Weekly Oxford Worldwide

DEPARTMENT FOR CONTINUING EDUCATION

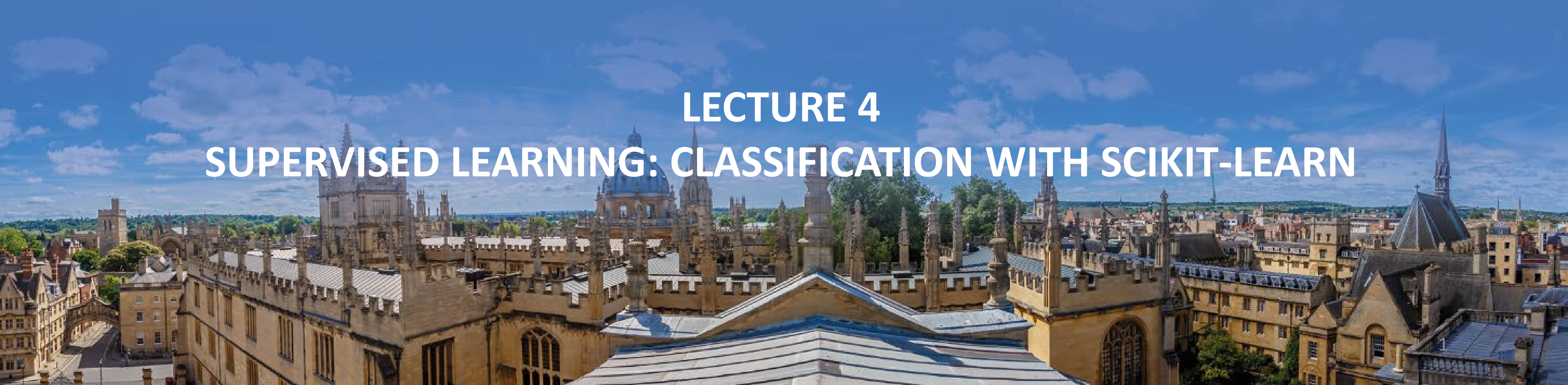
UNIVERSITY OF OXFORD

PYTHON PROGRAMMING FOR DATA SCIENCE – PART 2

MASSIMILIANO IZZO & NICHOLAS DAY

LECTURE 4

SUPERVISED LEARNING: CLASSIFICATION WITH SCIKIT-LEARN



This week

- Classifiers
- Performance metrics for Classification
- Classification using MNIST dataset

Classification

- Classification tasks can be:
  - binary (two classes, generally coded a 0 and 1)
  - multi-class
  - multi-label

- Performance metrics are tricker for classification. Accuracy would be the most intuitive obvious one to measure

$$accuracy = \frac{\#\, correctly\ predicted\ records}{\#\, total\ records}$$

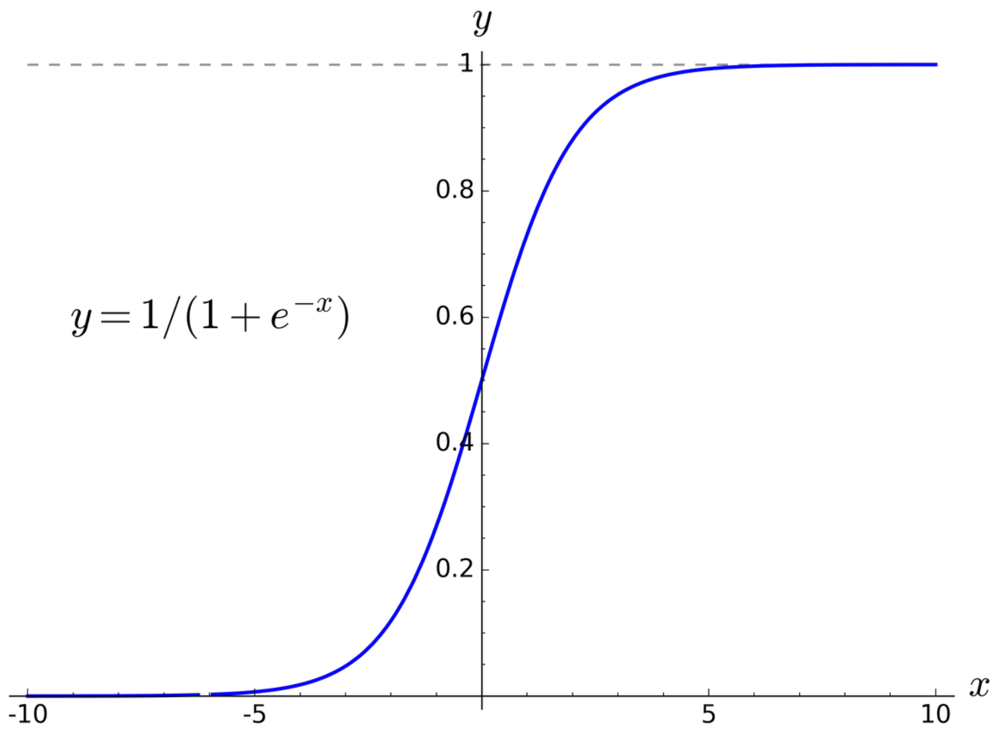
Logistic Regression

Binary classification (classes 0 and 1)

Estimates the probability that a sample belongs to a certain class by training a (linear) regressor that will return scores in the $(-\infty, +\infty)$interval

then passing the output of the regressor to a **logistic (sigmoid) function**

The output will be a value between 0 and 1. If the output is > 0.5 assign to class 1 otherwise assign to class 0



$$\hat{p}(\boldsymbol{\theta}) = \hat{p}(\boldsymbol{w}, b) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-(\boldsymbol{x}^T\boldsymbol{w}+b)}}$$

z and $\hat{p}$ are the regressor score and the predicted probability for the positive class ($y = 1$)

Logistic Regression: the log loss cost function

We can train a Logistic Regression classifier using Gradient Descent, but we cannot use the MSE as a cost function to minimise

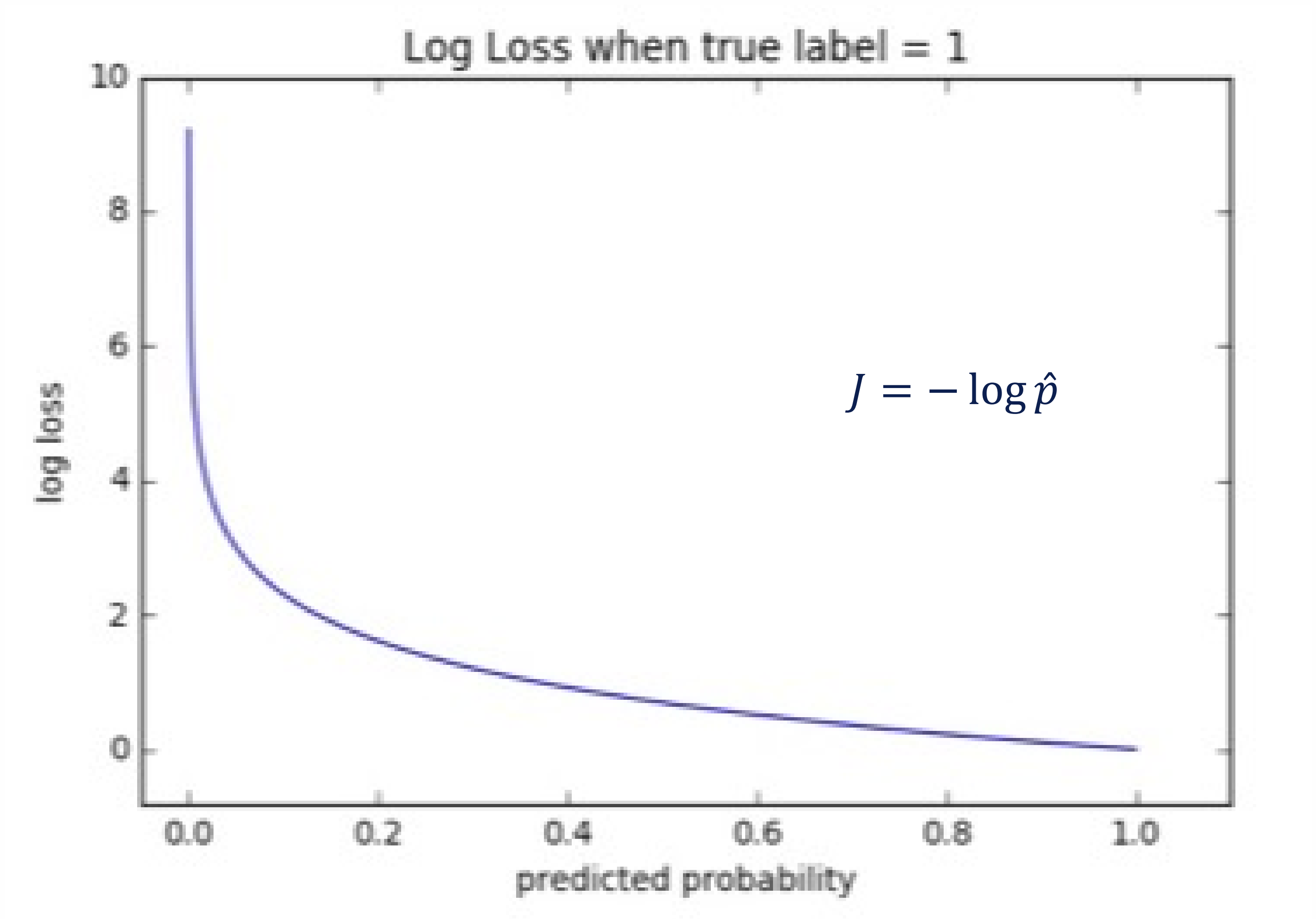
$$c(\boldsymbol{\theta}) = \begin{cases} -\log \hat{p} & if\ y = 1 \\ -\log(1-\hat{p}) & if\ y = 0 \end{cases}$$

where $y\ \in \{0, 1\}$ is the actual label value, and $\hat{p}$ is the predicted probability of positive instances ($y = 1$)

The overall cost function over a training set with m samples is:

$$logloss = J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i)$$

Log-loss cost function for class 1



$$J = -\log \hat{p}$$

https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

Softmax Regression

Extension of the logistic regression to multi-class scenarios.

The regressor will return a regression score z_k for each class k

Rather than using the sigmoid we use the softmax function:

- $\hat{p}_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
- $\sum_i^K \hat{p}_i = 1$

K = number of classes
k = current class
z_k = regressor score for class k
$\hat{p}_k$ = probability for class k

The class with highest probability will be the predicted class of the softmax regressor

Softmax Regression: the cross-entropy cost function

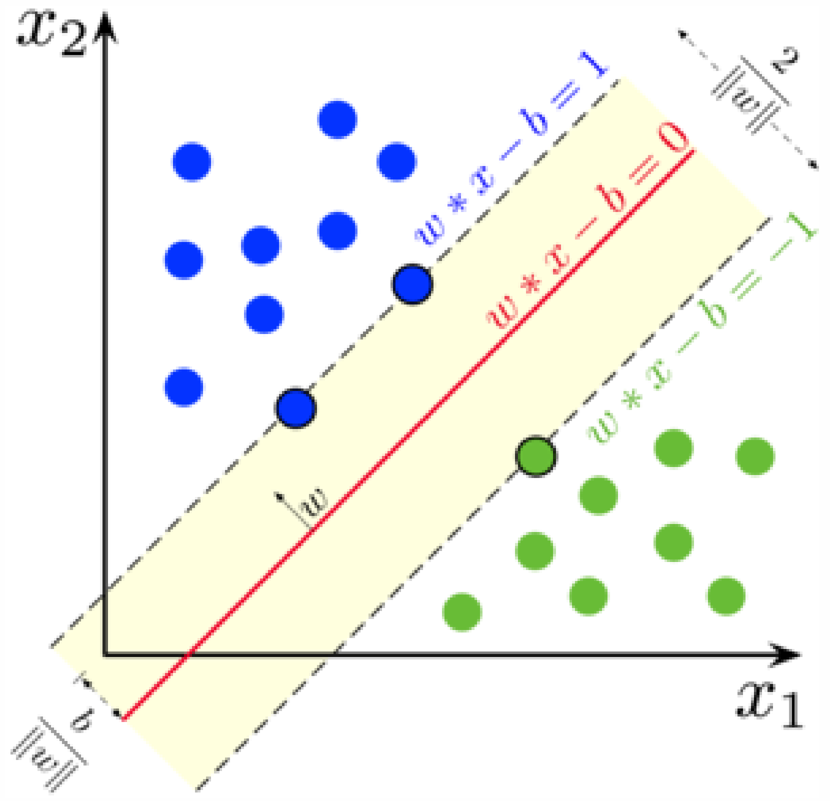
The cost function used for softmax regression is a multi-class extension of the log-loss function, called cross-entropy

$$\text{crossentropy} = \mathrm{J}(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k}^{K} y_k \log \hat{p}_k$$

m = number of records/samples in training set
K = number of classes
k = current class
z_k= regressor score for class k
$\hat{p}_k$= probability for class k (this is a function of $\boldsymbol{\theta}$)

Support Vector Machines

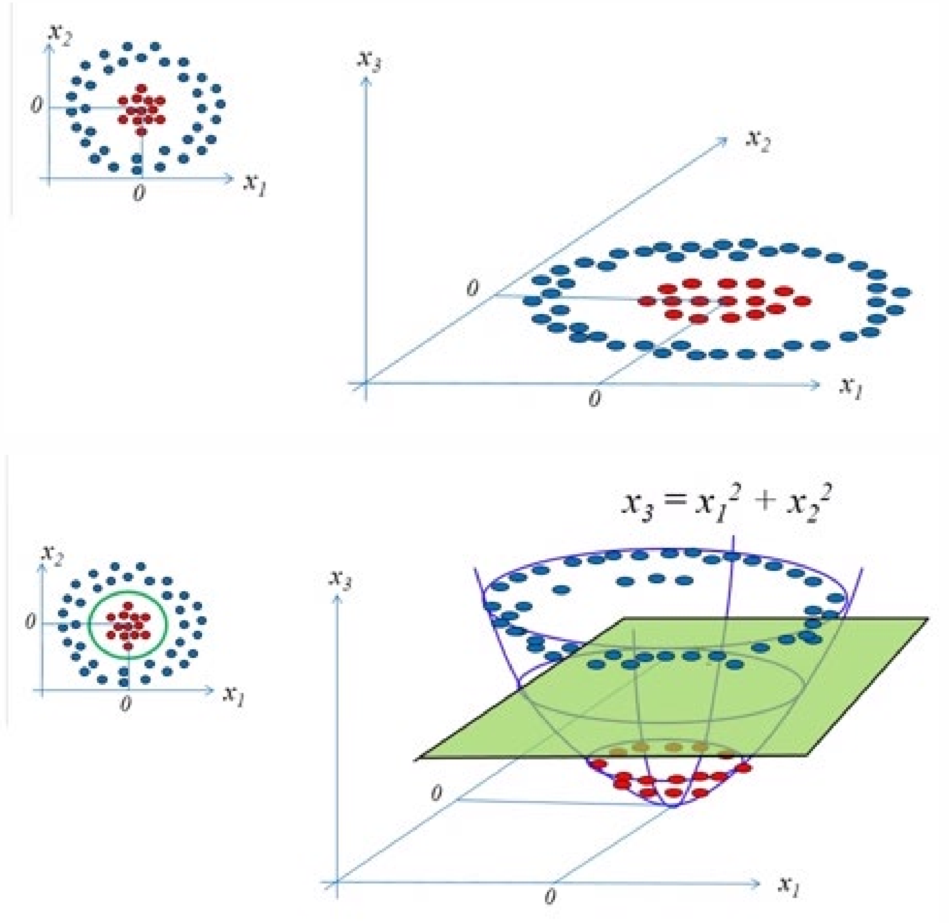
A support vector machine (SVM) constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier



SVMs: the Kernel Trick

To solve a nonlinear problem with SVM:
1.We transform the training data onto a **higher dimensional feature space** via a mapping function φ.
2.We train a linear SVM model to classify the data in this new feature space.
3.Then, we can use the same mapping function φ to transform unseen data to classify it using the linear SVM model.

The **kernel trick** avoids the explicit mapping that is needed to get linear learning algorithms to learn a nonlinear function or decision boundary.



https://www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_Support_Vector_Machines_SVM_2.php

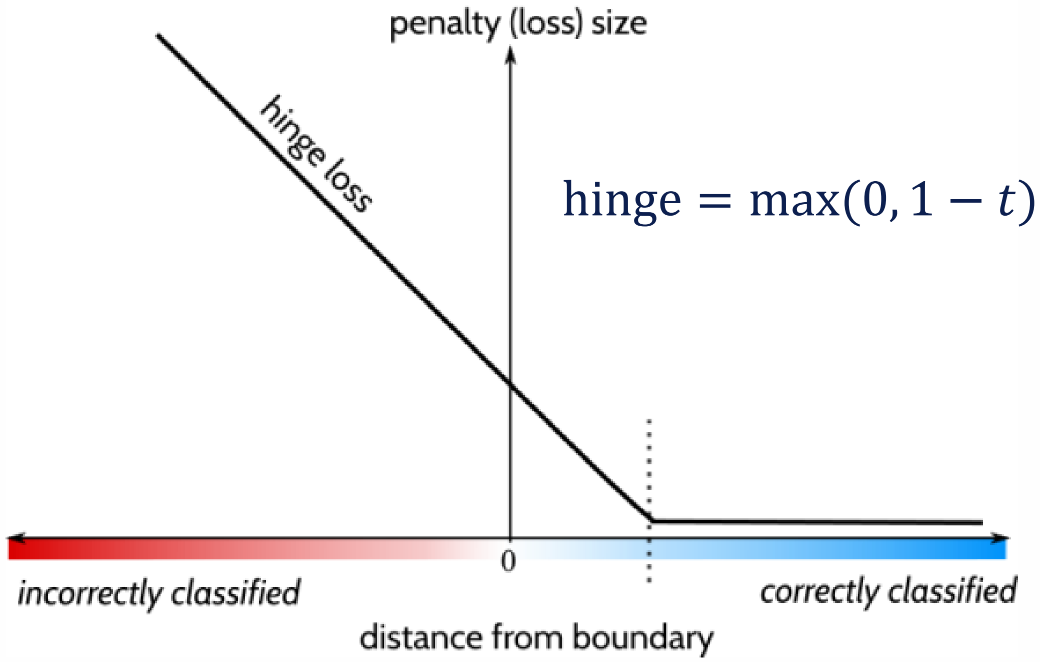
Online SVMs with the hinge loss

$$J(\theta) = J(\boldsymbol{w}, b) = \underbrace{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}}_{\text{regularization term}} + \underbrace{C\sum_{1=1}^{m}\max(0, 1 - t_i(\boldsymbol{w}^T\boldsymbol{x}_i + b))}_{\text{hinge loss term}}$$

Traditional SVM are trained offline (batch-training)

However, we can train online SVMs using gradient descent, just like we train logistic or softmax regression classifiers

Rather than using the log loss we use the **hinge function** as our cost function



$$\text{hinge} = \max(0, 1 - t)$$

https://www.analyticsvidhya.com/blog/2021/05/top-15-questions-to-test-your-data-science-skills-on-svm/

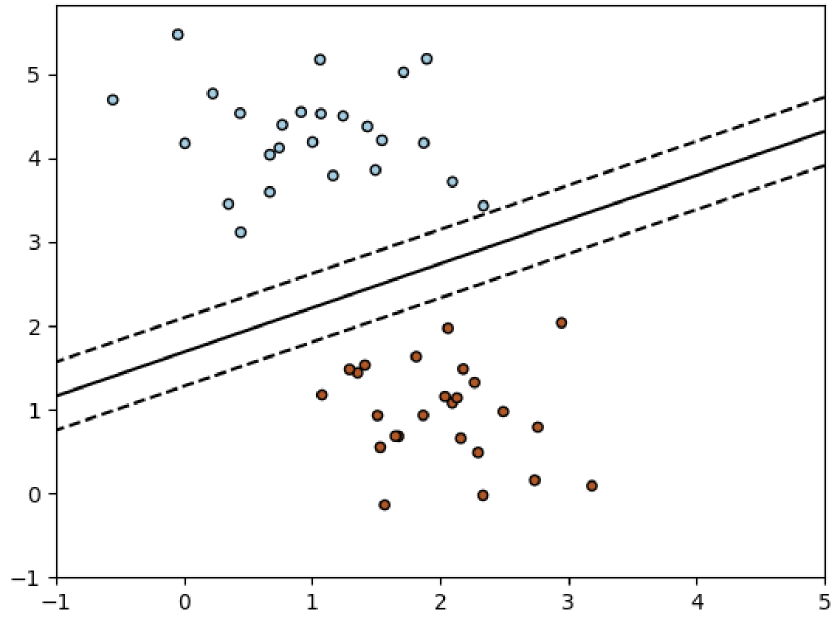
DEPARTMENT FOR CONTINUING EDUCATION

UNIVERSITY OF OXFORD

Stochastic Gradient Descent Classification

https://scikit-learn.org/stable/modules/sgd.html

- The class `SGDClassifier` implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties for classification.
- The default function scikit-learn is the Linear SVM decision function



https://scikit-learn.org/stable/auto_examples/linear_model/plot_sgd_separating_hyperplane.html

Nearest Neighbours Classification

- a type of *instance-based learning* or *non-generalizing learning*
- Classification is computed from a simple majority vote of the nearest neighbours of each point
- KNeighborsClassifier
- RadiusNeighborsClassifier

Naive Bayes

- Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of feature records given the value of the class variable.

$$\hat{y} = arg\,\max_{y} P(y) \prod_{i=1}^{N} P(x_i|y)$$

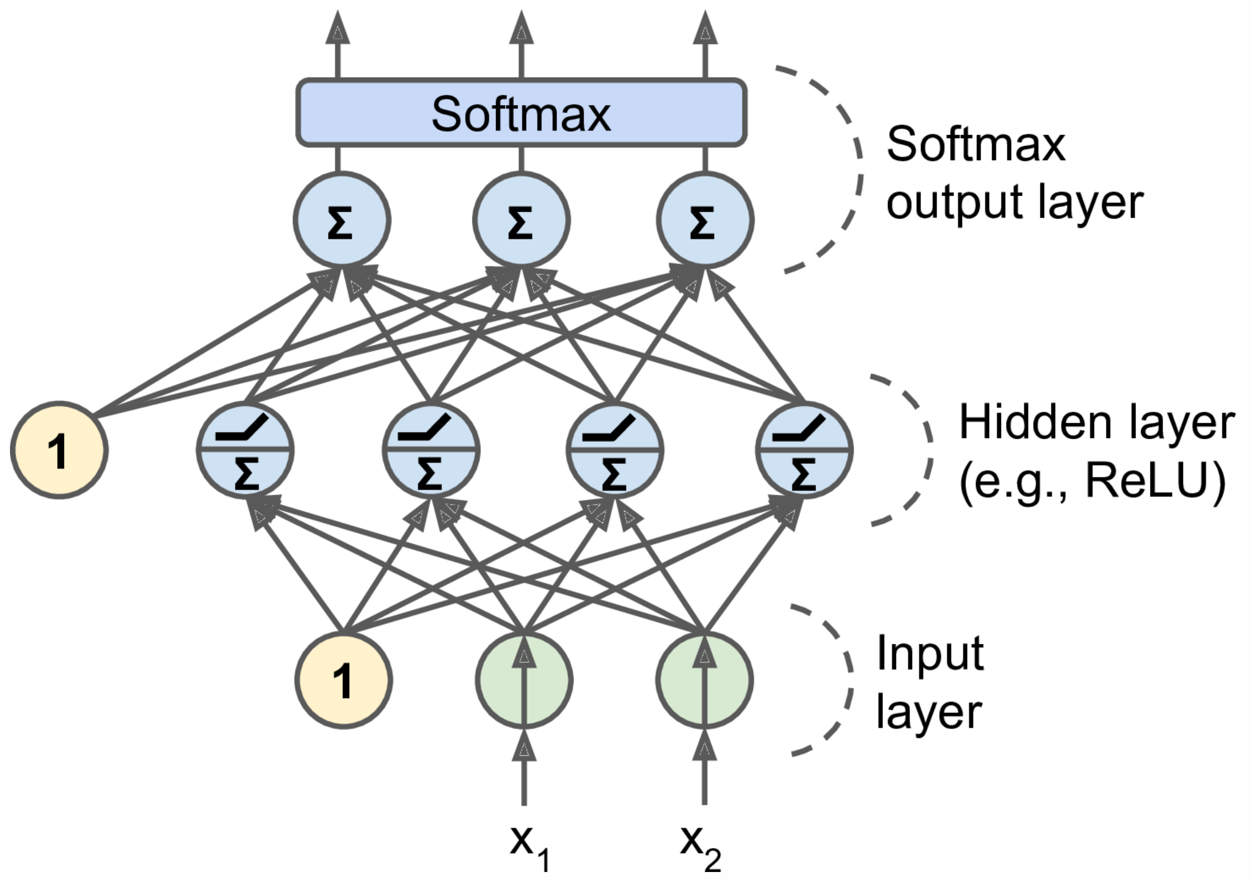
- The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$

Decision Trees

- Decision Trees (DTs) are a non-parametric supervised learning method used for classification (and regression). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- Advantages:
  - Simple to understand and to interpret. Trees can be visualised.
  - Requires little data preparation (no normalization)
  - Able to handle both numerical and categorical data.
- Disadvantages:
  - Overfitting
  - Learning the optimal DT is an NP-complete problem => heuristics

Neural Networks: Multi-layer Perceptron

- To be seen…



Ensemble methods

- The goal of **ensemble methods** is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator
  - Averaging methods: Random Forests
  - Boosting methods: Ada Boost, Gradient Tree Boosting
  - Voting Classifier

DEPARTMENT FOR CONTINUING EDUCATION

UNIVERSITY OF OXFORD

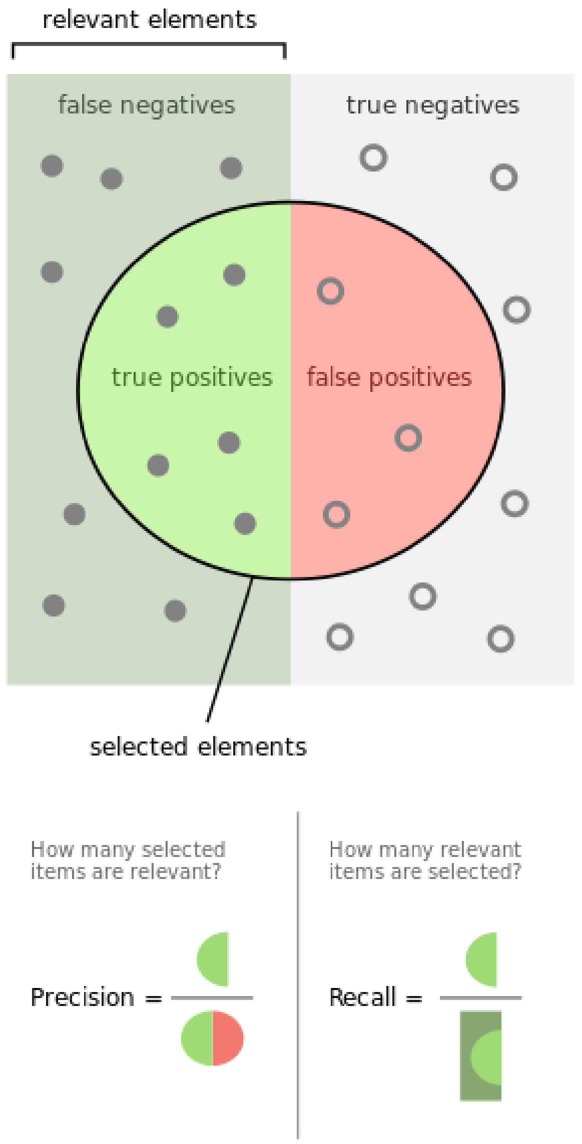
Performance Metrics: Confusion Matrix

$$accuracy = \frac{correct\ predictions}{total\ predictions} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

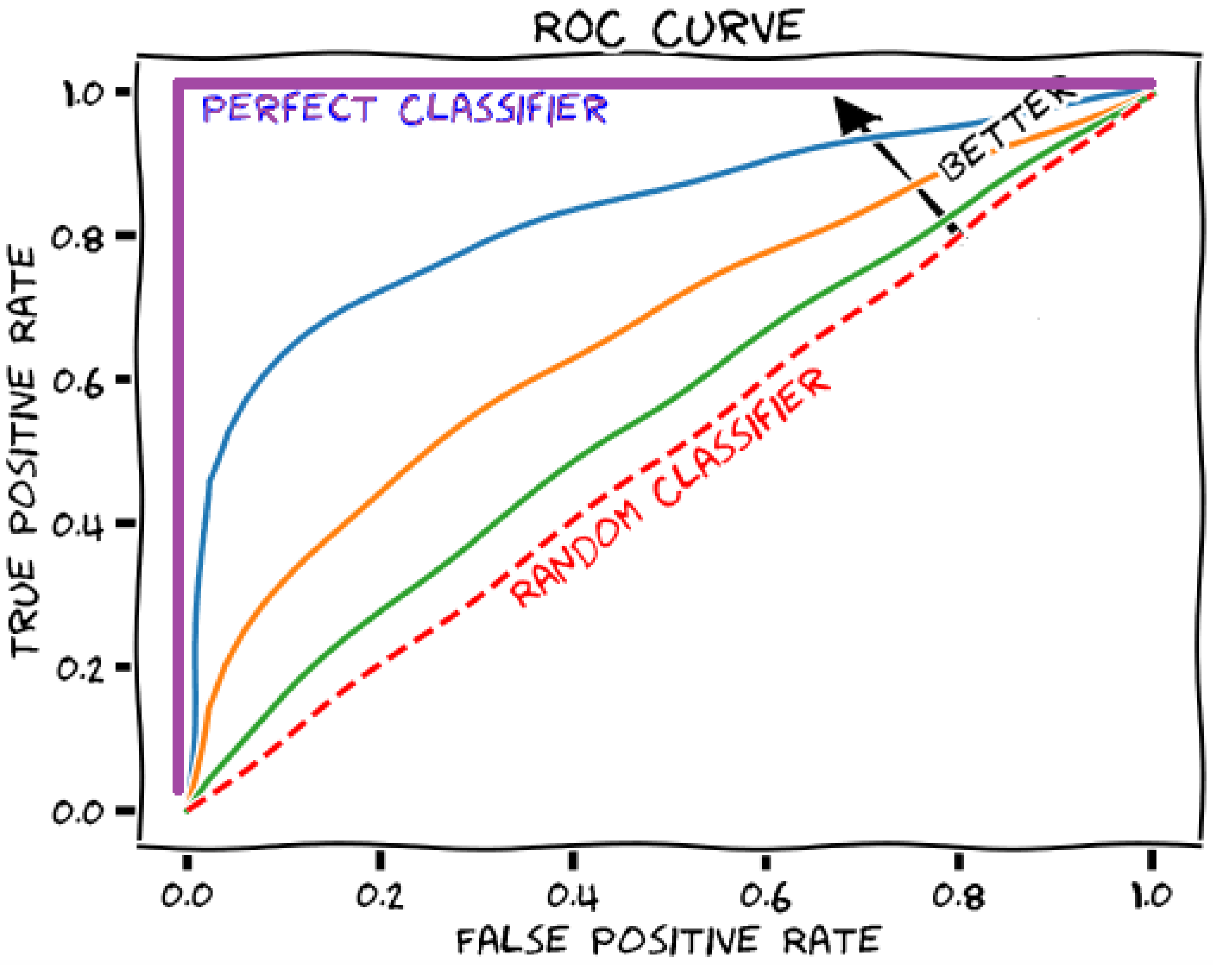
$$F1 = 2\ \frac{precision\ \times\ recall}{precision + recall}$$



Performance Metrics: Area under the ROC Curve

$$TPR = recall = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$



https://medium.datadriveninvestor.com/understanding-auc-roc-clearly-explained-74c53d292a02