Workshop 1

COMP90051 Machine Learning

Semester 2, 2018

Plan

1. Icebreaker
2. COMP90049 Knowledge & Technologies revision
3. Getting acquainted with Jupyter Notebook

COMP90049 Revision

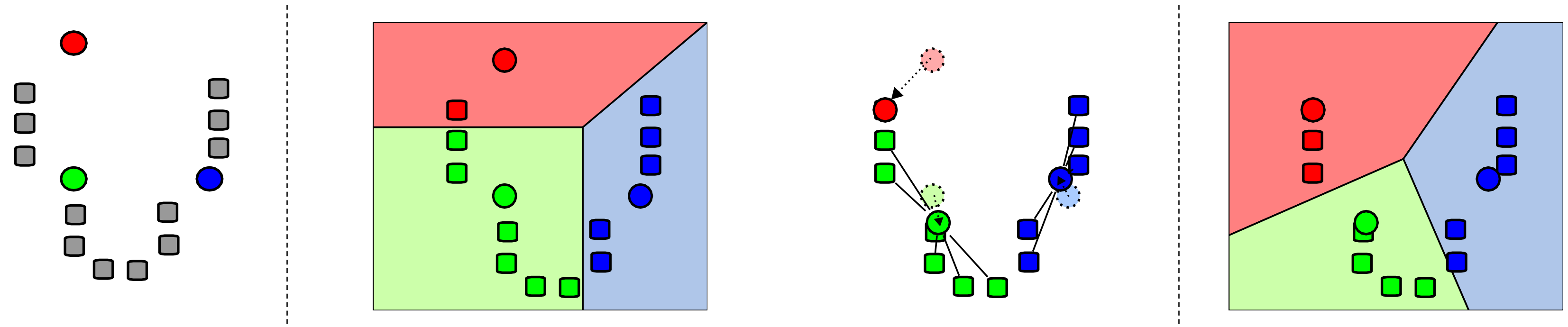
Assumed knowledge for this course

Key concepts

- **un/supervised learning**
- **probability theory**, entropy
- association rule mining
- **k-means clustering**
- **naive Bayes**
- instance-based learning
- feature selection (mutual information)
- **decision stump/tree induction** (0R, 1R, ID5)
- **basic sampling** (hold-out, cross-validation)
- **evaluation** (precision/recall/F, ROC)
- seen: SVMs, bit of Bayes nets

k-means clustering

- Set-up: a set of n data points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$
- Want to partition X into k clusters (minimal within-cluster variance)
- Often use k-means algorithm (has no guarantees)



1) Initialize k "means"

2) Assign points to nearest "mean"

3) Update "means" to cluster centroids

4) Repeat until convergence!

Image source: Wikimedia Commons

Naive Bayes (NB) classifiers

- Want to classify a data point $\boldsymbol{x} = \left(x_1, \dots, x_p\right)$ into one of k classes $\{c_1, \dots, c_k\}$
- NB models conditional probability:

$$P\left(c_j \middle| \boldsymbol{x}\right) \propto P\left(c_j\right) \prod_{i=1}^{p} P\left(x_i \middle| c_j\right)$$

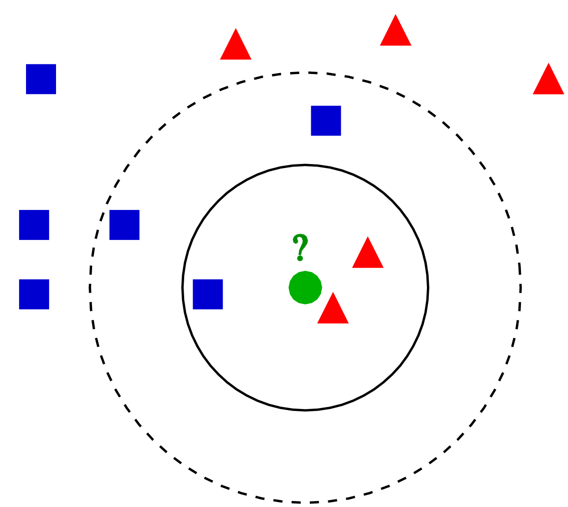
- Relies on Bayes' rule; "naive" conditional independence assumption; assumptions for distributions
- Turn into a classifier: select class c_j with max probability

$$\hat{y} \in \operatorname*{argmax}_j P\left(c_j\right) \prod_{i=1}^{p} P\left(x_i \middle| c_j\right)$$

Nearest Neighbour (NN) classifiers

- Set-up: have a training set (points + class labels)
- k-NN: classify an unseen point according to the *majority class* of its k nearest neighbours in the training set
- Need a distance metric
- E.g. of instance-based learning

assumed Euclidean distance matrisc in 2D space



Predicted class for unseen (green) point depends on k.

for $K=3$, $\longrightarrow$ red triangle

for $K=5$, $\longrightarrow$ blue square.

Image source: Wikimedia Commons

Similarity/distance metrics

- Euclidean distance

$$\|\boldsymbol{x}-\boldsymbol{y}\| = \sqrt{(\boldsymbol{x}-\boldsymbol{y})\cdot(\boldsymbol{x}-\boldsymbol{y})} = \sqrt{\sum_i (x_i - y_i)^2}$$

- Cosine similarity

$$\mathrm{sim}(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{\|\boldsymbol{x}\|\,\|\boldsymbol{y}\|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\,\sqrt{\sum_i y_i^2}}$$

discrete and non ordered
i.e. gender

Decision Trees

- Also works quite well, when your features are categorical

- Classifies an instance by applying a sequence of decision rules
  - Easy to interpret → so good for industry.
- The leaves of the tree represent predictions
- Often construct top-down: choose an attribute at each step that best splits the instances

- no. of ways to construct DT.

TITANIC DATA SET

passengers on the Titanic

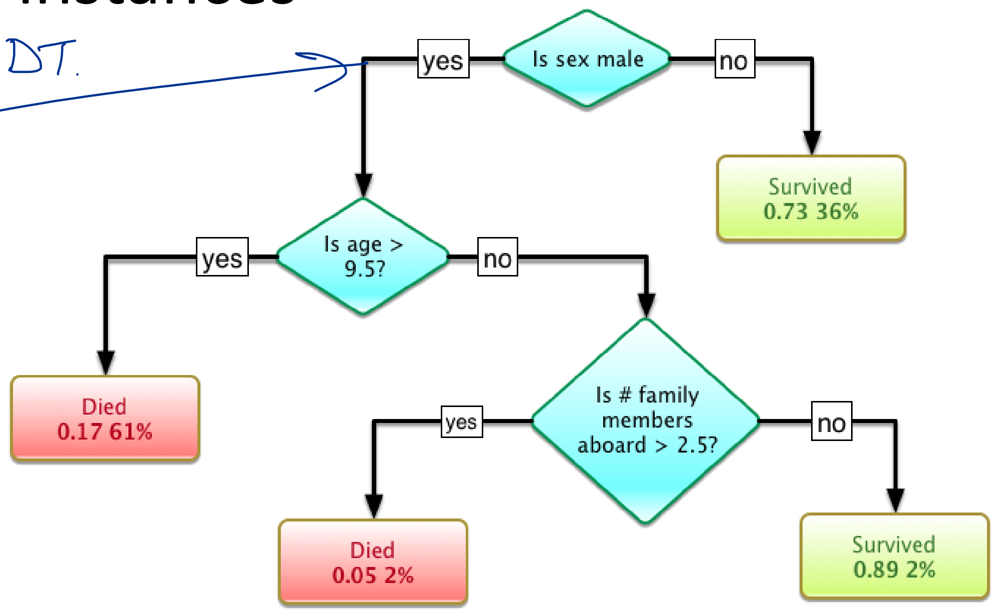


Image source: Wikimedia Commons

Iterative Dichotomiser 3 (ID3)

- Greedy algorithm for constructing a decision tree
- Recursive divide-and-conquer approach
- Relies on metric to determine splitting rule

FUNCTION ID3 (Root)

IF all instances at root have same class

THEN stop

ELSE Select a new attribute to use in partitioning root node instances

Create a branch for each attribute value and partition up root node instances according to each value

Call ID3(LEAF_i) for each leaf node LEAF_i

ID3 splitting metrics

- **Entropy** for node R is

$$H(R) = -\sum_{c \in C} p(c) \log_2 p(c)$$

where C is the set of classes and $p(c)$ is the rel. freq. of class c in R.

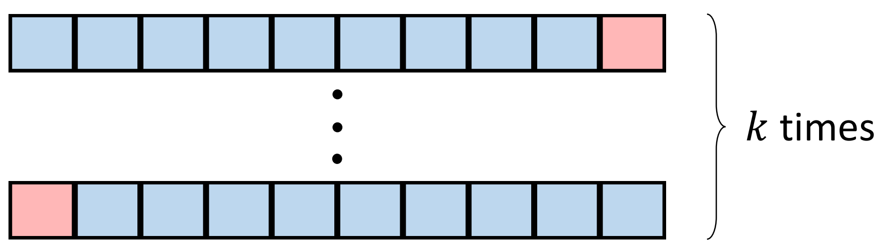
- **Information gain** after splitting node R on attribute A

$$IG(R, A) = H(R) - \sum_{r \in R'} p(r) H(r)$$

where R' are the subsets created by splitting R on A and $p(r)$ is the relative size of r (vs R).

Sampling for evaluation

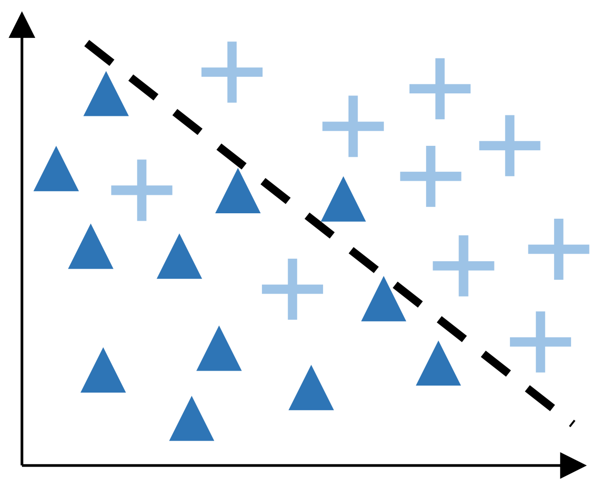
- **Holdout:** randomly partition data into a training set and test set. training set ↔ model fitting. test set ↔ evaluation.
- **Random subsampling:** repeat holdout multiple times (fixing the train:test ratio) and take the average evaluation result. • You may have class imbalance.
  - Random split, may not be representative of underlying data distribution
- **Stratified random sub-sampling:** same as above, but preserves the class distribution in the training/test sets
  - useful for imbalanced classes
- ***k*-fold cross-validation:** partition data into k folds, use $k-1$ for training and 1 as test, repeat for all k permutations



Classifier evaluation

- Compare model *predictions* with *ground truth* on a held-out test set
- **Confusion matrix:** counts correct/incorrect predictions
- **Evaluation measure:** summarises error as a scalar (often between 0 and 1)

- Evaluation measures combine these 4 numbers in some way to produce a metric



		Truth	
	Binary class	Positive	Negative
Prediction	Positive	True positive	False positive
	Negative	False negative	True negative

Classifier evaluation

- Many evaluation measures to consider—depending on types of errors that are important

not good for imbalanced application problems

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Error rate} = \frac{FP + FN}{TP + TN + FP + FN}$$

$$F_\beta = (1 + \beta^2)\frac{\text{Pr} \cdot \text{Re}}{\text{Re} + \beta^2 \cdot \text{Pr}}$$

Receiver operating characteristic (ROC) curves

- Plots true positive rate (TPR) vs false positive rate (FPR) as the *classifier threshold varies*
- E.g. for a NB classifier, could compute for thresholds $[0.0, 0.1, 0.2, \ldots, 1.0]$
- $FPR = \frac{FP}{FP+TN} = 1 - \text{Specificity}; TPR = \frac{TP}{TP+FN} = \text{Recall}$
- Another commonly used measure, Area Under Curve (AUC) is derived from the area under the ROC curve.

Receiver operating characteristic (ROC) curves

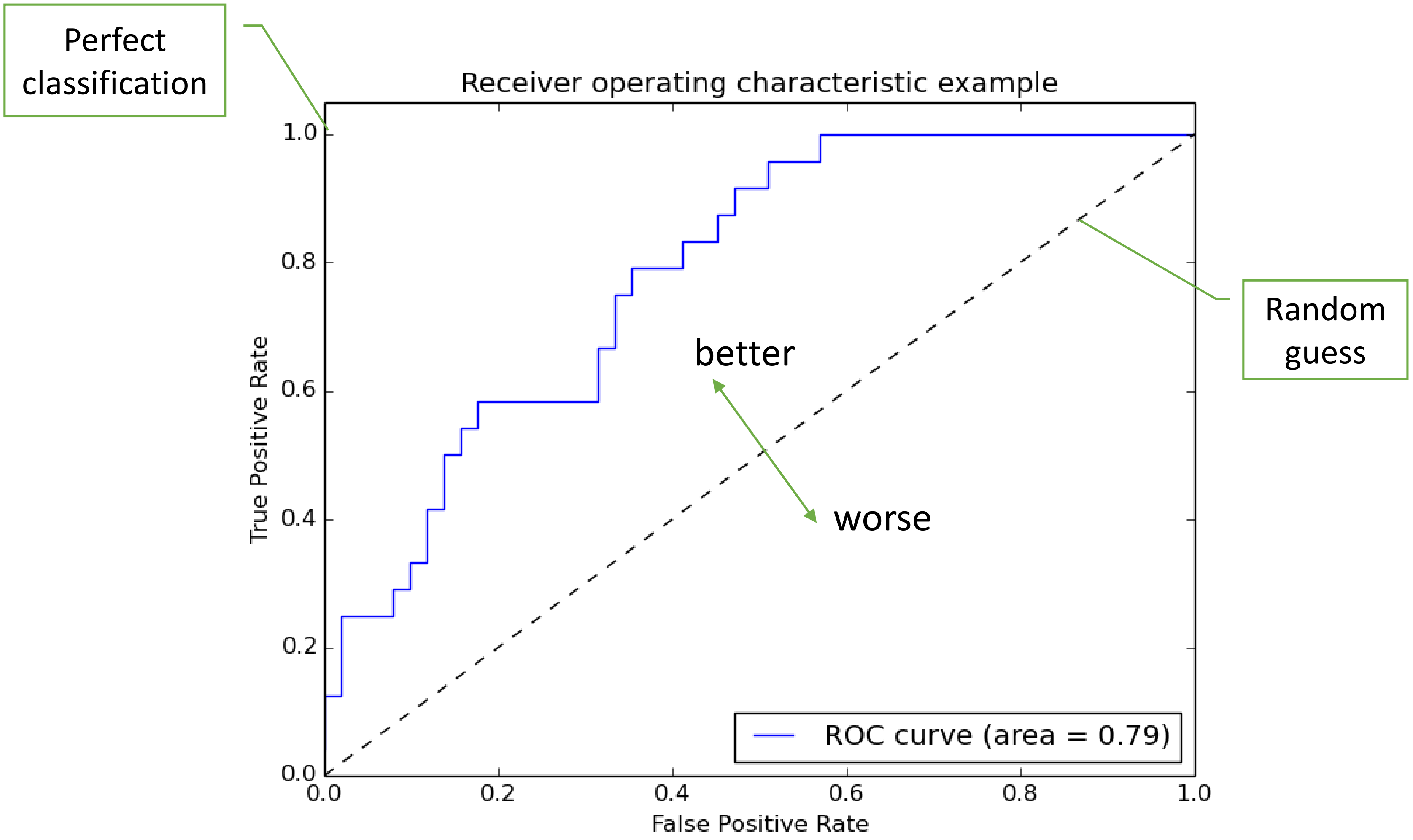


Image source: sklearn documentation

Jupyter Notebook

Running Jupyter Notebook

1. Download worksheet01.ipynb from the LMS
2. Move the downloaded file to a working directory `%WORKDIR%`
3. Start → *Anaconda3 (64-bit)* → *Anaconda Prompt*
4. Type the following command at the prompt:

```
jupyter notebook --notebook-dir=%WORKDIR%
```

5. The Jupyter UI should open in a web browser.
6. Click on worksheet1.ipynb to get started.