea bass**, and assume we have only two features:
  - Length (x_1)
  - Width (x_2)
- We classify fish based on these two features

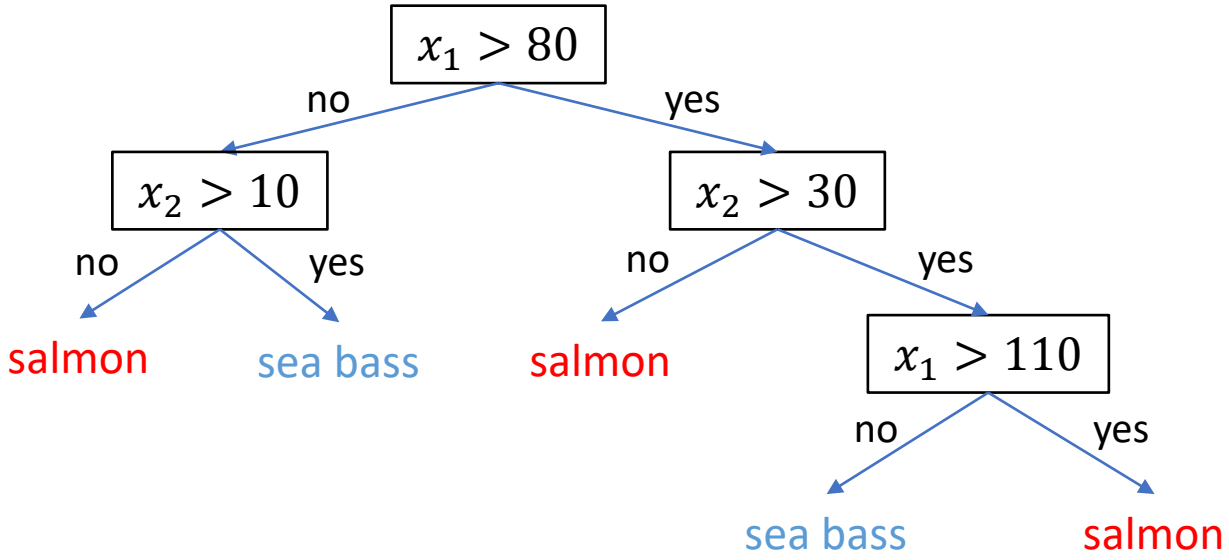






Decision Trees: Example

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



Decision Trees: Summary

- **Approach**
  - Classify a sample through a sequence of questions
  - 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
  - Each branching node has one child for each possible value of the parent feature
- **Classification**
  - Begins at the root node and follows the appropriate link to a leaf node
  - Assigns the class label of the leaf node to the test sample

Decision Trees Construction

- **Binary decision tree**: a binary tree structure that has a decision function associated with each node
- **Simple case**: numeric feature values and a decision function selects the left/right branch if the value of a feature is below/above a threshold
- **Advantages**: each node uses only one feature and one threshold value
- For any given set of training samples, **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
- Computationally the **smallest tree is 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 (nominal) or range (numerical)
  3. For each branch node, repeat step 1 and 2 using only those samples that actually reach the branch
  4. When all samples 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

- To construct optimal decision trees from training data we must define 'optimal'
- One simple criterion based on information theory is **entropy**

  The entropy of a set of events $y = \{y_1, y_2, \ldots, y_n\}$ is:

$$H(y) = \sum_{i=1}^{n} -p(y_i)\log_2 p(y_i)$$

  where $p(y_i)$ is the probability of event y_i

- Entropy may be viewed as the **average uncertainty** of the information source
  - If the source information has no uncertainty: $H = 0$
  - If the source information is uncertain: $H > 0$

Decision Trees: Entropy

Example of entropy computations

Let us look at the fish example

$p(c_i)$	Salmon	Sea bass	Other
Prior	0.3	0.6	0.1

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

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

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 the information gain with respect to feature f is:

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

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

Use the feature with highest information gain to split on

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

Decision Trees: Information Gain Example

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

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

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

- The table lists 15 observations/samples from two classes of **salmon** and **sea bass**

$$\#\text{Salmon} = 6$$

$$\#\text{Sea bass} = 9$$

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:

$$p(\text{salmon}) = 6/15 = 0.4$$
$$p(\text{sea bass}) = 9/15 = 0.6$$
$$H_{\text{base}} = -0.6\log(0.6) - 0.4\log(0.4) = 0.97$$

- To estimate $IG(S, x_1)$ we need to use the 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

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

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

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

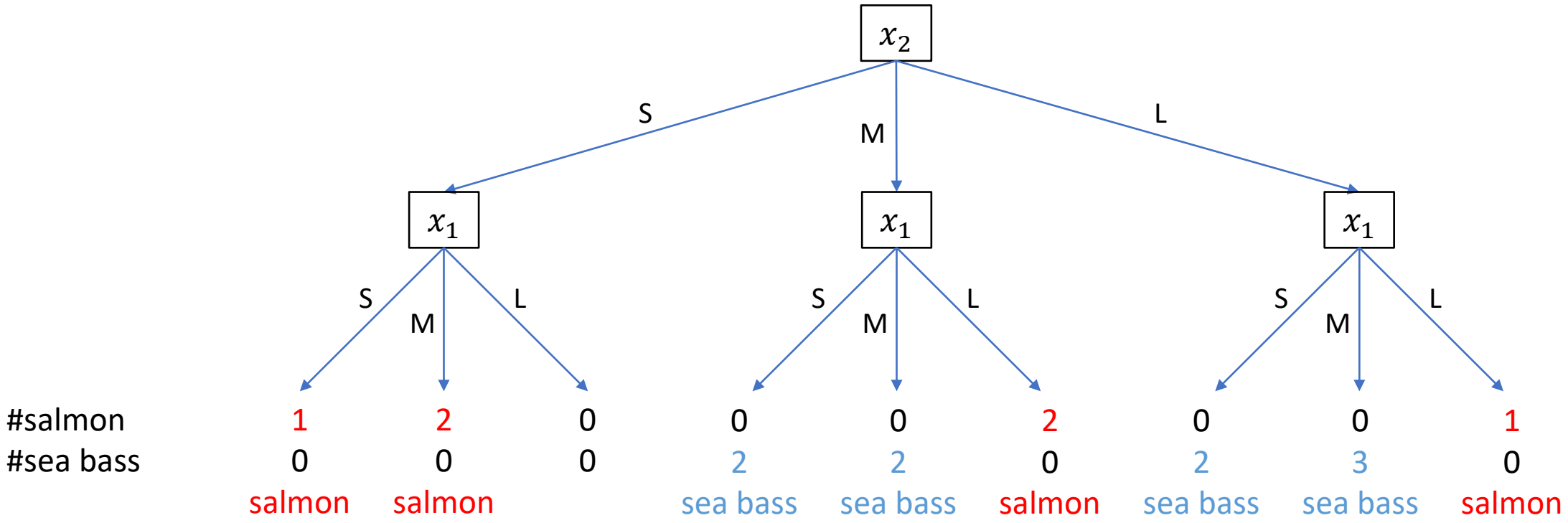
$$IG(S, x_2) = H_{\text{base}} - H(S|x_2) = 0.97 - 0.62 = 0.35$$

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

		Type		
		Salmon	Sea bass	
x_2	S	3	0	3
	M	2	4	6
	L	1	5	6
				15

- Since $IG(S, x_2) > IG(S, x_1)$ the decision tree starts with spliting x_2
- 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: Example



Decision Trees Classifier

- **Pros**
  - ✓ Easy to interpret
  - ✓ Can handle both numerical and categorical data
  - ✓ Robust to 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)

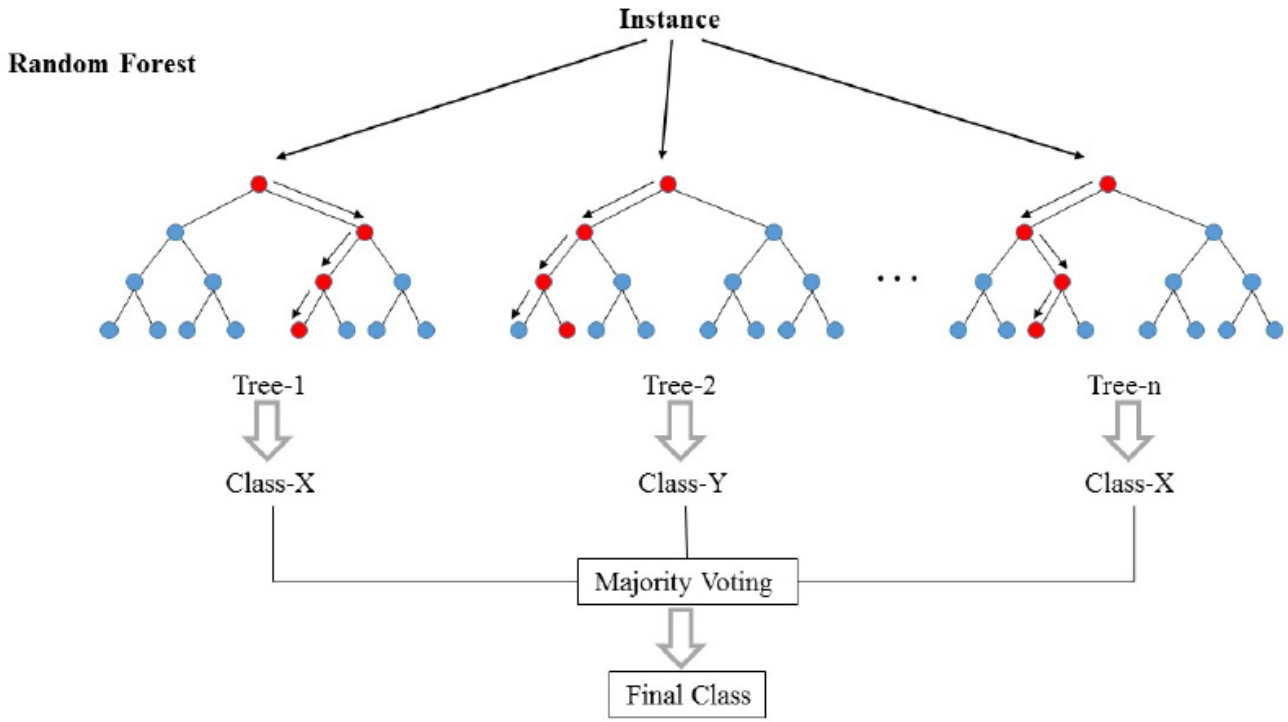
Ensemble Learning

- **Ensemble learning combines multiple models** to improve the 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
- Outputs the majority of all class outputs by individual trees
- Overcomes the decision trees' habit of overfitting



https://en.wikipedia.org/wiki/Random_forest

Random Forests: Breiman's algorithm

Training

- Let N be number of training instances and M the number of features (-- bagging)
- Sample N instances at random **with replacement** from the original data
- At each node select $m \ll M$ features at random and split on the best feature (-- "feature bagging")
- Grow each tree to the largest extent possible (no pruning)
- Repeat B times. Keep the value of m constant during the forest growing

Testing

- Push a new sample down a tree and assign the label of the terminal node it ends up in
- Iterate over all trees in the ensemble to get B predictions for the sample
- Report the majority vote of all trees as the random forest prediction

Random Forests

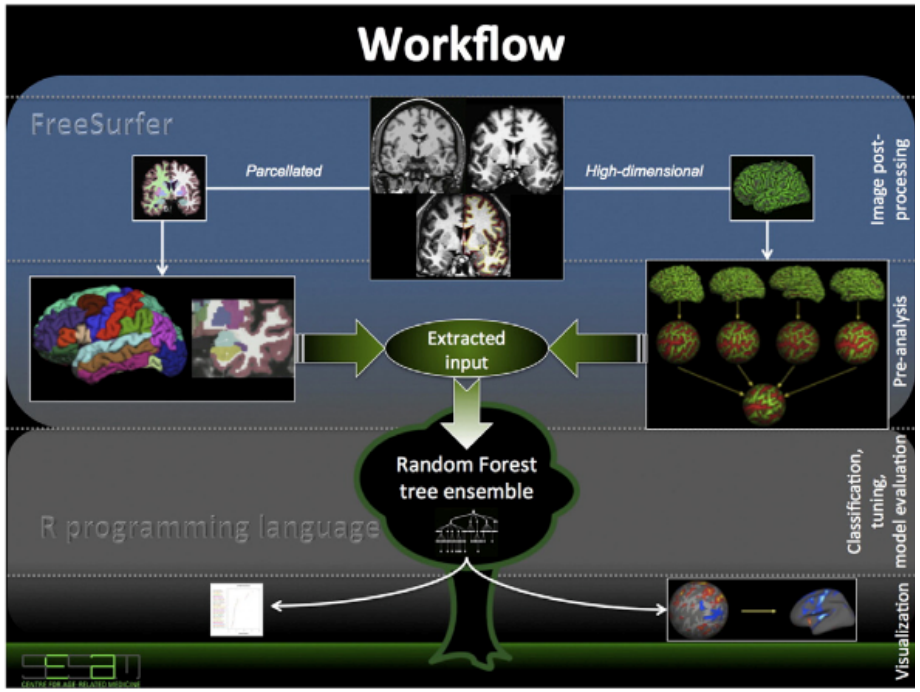
- The random forest error rate depends on two factors:
  1. Correlation between any two trees in the forest
     Increased correlation increases the forest error rate; uncorrelated trees lead to better generalization.
  2. Strength of each individual tree in the forest
     - Strong tree has low error rate
     - Increasing the strength of individual trees decreases the forest error rate
- Selecting parameter m
  - Reducing m reduces both the correlation and the strength
  - Increasing m increases both the correlation and the strength
  - Somewhere in between is an "optimal" range of values for m

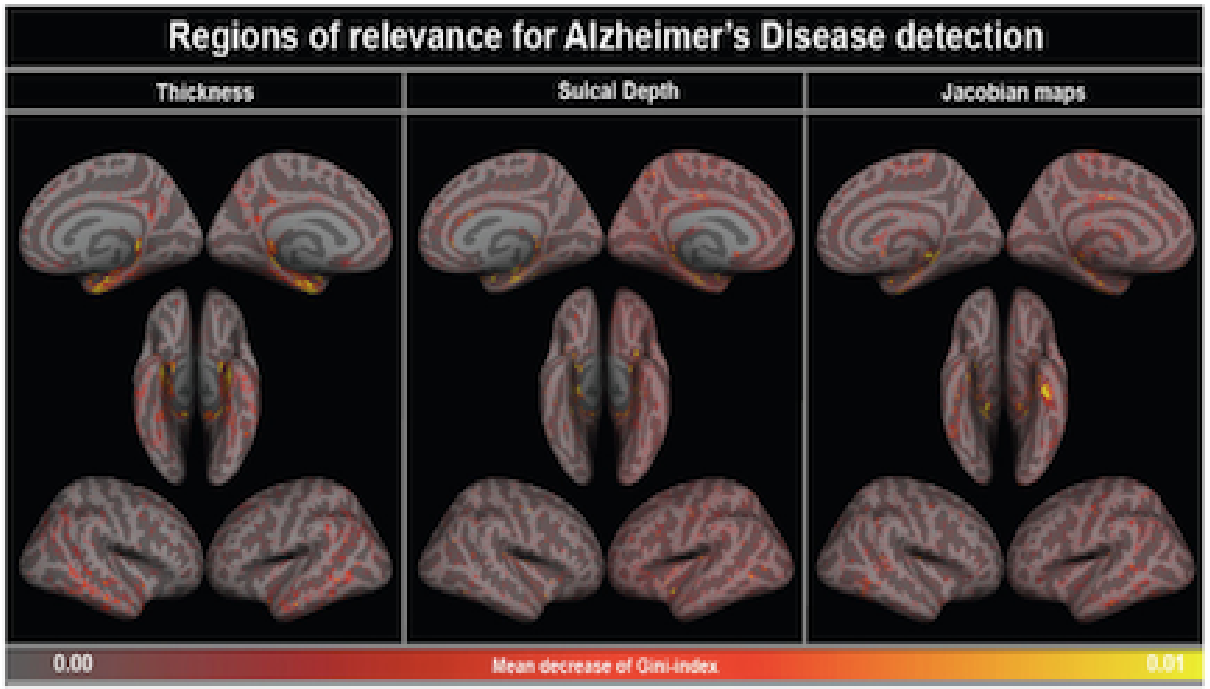
Random Forests

- **Pros**
  - ✓ High accuracy among traditional classification algorithms for many problems
  - ✓ Works efficiently and effectively 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: An Application





- Random forests for predicting Alzheimer's disease
- Features: cortical thickness, Jacobian maps, sulcal depth

https://doi.org/10.1016/j.nicl.2014.08.023

References and Acknowledgements

- Shapiro & Stockman, Chapter 4
- Duda, Hart, Stork, Chapters 1, 2.1
- Hastie, Tibshirani, 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