UNIVERSITÀ
DEGLI STUDI
DI MILANO

LA STATALE

Master Degree in Computer Science

Information Retrieval

Supervised Text Classification

Prof. Alfio Ferrara

Department of Computer Science, Università degli Studi di Milano
Room 7012 via Celoria 18, 20133 Milano, Italia alfio.ferrara@unimi.it

sed noli modo

Introduction to text classification

Text classification is the problem of **associating texts** (i.e., documents) **to classes** (or labels denoting classes).

Classes may be

- A partition of the document space (i.e., disjoint classes)
- Overlapping (i.e., a document may be classified in more than one class)

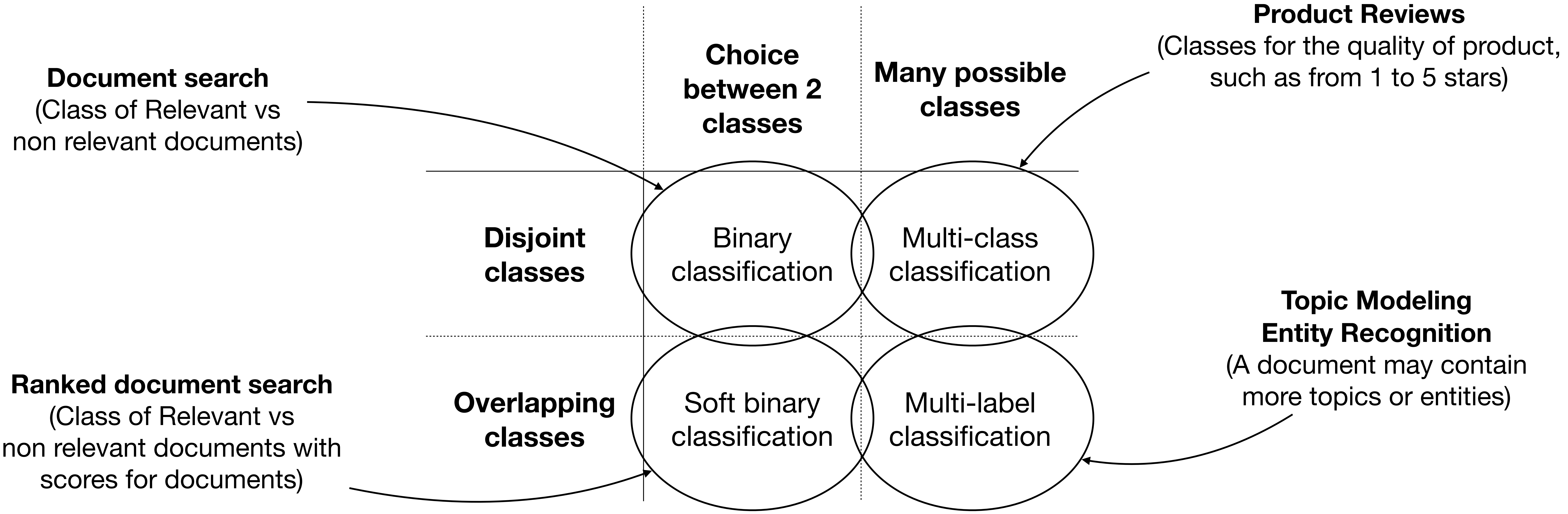
Methods may be

- Based on a training set of pre-classified documents: **supervised**
- Based on documents only: **unsupervised** (today's topic)

	Choice between 2 classes	**Many possible classes**
Disjoint classes	Binary classification	Multi-class classification
Overlapping classes	Soft binary classification	Multi-label classification

Introduction to text classification

Many problems may be modeled as a classification problem. Examples:



	Choice between 2 classes	Many possible classes
Disjoint classes	Binary classification	Multi-class classification
Overlapping classes	Soft binary classification	Multi-label classification

Document search
(Class of Relevant vs
non relevant documents)

Product Reviews
(Classes for the quality of product,
such as from 1 to 5 stars)

Ranked document search
(Class of Relevant vs
non relevant documents with
scores for documents)

Topic Modeling
Entity Recognition
(A document may contain
more topics or entities)

Supervised classification

The problem of supervised text classification is to **partition data into pre-defined groups (i.e., classes)** that are identified by their **labels**.

We might have k different classes, and there is no **inherent ordering among these classes**.

A **training data set** is provided with **examples of documents belonging to the classes** (labeled data set, sometimes called document annotations).

The goal is, for an **unlabeled test data set**, to **classify unlabeled documents into** one (or more in case of multi-label classification) of these **pre-defined classes**.

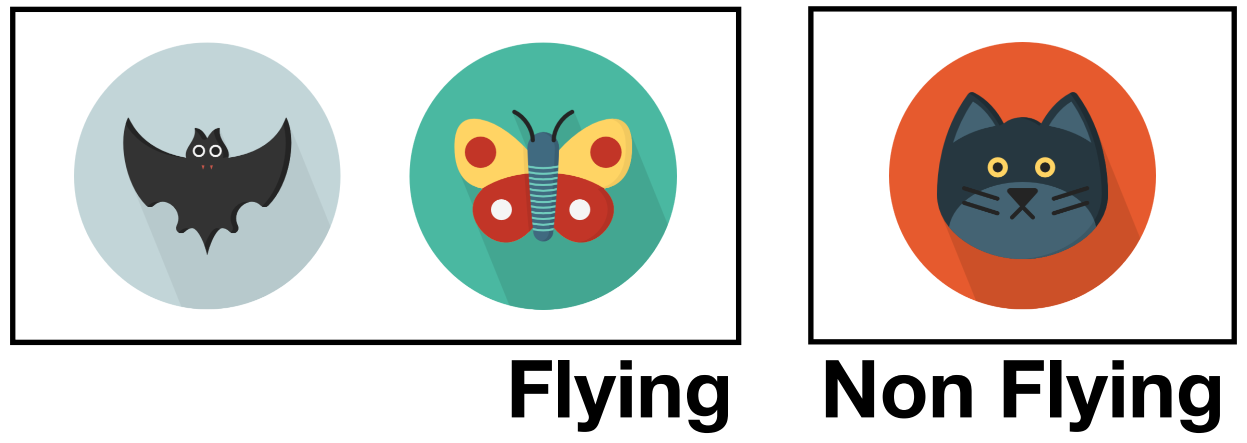
Supervised vs unsupervised classification

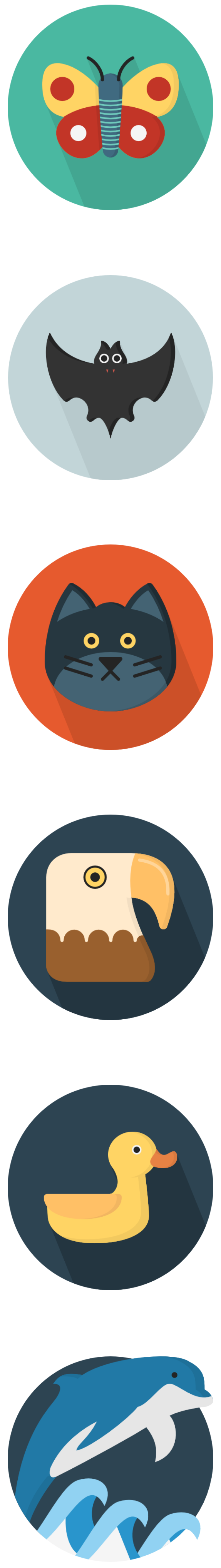
In general, there are many **different possible classification schemes** for the same data, according to different criteria.

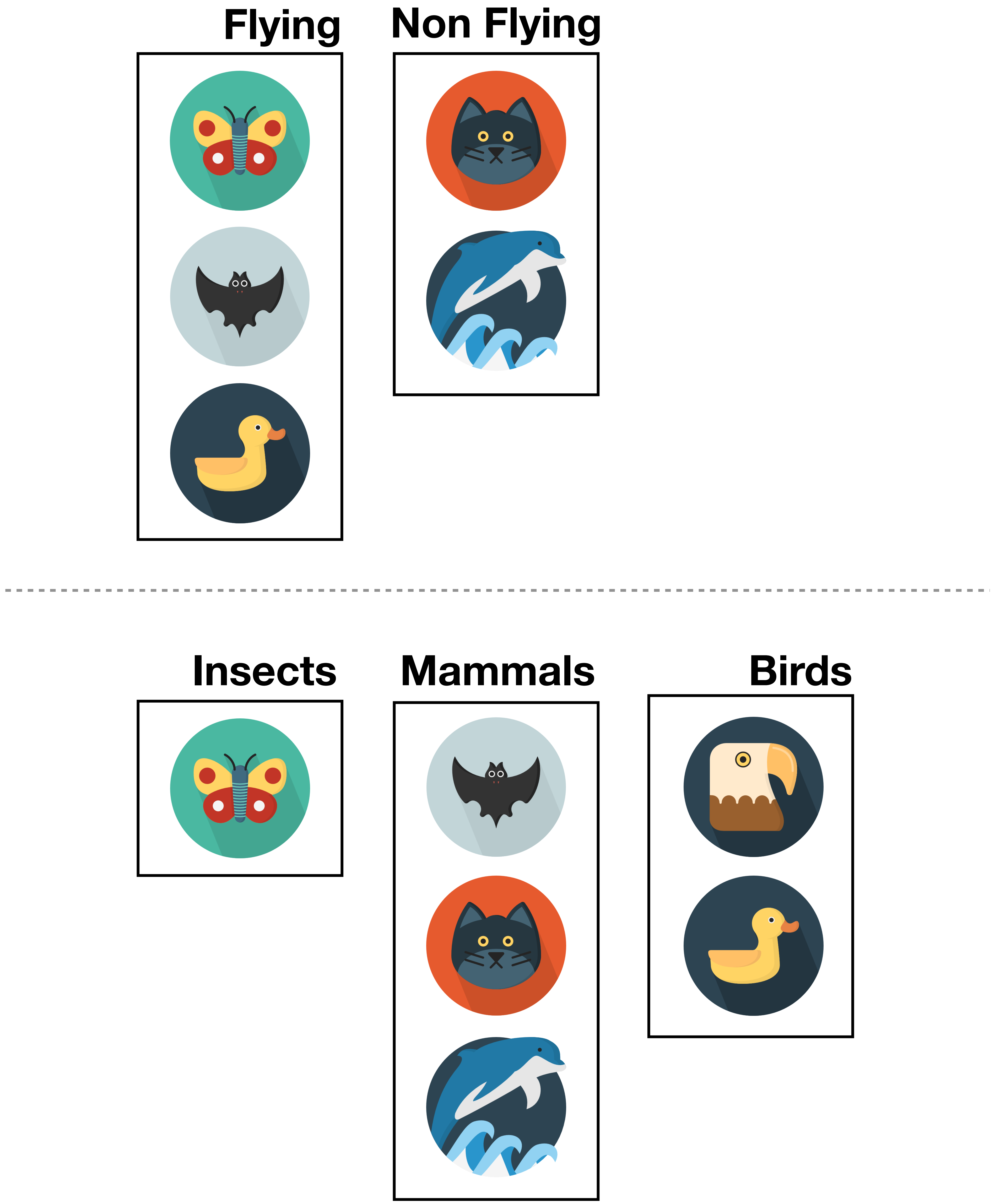
In **unsupervised classification**, we **do not have any information of the criteria** to use for classifying, we can just exploit data features.

In **supervised classification** instead, the training set gives information not only on which **classes are expected** but also on which **data features may be correlated with the target classes**.

Example of training set







Supervised classification

Given a data instance X, supervised classification may be seen as the problem of learning a function $f(\cdot)$ such as:

$$\hat{y} = f(X)$$

In **multi-class classification**, the range of $f(\cdot)$ is a **set of discrete class labels** ({0, 1} for binary classification).

In **multi-label classification** instead, $f(\cdot)$ is not a function but a **multivalued function** because each data item can be mapped on more than a single class. However, **multi-label classification can be reduced to multiple applications of binary classification** with simple meta-algorithms

The function $f(\cdot)$ can also **map to a numerical value**. This problem is referred to as **regression modeling**, and it no longer partitions the data into discrete groups like classification.

Overview of the main categories of supervised classification algorithms

Aggarwal, C. C. (2018). Machine learning for text. Cham: Springer International Publishing.

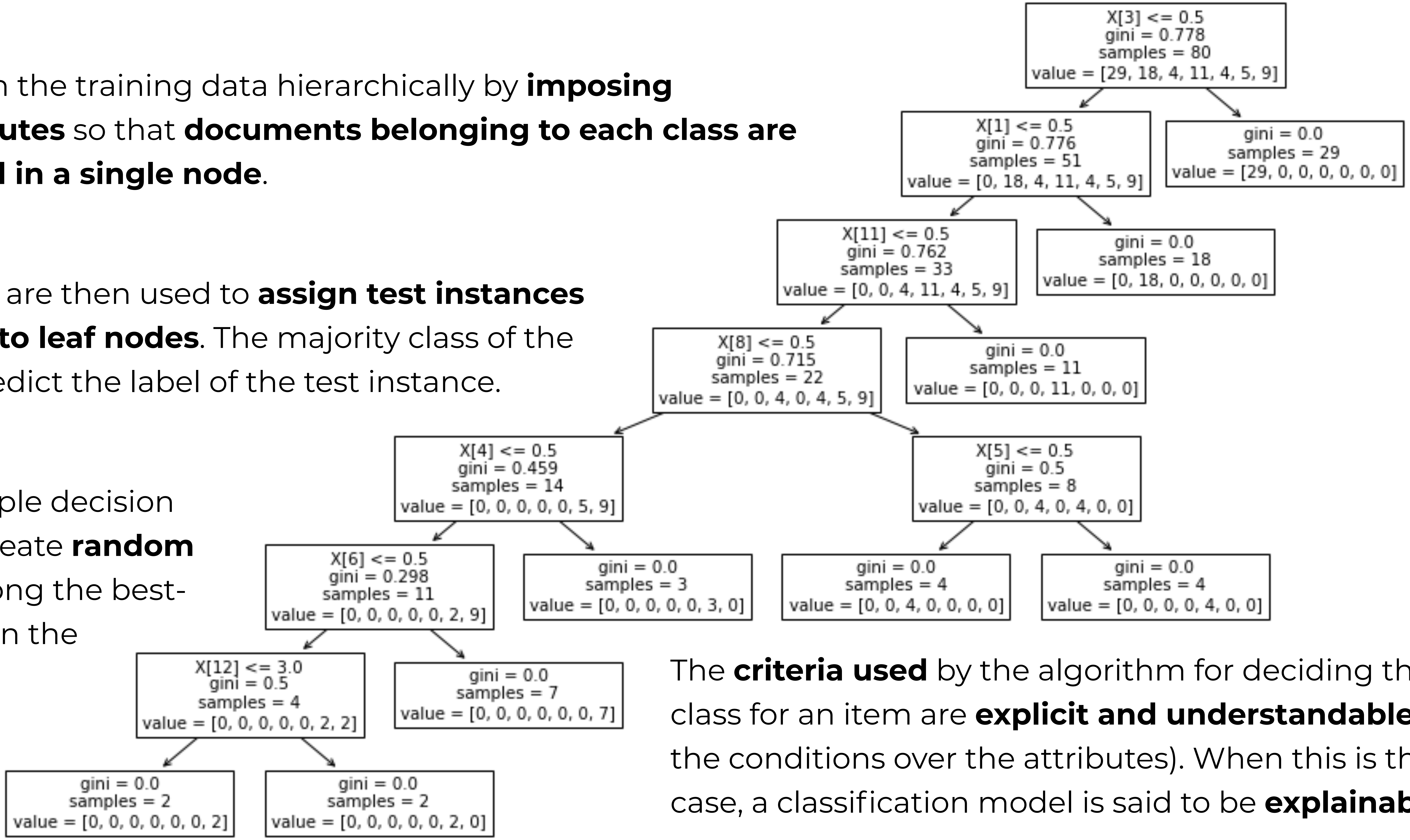
Decision Trees

Loh, W. Y. (2014). Fifty years of classification and regression trees. International Statistical Review, 82(3), 329-348.

Decision trees partition the training data hierarchically by **imposing conditions over attributes** so that **documents belonging to each class are predominantly placed in a single node**.

These **split conditions** are then used to **assign test instances with unknown labels to leaf nodes**. The majority class of the leaf node is used to predict the label of the test instance.

Combinations of multiple decision trees can be used to create **random forests**, which are among the best-performing classifiers in the literature.


X[3] <= 0.5
gini = 0.778
samples = 80
value = [29, 18, 4, 11, 4, 5, 9]
X[1] <= 0.5
gini = 0.776
samples = 51
value = [0, 18, 4, 11, 4, 5, 9]
gini = 0.0
samples = 29
value = [29, 0, 0, 0, 0, 0, 0]
X[11] <= 0.5
gini = 0.762
samples = 33
value = [0, 0, 4, 11, 4, 5, 9]
gini = 0.0
samples = 18
value = [0, 18, 0, 0, 0, 0, 0]
X[8] <= 0.5
gini = 0.715
samples = 22
value = [0, 0, 4, 0, 4, 5, 9]
gini = 0.0
samples = 11
value = [0, 0, 0, 11, 0, 0, 0]
X[4] <= 0.5
gini = 0.459
samples = 14
value = [0, 0, 0, 0, 0, 5, 9]
X[5] <= 0.5
gini = 0.5
samples = 8
value = [0, 0, 4, 0, 4, 0, 0]
X[6] <= 0.5
gini = 0.298
samples = 11
value = [0, 0, 0, 0, 0, 2, 9]
gini = 0.0
samples = 3
value = [0, 0, 0, 0, 0, 3, 0]
gini = 0.0
samples = 4
value = [0, 0, 4, 0, 0, 0, 0]
gini = 0.0
samples = 4
value = [0, 0, 0, 0, 4, 0, 0]
X[12] <= 3.0
gini = 0.5
samples = 4
value = [0, 0, 0, 0, 0, 2, 2]
gini = 0.0
samples = 7
value = [0, 0, 0, 0, 0, 0, 7]
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 0, 0, 2]
gini = 0.0
samples = 2
value = [0, 0, 0, 0, 0, 2, 0]

The **criteria used** by the algorithm for deciding the class for an item are **explicit and understandable** (i.e., the conditions over the attributes). When this is the case, a classification model is said to be **explainable**.

Rule-Based Classifiers

Nutaro, J., & Ozmen, O. (2020). Quantifying the Uncertainty of Precision Estimates for Rule based Text Classifiers. arXiv preprint arXiv:2005.09198.

Rule-based classifiers relate conditions on subsets of attributes to specific class labels. Thus, the **antecedent of a rule contains a set of conditions**, which typically correspond to the presence of a subset of words in the document. The **consequent of the rule contains a class label.** For a given test instance, the rules whose antecedents match the test instance are discovered. The (possibly conflicting) predictions of the discovered rules are used to predict the labels of test instances.

Naive Bayes Classifier(s)

Ting, S. L., Ip, W. H., & Tsang, A. H. (2011). Is Naive Bayes a good classifier for document classification? International Journal of Software Engineering and Its Applications, 5(3), 37-46..

In naive Bayes classifier the basic idea is that the **data is generated by a mixture of k components**, where k is the number of classes in the data. The words in each class are defined by a specific distribution.

Therefore, the **parameters of each mixture component specific distribution** need to be **estimated in order to maximize the likelihood of these training instances being generated by the component**. These probabilities can then be used to estimate the probability of a test instance belonging to a particular class.

This classifier is referred to as “naive” because it makes some simplifying assumptions about the independence of attribute values in test instances.

POSTERIOR IS PROPORTIONAL TO PRIOR TIMES LIKELIHOOD

$$P(class \mid features) = \frac{P(class)P(features \mid class)}{P(features)} \propto P(class)P(features \mid class)$$

$$P(class) = \frac{size(class)}{size(corpus)}$$

We need to estimate the class given the features (i.e., words)

We can estimate the class prior by observing the class size in the training set

We can estimate the likelihood of features given the class by observing the frequency of words in the training set class-by-class

$$P(features \mid class) = \prod_{i=1}^{n} \frac{count(w_i, class)}{\sum_{j=1}^{n} count(w_j, class)}$$

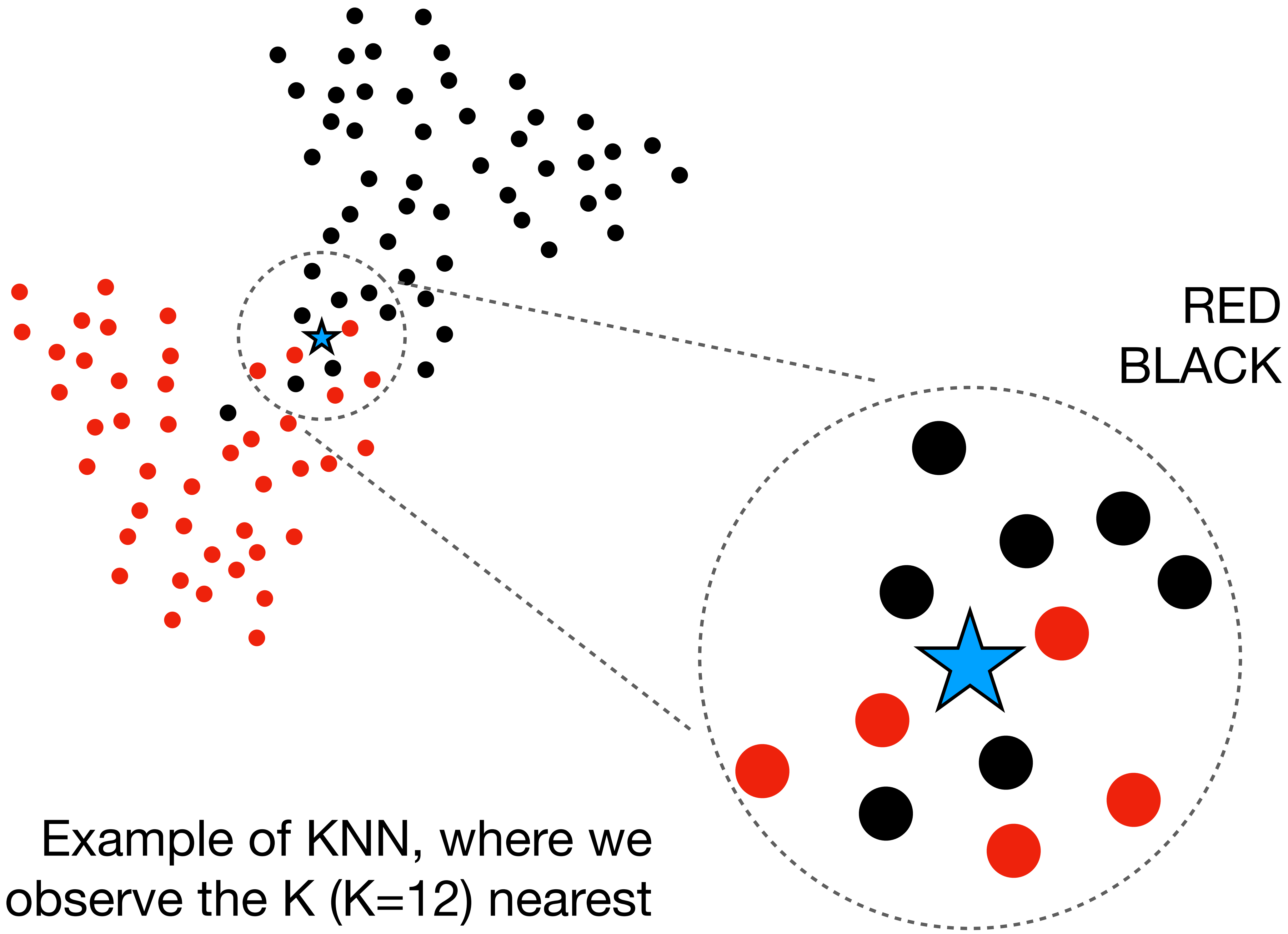
Nearest Neighbor Classifiers

Bhatia, N. (2010). Survey of nearest neighbor techniques. arXiv preprint arXiv:1007.0085.

Nearest neighbor classifiers are also referred to as **instance-based learners**, **lazy learners**, or **memory-based learners**.

The basic idea in a nearest neighbor classifier is to **retrieve the k-nearest training examples to a test instance and report the dominant label of these examples**.

In other words, it works by memorizing training instances, and leaves all the work of classification to the very end (in a lazy way) without doing any training up front.



RED = 5 / 12
BLACK = 7 / 12

Example of KNN, where we observe the K (K=12) nearest training documents to a given item.

Linear Classifiers

The most natural case of linear classification is **regression modeling** in which the **dependent variable is numeric**. The basic idea is to assume that the prediction function is in the following linear form:

N-dimensional vector of coefficients

$$\hat{y} \approx W \cdot X + b$$

Scalar *bias* coefficient

Where we need to learn *W* and *b* in order to minimize the prediction error

$$Minimize \sum_{i=1}^{n} loss(y_i - (W \cdot X_i - b)) + regularizer$$

Quantify the prediction error

This factor has the goal of preventing overfitting for small datasets

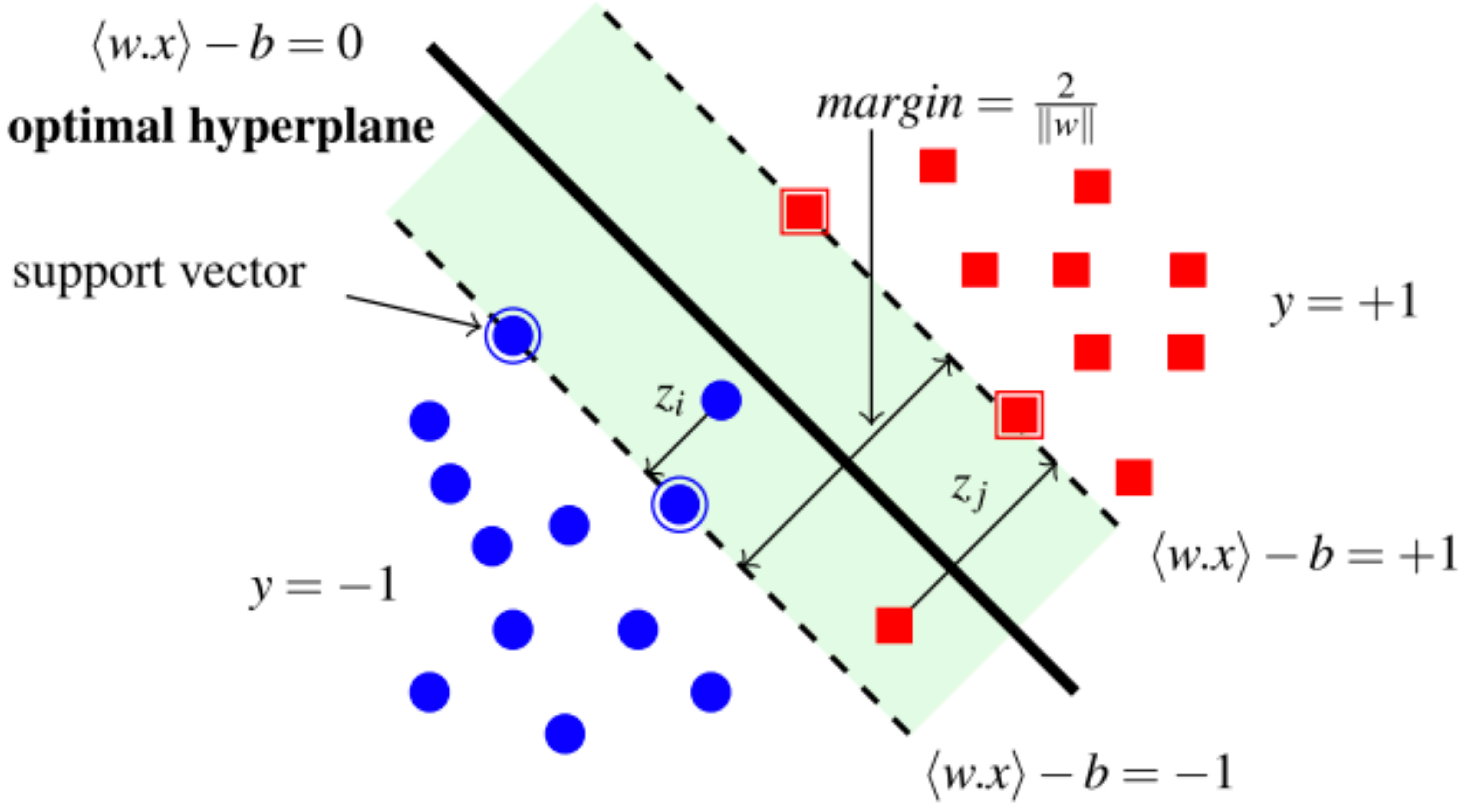
Main examples: ***support vector machine*** and ***logistic regression***

Support Vector Machines

Vapnik, V. (2013). The nature of statistical learning theory. Springer science & business media.
Do, T. N. (2020). Automatic Learning Algorithms for Local Support Vector Machines. SN Computer Science, 1(1), 1-11.

The main idea is to construct a hyperplane or set of hyperplanes in a high-dimensional space that can be used to separate different classes of data instances.

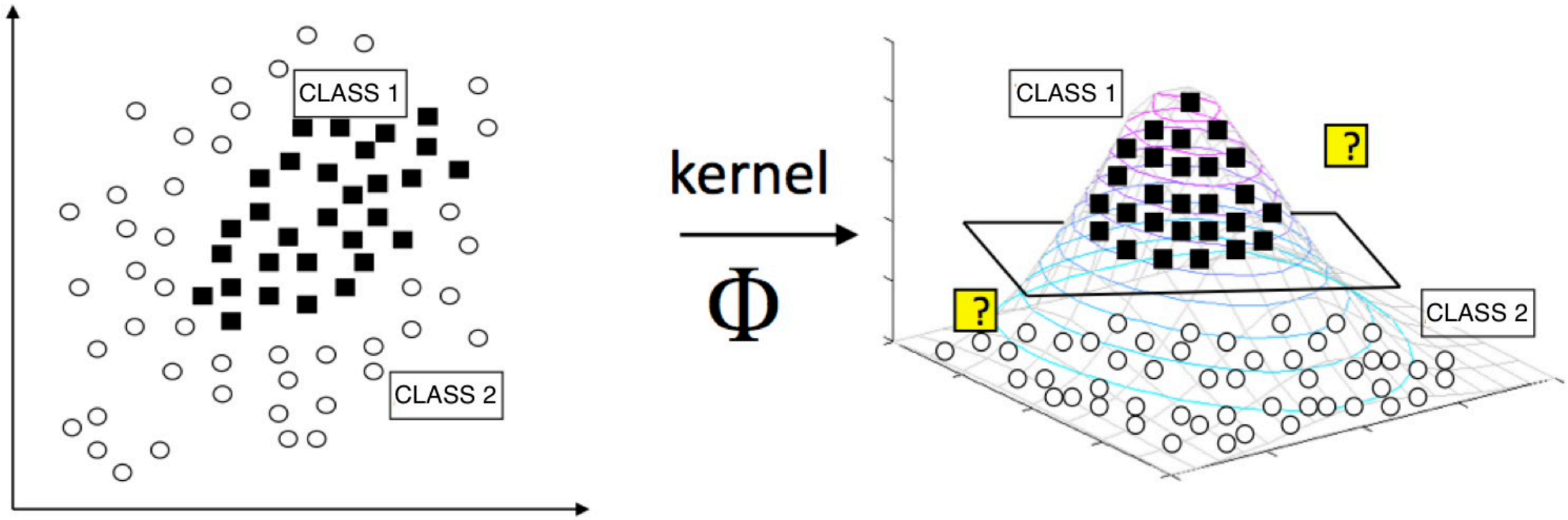
The good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.



$\langle w.x \rangle - b = 0$
optimal hyperplane
$margin = \frac{2}{\|w\|}$
support vector
$y = +1$
z_i
z_j
$y = -1$
$\langle w.x \rangle - b = +1$
$\langle w.x \rangle - b = -1$

Non linearly separable data (kernel trick)

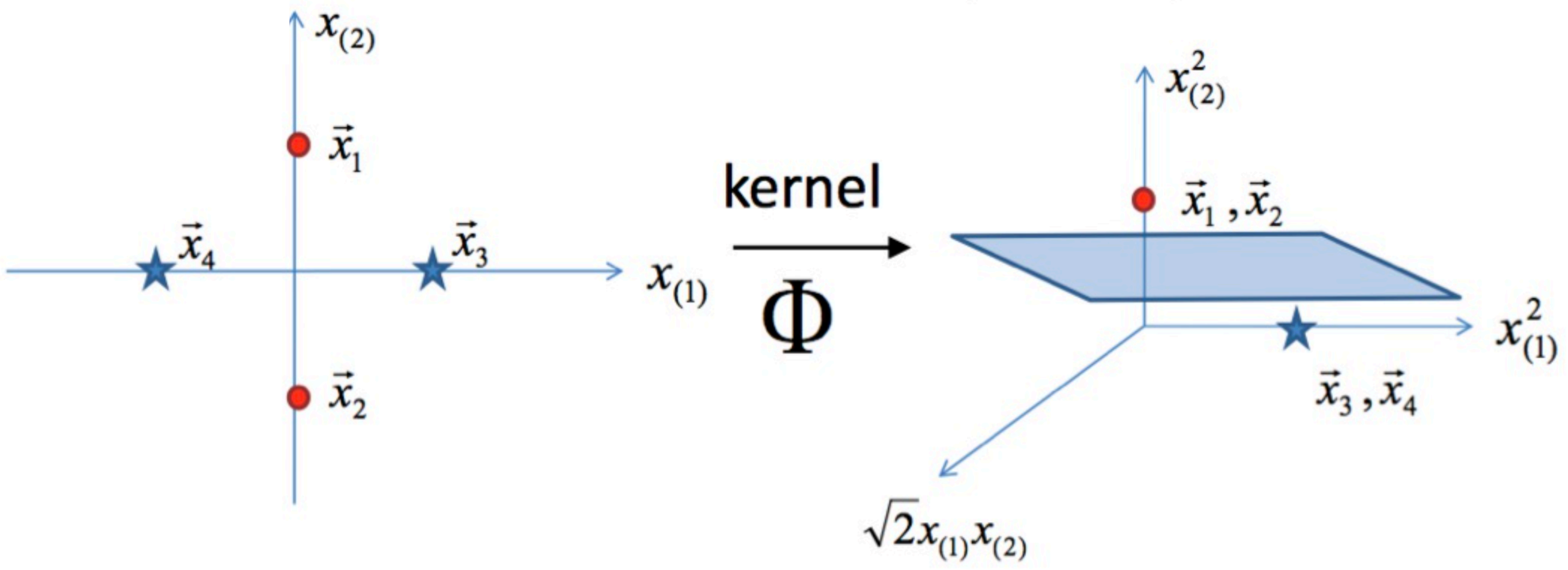
When data are not linearly separable, we reshape the feature space in order to find a way to detect a linear separation boundary among the classes



A Kernel is a dot product in some different feature space

$$K(\vec{x_i}, \vec{x_j}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$$

mapping is $\Phi(\vec{x}) = \begin{pmatrix} x_{(1)}^2 \\ \sqrt{2}x_{(1)}x_{(2)} \\ x_{(2)}^2 \end{pmatrix}$



Example of Kernels

$\vec{x_i} \cdot \vec{x_j}$ Linear

$\exp\left(-\lambda || \vec{x_i} - \vec{x_j} ||\right)$ Exponential

$\exp\left(-\lambda || \vec{x_i} - \vec{x_j} ||^2\right)$ Gaussian

$(p - \vec{x_i} \cdot \vec{x_j})^q$ Polynomial

$\tanh\left(k\vec{x_i} \cdot \vec{x_j} - \delta\right)$ Sigmoidal

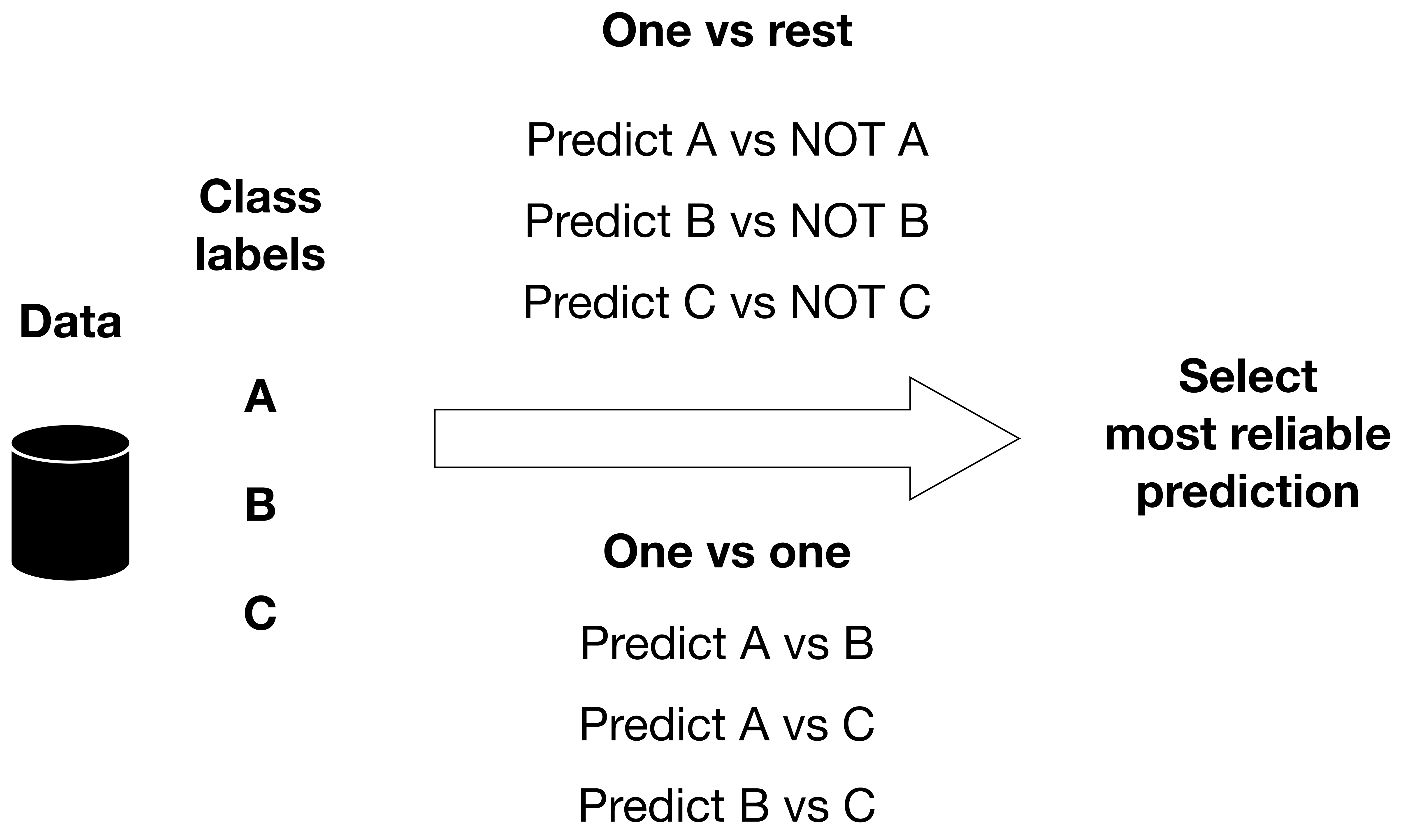
$(p - \vec{x_i} \cdot \vec{x_j})^q \exp\left(-\lambda || \vec{x_i} - \vec{x_j} ||^2\right)$ Hybrid

Classification strategies

Some classifiers are mainly conceived for **binary classification**. In order to adapt those to multi-class classification two alternative strategies may be enforced

<u>One vs rest</u>
In the One vs rest strategy a multi class classification problem is tackled by a number of binary classifier instances, one trained for predicting a class vs all the others. The final prediction is from the most confident binary classifier.

<u>One vs one</u>
Like One vs rest but this time all the classes are compared against the others pairwise



Data

Class labels

A

B

C

One vs rest

Predict A vs NOT A

Predict B vs NOT B

Predict C vs NOT C

One vs one

Predict A vs B

Predict A vs C

Predict B vs C

Select most reliable prediction

Classification strategies for multi-label

Exploit **binary classification**

A classifier is trained on each class independently and then we set a threshold on the confidence of each prediction to make it possible to assign an item to more than one class

Classifier A	$\hat{y}_i = A$	confidence 0.8
Classifier B	$\hat{y}_i = B$	confidence 0.6
Classifier C	$\hat{y}_i = C$	confidence 0.2

Exploit **multiclass classification**

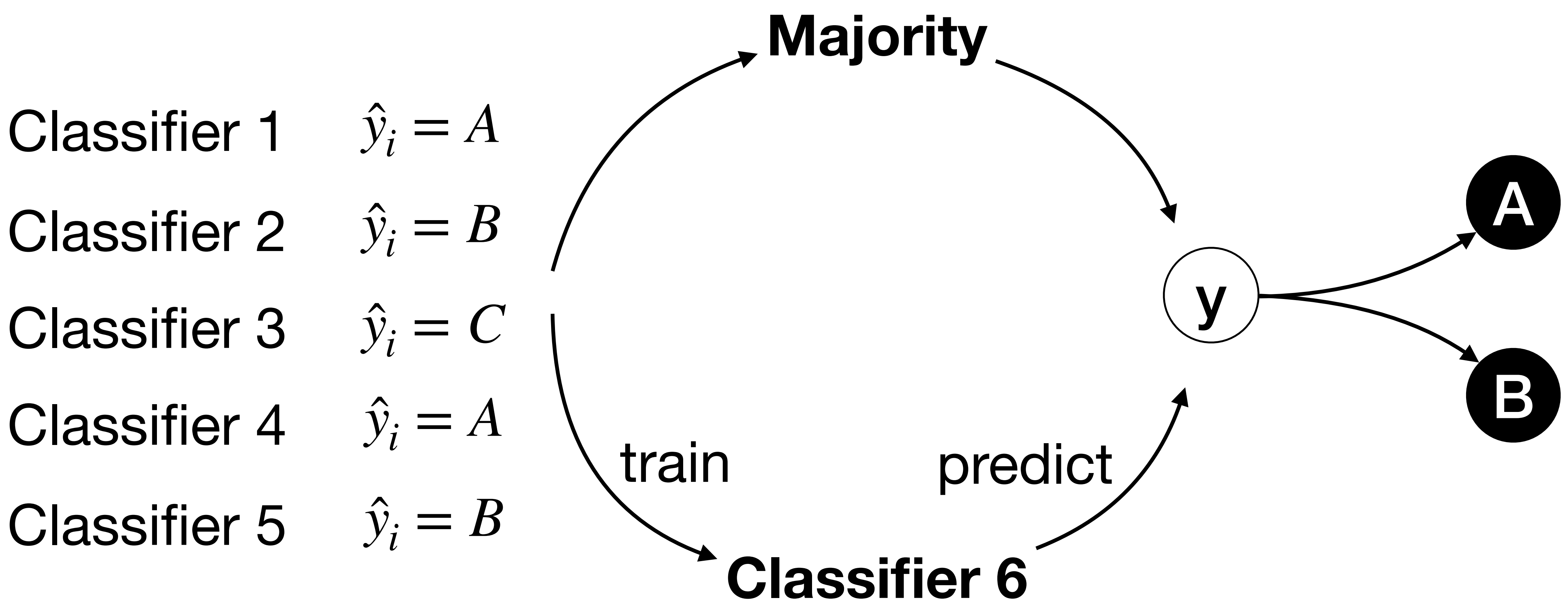
All the elements in the power set of the class labels are taken as a possible class and we perform multi-class classification is usual by modifying the training set

Exploit a classifier supporting multiple **class assignments**

Some classifiers, such as KNN and Naive Bayes can estimate a relevance measure for each class. This can be used to associate an item with more than one class

Ensemble

A set of individual multi-class classifiers is exploited to make distinct predictions. The output is used to select the final classes by a majority voting system of by training a further classifier


Classifier 1 $\hat{y}_i = A$
Classifier 2 $\hat{y}_i = B$
Classifier 3 $\hat{y}_i = C$
Classifier 4 $\hat{y}_i = A$
Classifier 5 $\hat{y}_i = B$
Majority
train
predict
Classifier 6
y
A
B

Hyperparameters tuning

Model selection is the activity of tuning the hyper parameters of a model with respect to the task at hand

Grid search. The possible values of different parameters are organized in a grid and the model is then trained on a dataset for all the possible combinations of parameter values. After evaluation the best combination is retained.

Grid search may be executed also by **randomly select parameters combinations** or by **halving the combinations** as in a sort of tournament

Hyperparameters tuning

When evaluating different settings for estimators there is a risk of **overfitting** on the test set. To solve this problem, yet another part of the dataset can be held out as a so-called **“validation set”**: training proceeds on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set.

However, by partitioning the available data into three sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets.

A solution to this problem is a procedure called **cross-validation (CV for short)**. A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV. In the basic approach, called **k-fold CV**, the training set is split into k smaller sets. The following procedure is followed for each of the k “folds”:

- A model is trained using **k - 1 of the folds** as training data;
- the resulting model is validated on the **remaining part of the data**.

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop.

Evaluation

Evaluation is based on the **confusion matrix** where the entries are the expected and predicted class labels. This way, we have a notion of precision and recall at the class level.

```
              precision    recall  f1-score   support

     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3

    accuracy                           0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
```

To achieve a comprehensive score we can take:

- **macro average:** averaging the unweighted mean per label
- **weighted average:** averaging the support-weighted mean per label
- **micro average:** averaging the total true positives, false negatives and false positives (this is only for multi-label or multi-class with a subset of classes, because it corresponds to accuracy otherwise and would be the same for all metrics)