School of Electronic Engineering and Computer Science
Queen Mary University of London

CBU5201 Principles of Machine Learning

Supervised learning: Classification I

Dr Chao Liu

Credit to Dr Jesús Requena Carrión

Oct 2023

Flexibility, complexity and overfitting

In any machine learning project we assume that our samples follow a **pattern** and any deviation from this pattern is considered to be **noise**.

Our models need to be flexible enough to **capture the complexity of the underlying pattern**, but no too flexible, as we might end up **memorising irrelevant noise** details in our data.

A **rigorous methodology** is crucial to avoid falling into common traps, such as overfitting.

So someone has collected our dataset? Great! Let's go ahead, put our methodology to work and build a fantastic model. **Great?**

The 1936 Literary Digest Poll

Alfred Landon
Republican Party

Franklin D. Roosevelt
Democratic Party

- The Literary Digest conducted one of the largest polls ever.
- Predicted Landon would get 57 % of the vote, Roosevelt 43 %. Landon ended up getting 38 % of the votes and Roosevelt 62 %.
- What happened? **Bad sampling**. Names were taken from telephone directories, club membership lists, magazine subscribers lists, etc. Samples were **not representative** of the population.

Know your data!

Agenda

Formulating classification problems

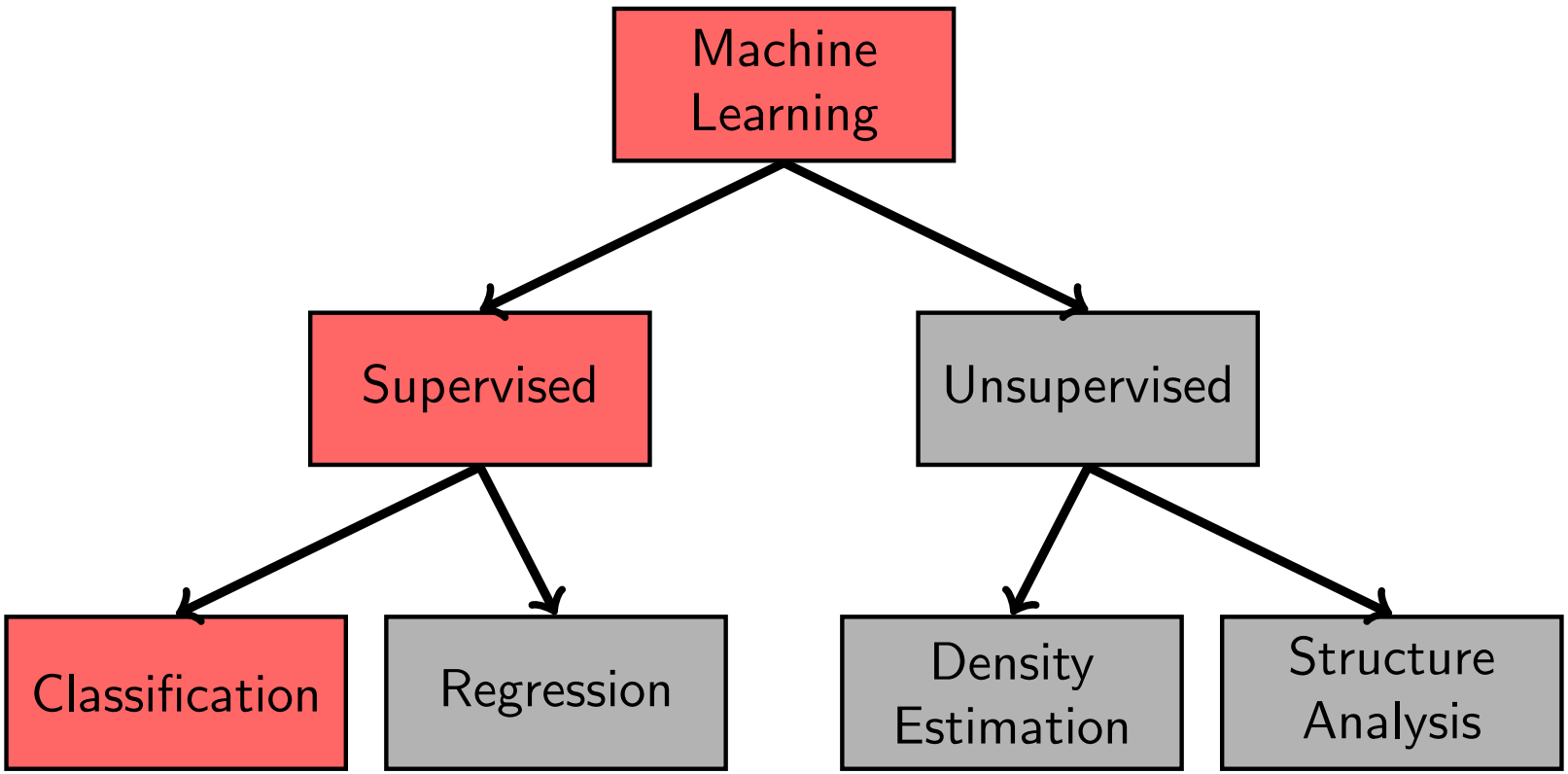
Linear classifiers

Logistic model

Nearest neighbours

Summary

Machine Learning taxonomy



Classification: Problem formulation

In **classification** (also known as **decision** or **detection**) problems:

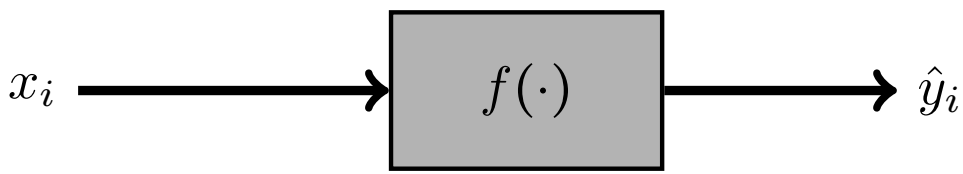
- We want to build a model that produces a **label** when shown a set of **predictors**.
- The label is **discrete** and each of its values is known as a **class**.

This is not machine learning yet. Can you tell why?

Classification: Problem formulation

In a machine learning classification problem:

- We build a model $\hat{y} = f(x)$ **using a dataset** $\{(x_i, y_i) : 1 \leq i \leq N\}$.
- We have a notion of **model quality**.
- The pair (x_i, y_i) can be read as "sample i belongs to class y_i", or "the label of sample i is y_i".



A binary classification problem

Sentiment analysis seeks to identify human opinions implied by a fragment of text. Multiple opinions can be considered, but in its simplest form two are defined, namely positive and negative.

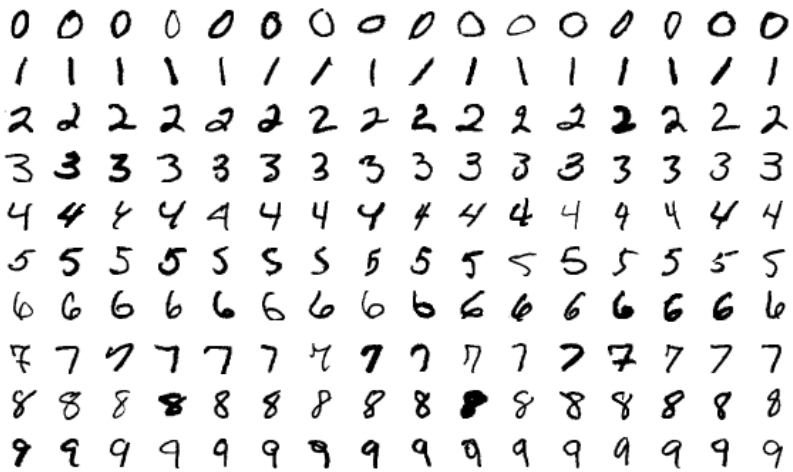
The *Large Movie Review Dataset* was created to build models that recognise polar sentiments in fragments of text:

- It contains 2500 samples for training and 2500 samples for testing
- Each instance consists of a **fragment of text** used as a **predictor** and a **binary label** (0 being negative opinion, 1 positive opinion).
- Downloadable from ai.stanford.edu/~amaas/data/sentiment/

A multiclass classification problem

Recognising digits in images containing handwritten representations is a classic multiclass classification problem. The predictor is an array of values (image) and there are 10 classes, namely $0, 1, 2, \ldots 9$.

In machine learning we use datasets of **labelled images**, i.e. pairs of **images** (predictors) and **numerical values** (label), to build such models.



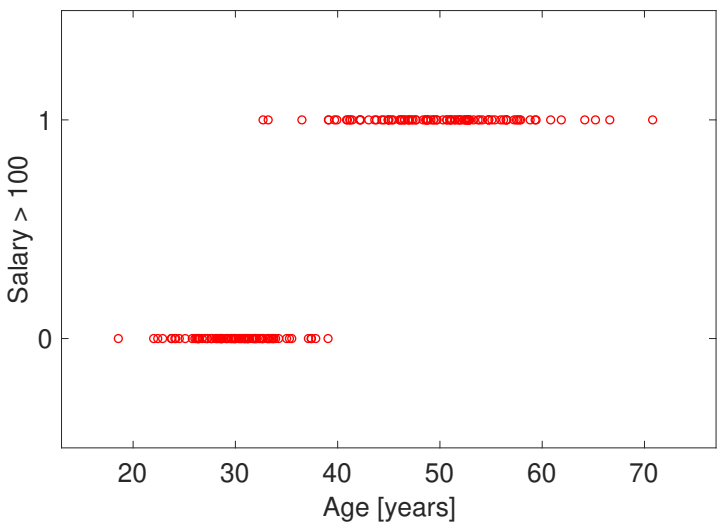
The MNIST dataset is a collection of handwritten digits:

- 60,000 images for training, 10,000 for testing
- Images are black and white, 28×28 pixels
- Downloadable from yann.lecun.com/exdb/mnist

The dataset in the attribute space

Labels can be represented by numerical values on a vertical axis. Be careful: the usual notions of **ordering and distance do not apply** to categorical variables.

One predictor, two classes

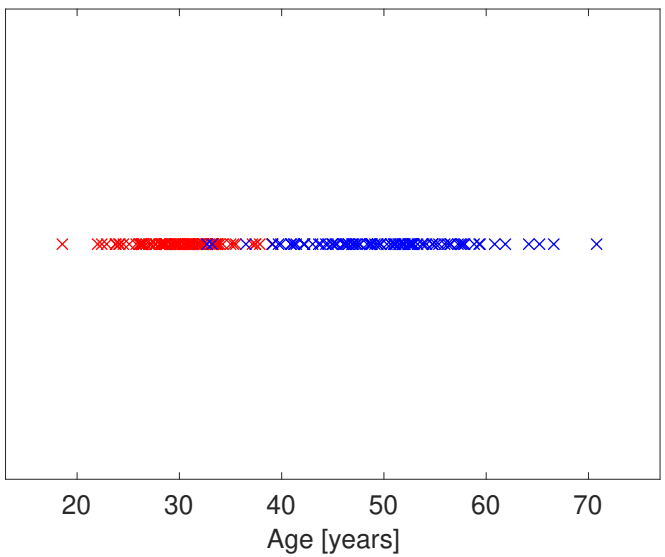


Two predictors, three classes

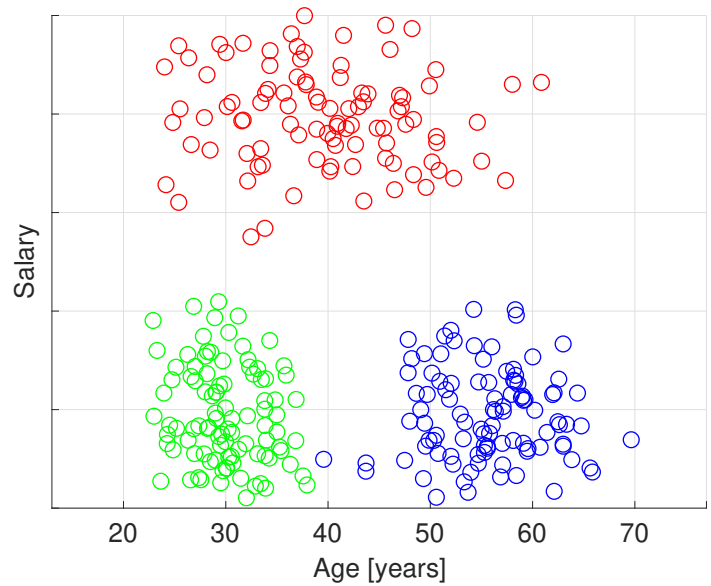
The dataset in the predictor space

A more convenient representation consists of using **different symbols for each label** in the **predictor space**.

One predictor, two classes



Two predictors, three classes

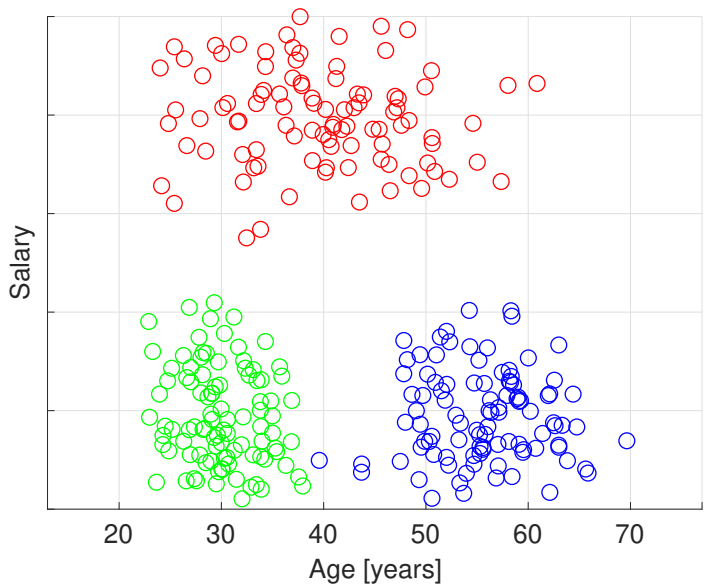


What does a classifier look like?

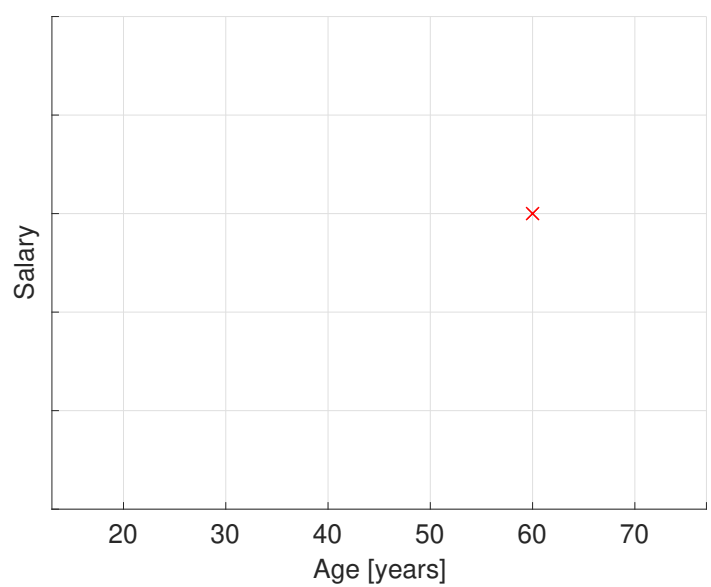
Regression models can be represented as curves/surfaces/hypersurfaces in the attribute space.

Now that we know how to represent our dataset in the predictor space, how can we represent a **classification model**?

Training data



New data point



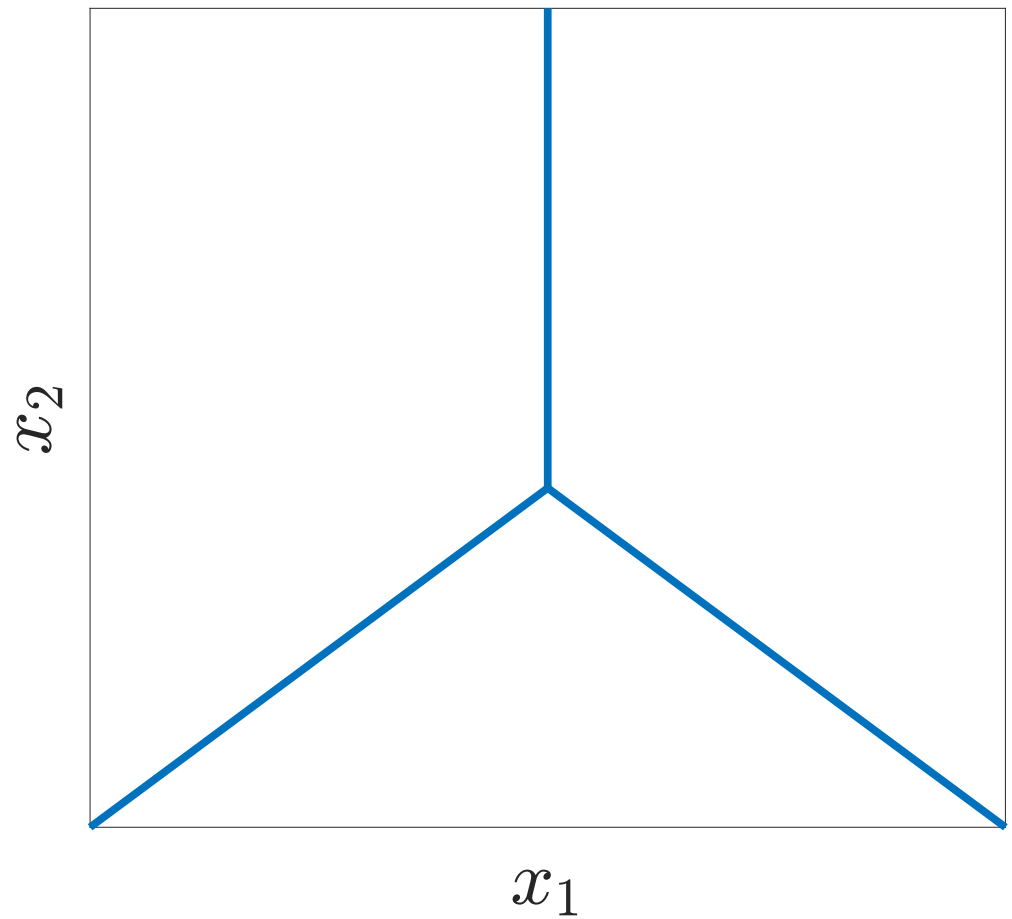
What does a classifier look like?

In classification problems we use the notion of **decision regions** in the predictor space.

- A decision region is made up of **points that are associated to the same label**.
- Regions can be defined by identifying their **boundaries**.
- A solution model in classification is a **partition of the predictor space** into decision regions separated by decision boundaries.

What does a classifier look like?

In machine learning we use data to build models: When we build classifiers, we use data to define their decision regions.



Agenda

Formulating classification problems

Linear classifiers

Logistic model

Nearest neighbours

Summary

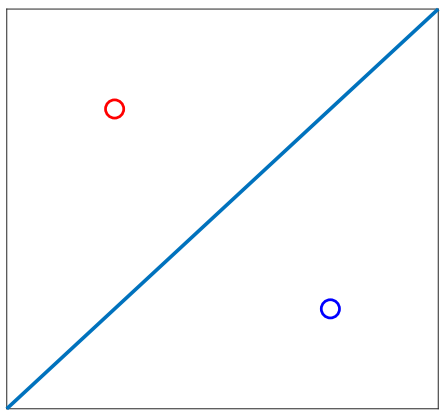
The simplest boundary

Let's consider a **binary** classification problem. The simplest boundary is:

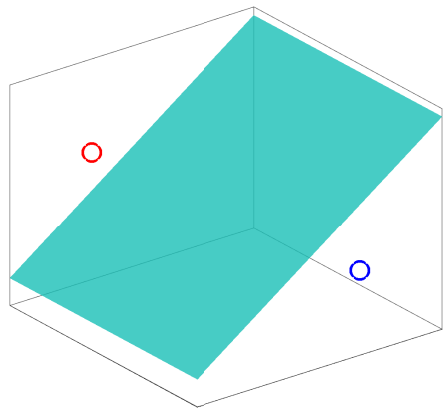
- A single point (known as **threshold**) in 1D predictor spaces.
- A straight line in 2D predictor spaces.
- A plane surface in 3D predictor spaces.

These are **linear** boundaries.

1 predictor



2 predictors



3 predictors

Definition of linear classifiers

Linear classifiers use **linear boundaries** between decision regions:

- Linear boundaries are defined by the **linear equation** $\boldsymbol{w}^T\boldsymbol{x} = 0$.
- The extended vector $\boldsymbol{x} = [1, x_1, x_2 ...]^T$ contains the predictors and $\boldsymbol{w}$ is the coefficients vector.
- To classify a sample we simply identify the **side of the boundary** where it lies.

Definition of linear classifiers

If we know the coefficients vector $\boldsymbol{w}$ of a linear boundary, classifying a sample is very simple:

- Build the extended vector $\boldsymbol{x}_i$.
- Compute $\boldsymbol{w}^T \boldsymbol{x}_i$.
- Classify using the following **facts**:
  - If $\boldsymbol{w}^T \boldsymbol{x}_i > 0$, we are on one side of the boundary.
  - If $\boldsymbol{w}^T \boldsymbol{x}_i < 0$, we are on the other!
  - If $\boldsymbol{w}^T \boldsymbol{x}_i = 0$... where are we?

Now that we know how to **use** linear classifiers during deployment, let's consider the problem of **building the best** one given a dataset. To answer this question, we need to define our **quality metric** first.

A basic quality metric

The only operation that we can perform with categorical variables is **comparison**, i.e. we can assess whether $y_i = \hat{y}_i$ is either true or false.

By comparing predictions and true labels, we can identify in a dataset:

- The correctly classified samples (**true predictions**) in each class.
- The incorrectly classified samples (**false predictions**) in each class.

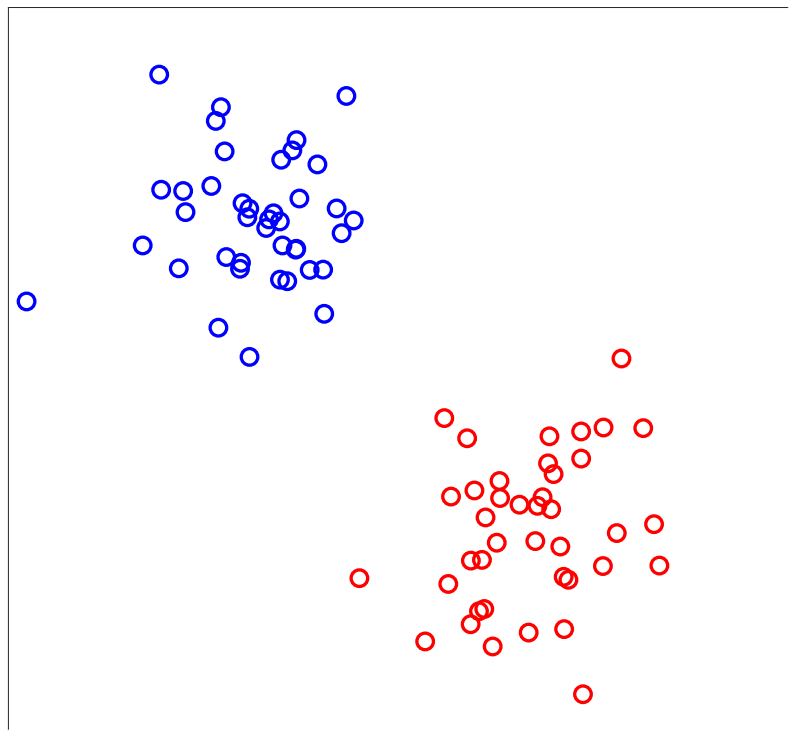
Two common and equivalent notions of quality are the **accuracy** A and the **error** (or misclassification) **rate** $E = 1 - A$, defined as:

$$A = \frac{\#\text{correctly classif. samples}}{\#\text{samples}} \quad , \quad E = \frac{\#\text{incorrectly classif. samples}}{\#\text{samples}}$$

Using these notions of quality, the best classifier can be defined as the one with the highest accuracy (or the lowest misclassification rate).

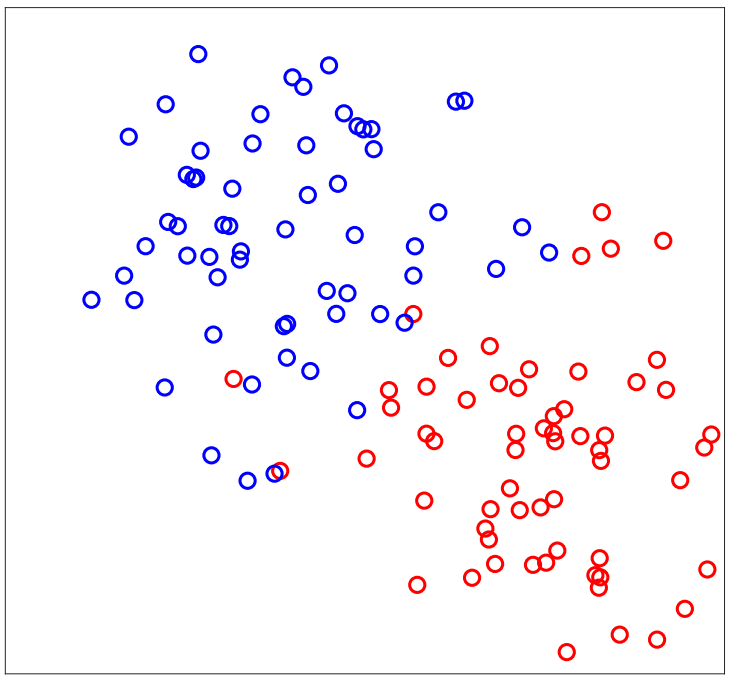
The best linear classifier: Separable case

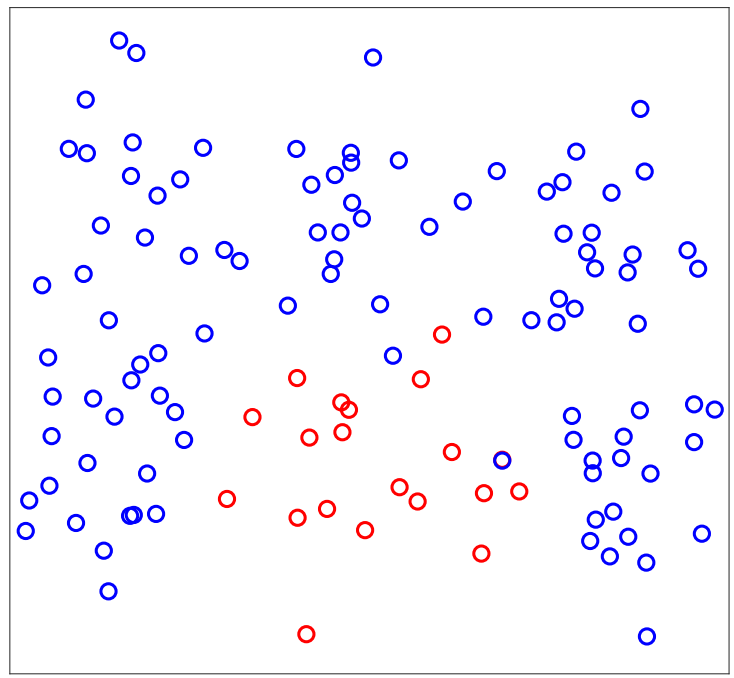
In linearly separable datasets, we can find a linear classifier that achieves the maximum accuracy ($A = 1$, $E = 0$).



The best linear classifier: Non separable case

In non linearly-separable datasets, the accuracy of a linear classifier is $A < 1$ ($E > 0$). The best one will achieve the highest accuracy.





Finding the best linear classifier

We have introduced the linear classifier and presented two quality metrics (accuracy and error rate). If we are given a linear classifier $\boldsymbol{w}$, we can use a dataset to calculate its quality.

The question we want to answer now is, how can we **use data to find** the best linear classifier?