Introduction to Information Sciences

Machine Learning

mcuturi@i.kyoto-u.ac.jp

Machine Learning: Definition

A **computer program** is said to learn from **experience** E
with respect to **some class of tasks** T and **performance measure** P,
if its performance on such **tasks** T, as **measured by** P,
improves with **experience** E.

Typically, behind these concepts lie simple ideas

- **experience** E : database, collected information
- **tasks** T : classification, regression, clustering,
- **measured by** P: mistakes, successes $etc.$

The big picture

Some intuitions on machine learning

- Imagine you have seen this movie:

- A friend comes to you and asks you:

  I feel like going to the movies tonight, do you think I will like this movie?

- How would you build your answer?

Some intuitions on machine learning

Machine learning helps industries build such **answers** *automatically*

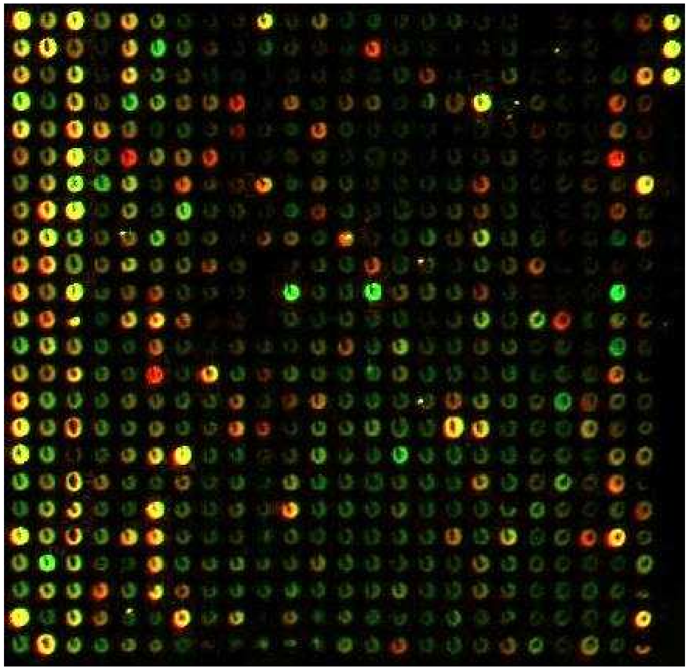
- Imagine you are a DVD rental company.
- It is **part of your business** to recommend good movies to your customers.
- **large scale task:** for 1,000's or 1,000,000's of customers every day!
- Still the same question: would you recommend *Ironman* to customer `AD13242`?

Some intuitions on machine learning

- A computer program **also needs side information**
- For instance:
  - age & background of the user → Check his inscription form.
  - Better! a few examples of movies `AD13242` has seen, with his **ratings**
  - *Lord of the rings I* (+++), *Star Wars I* (++), *Shrek 2* (-) *etc.*.
- How can we decide if we should recommend *Ironman* to `AD13242`?

A more serious problem

- Given the DNA profile of a patient...



- Can we answer (approximately) the questions:
  - What is this patient's **cancer** risk in the next years?
  - What **treatments** can be effective for this patient?

Very fast progress in last years, from theory to practice

You can do a websearch on `mammaprint` or `23andme`

Not only biology or movies..

Data we can learn from is everywhere

Biology : DNA chips, complex biological pathways.
Medicine : scans, 24/24 measurements of patients.
Business : commercial transactions online and offline.
Search engines : audio, video and textual contents.
Finance : electronic markets, quotes and transactions tick by tick.
Physical interactions : highway networks, mobile phones, GPS localization.
Sociological and physical interactions : social networks on internet, surveillance.
etc.

$\Downarrow$

Data *acquisition* is *cheap* $\neq$ **Data analysis is more difficult**

$\Downarrow$

Need for **data-driven algorithms**
to **fill the gap** between
storing complex data and **understanding it**

Build decision functions

- In many situations, we want to answer a question:

> Given a certain observation, summarized by measurements x,
> what can happen/should we do?

- In mathematical terms, we want to build a function:

$$\begin{aligned} f: \quad \mathcal{X} &\rightarrow \mathcal{Y} \\ x &\mapsto f(x) \end{aligned}$$

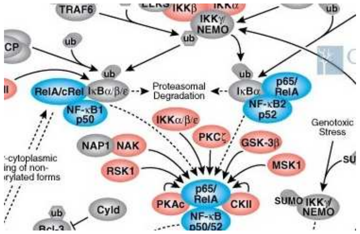
  - $\mathcal{X}$ could be: images, texts, movies, *etc.*
  - $\mathcal{Y}$ could be: "yes/no", real numbers, sentences *etc.*

> Our goal: build a **computer program** that outputs a **useful** $f(x)$

Build decision functions

A few examples in the industry

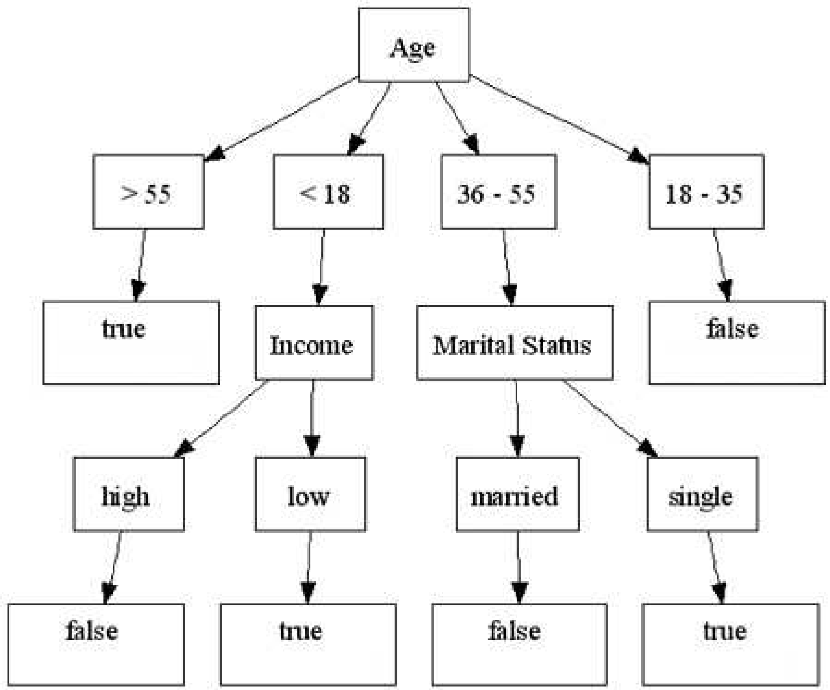
- Ranking answers to a problem,
- Learning jointly different related tasks,
- Learn maps between structured data, $e.g.$ translation
- Build interaction maps, *e.g.* for proteins,



- Trade automatically stocks and financial products
- Learn with very large databases: shopping.
- $etc.$

What Machine Learning is NOT about

- 100% Man-made, rule-based decision trees.



- **advantages**: sometimes expertise available, just need to **rationalize** it.*etc.*
- **disadvantages**: difficult to **replicate**, unadapted for **large** systems and **new problems** (DNA) where no expertise exists by definition!

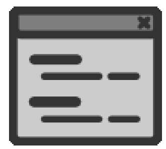
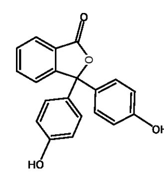
What Machine Learning is about

- Use data collected in databases as the **main ingredient** to build f.

$\rightarrow \quad f$

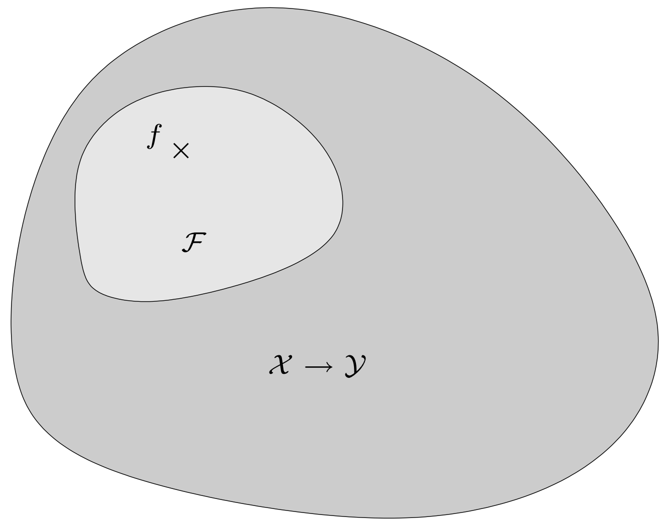
- Build architectures where **machines** can **learn** from these databases.

Probabilistic Framework / Structures

- **Random**
  - Unlike deterministic systems, we assume **randomness**.
  - **Future** requests are **not known**. Some are **more likely**.
- **Structured, complex**
  - strings, texts and sequences, 
  - images, audio and video feeds,  
  - graphs, interaction networks and 3D structures 

Ingredients to pick a good f

- A set of candidates $\mathcal{F}$.



- A way to use the database (past observations)
  - **Data-dependent** criterion C_{data} to select f.
  - Usually given a function g, $C_{\text{data}}(g)$ big if g not accurate on the data.
- A method to find an **optimal** candidate in $\mathcal{F}$.

$$f = \text{argmin}_{g \in \mathcal{F}} \quad C_{\text{data}}(g).$$

Depending on $\mathcal{Y}$...

- When $\mathcal{Y}$ is a **subset** of $\mathbb{R}^d$, finding a good f $\Leftrightarrow$ regression.

- When $\mathcal{Y}$ is a **finite** set of labels, finding a good f is called classification.

Regression : Nearest-Neighbour Methods

Pricing the Rent of Apartments near Kyoto University

Collected information about 285 (out of 1226) apartments close to Kyoto U.

Kept 4 variables: **Age of Building**, **Surface**,**Walking distance to station**, **Rent**.

What does the data look like?

imagecs(H); colorbar;

285 columns, 4 lines.

Each column represents one apartment.

In these slides, we will **guess** the rent of using age, surface and distance

Nearest-Neighbours of a given point $\mathbf{x}$

- Consider matrix H
- This matrix describes
  - apartments seen as triplets (area,surface,distance),
  - with their rent.
- Namely, for i going from 1 to $N = 285$,
  - $\mathbf{x}_i \in \mathbb{R}^3, \mathbf{x}_i = (a_i, s_i, d_i)$,
  - rent r_i.

Nearest-Neighbors of a given point $\mathbf{x}$

- Given an apartment described as $\mathbf{x} = (a, s, d)$, how can we guess its rent?
- Consider the Euclidian distance between $\mathbf{x}$ and $\mathbf{x}_i$.

$$d(\mathbf{x}, \mathbf{x}_i) = \|\mathbf{x} - \mathbf{x}_i\|^2.$$

- A **nearest-neighbor** of $\mathbf{x}$ is any element $\mathbf{x}_j$ such that

$$d(\mathbf{x}, \mathbf{x}_j) = \min_{i=1,\cdots,N} d(\mathbf{x}, \mathbf{x}_i)$$

- By extension, a **set of k nearest neighbors** $\mathbf{x}_{j_1}, \mathbf{x}_{j_1}, \cdots, \mathbf{x}_{j_k}$ is simply the set of k distinct points of $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ which are the closest to $\mathbf{x}$.
- Equivalently, for $r \leq k$,

$$d(\mathbf{x}, \mathbf{x}_{j_r}) = \min_{i \in \{1,\cdots,N\} \setminus \{j_1,\cdots,j_{r-1}\}} d(\mathbf{x}, \mathbf{x}_i)$$

k-Nearest-Neighbors Rule

- Given a point $\mathbf{x}$,
  - Find its k-nearest neighbours, $j_1, \cdots, j_k$,
  - $e.g.$by ordering $d(\mathbf{x}, \mathbf{x}_i)$ from smallest to highest and taking the k first corresponding indices
  - Return

$$\frac{\text{rent}_{j_1} + \cdots + \text{rent}_{j_k}}{k}.$$

- Let us program this directly for the set of apartments and check how it works.

Classification : Perceptron

When the set $\mathcal{Y}$ is a finite collection of labels

Multiclass Classification

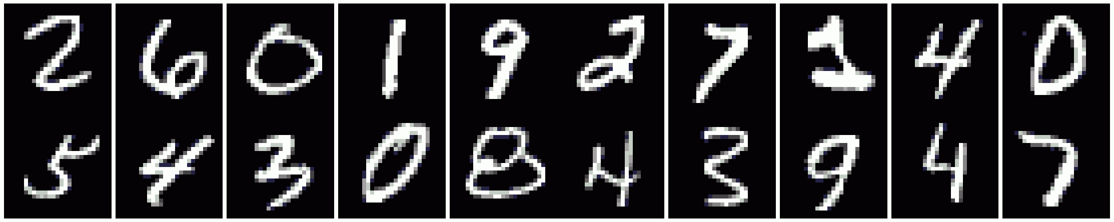
- Classify images of fruits into fruit category

- Classify musical tunes, books, movies into genres
- Classify proteins into functional classes

img source

Digits recognition

- Classify images of handwritten digits into digits from 0 to 9
- Use a database such as



paired with the corresponding labels,

$$(2, 6, 0, 1, 9, 2, 7, 1, 4, 0, 5, 4, 3, 0, 8, 4, 3, 9, 4, 7).$$

to build an **automated recognition system** for handwritten digits.

- useful for post office, check recognition, tax office, *etc.*.

When the set $\mathcal{Y}$ only has two elements

Binary Classification

- Using elementary measurements, guess if someone **has or not** a disease that is
  - difficult to detect at an early stage
  - difficult to measure directly (fetus)
- Classify chemical compounds into **toxic / nontoxic**
- Classify a passenger as **suspect/not suspect**
- Classify body tumor as **begign/malign** to detect cancer
- *etc.*

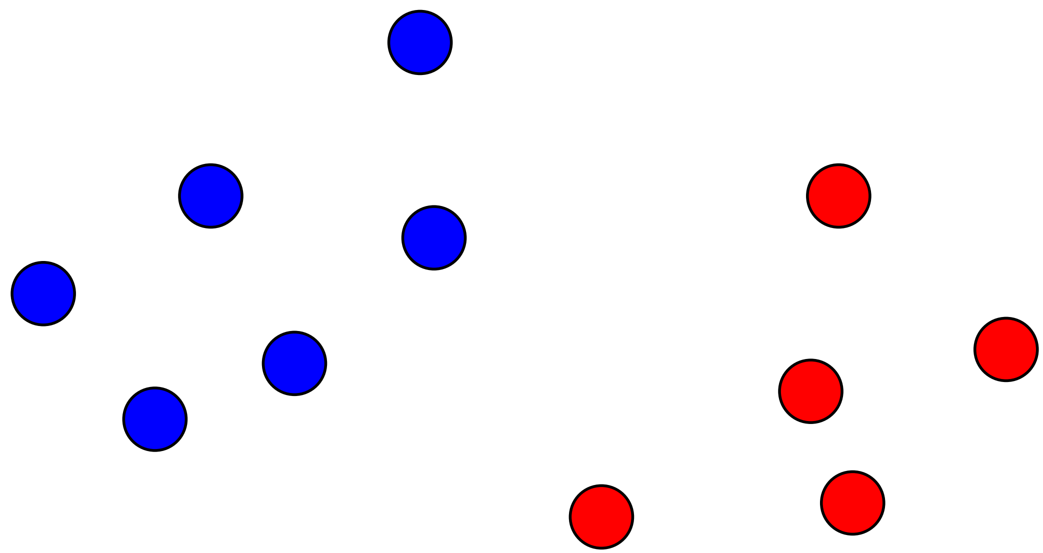
Mathematical Formulation for Binary Classification

- The **Data** we have: a bunch of vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots, \mathbf{x}_N$.
- Ideally, to infer a "yes/no" rule, we need the **correct answer** for each vector.
- We consider thus a set of **pairs of vector/bit**

$$\text{"training set"} = \left\{ \left(\mathbf{x}_i = \begin{bmatrix} x_1^i \\ x_2^i \\ \vdots \\ x_d^i \end{bmatrix} \in \mathbb{R}^d, \ \mathbf{y}_i \in \{0,1\} \right)_{i=1..N} \right\}$$

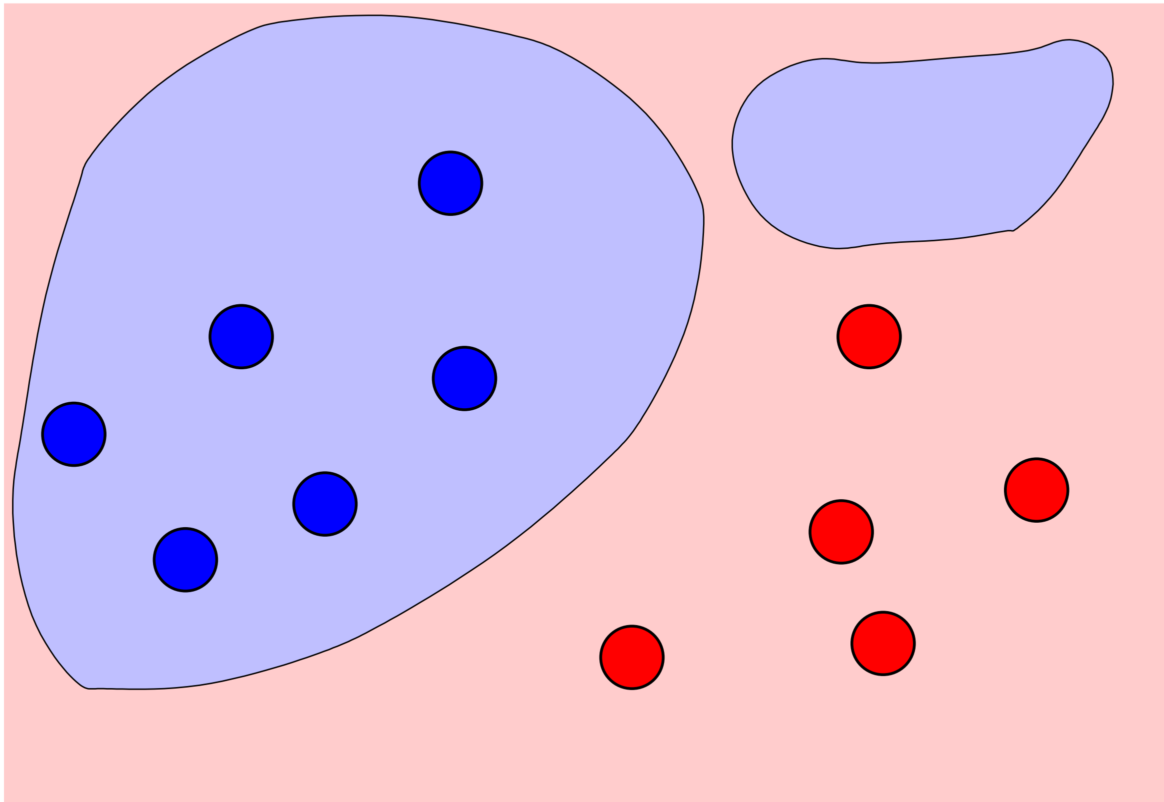
- For illustration purposes **only** we will consider **vectors in the plane**, $d = 2$.
- Points are easier to represent in 2 dimensions than in 20.000...
- The ideas for $d \gg 3$ are **exactly the same**.

Binary Classification Separation Surfaces for Vectors



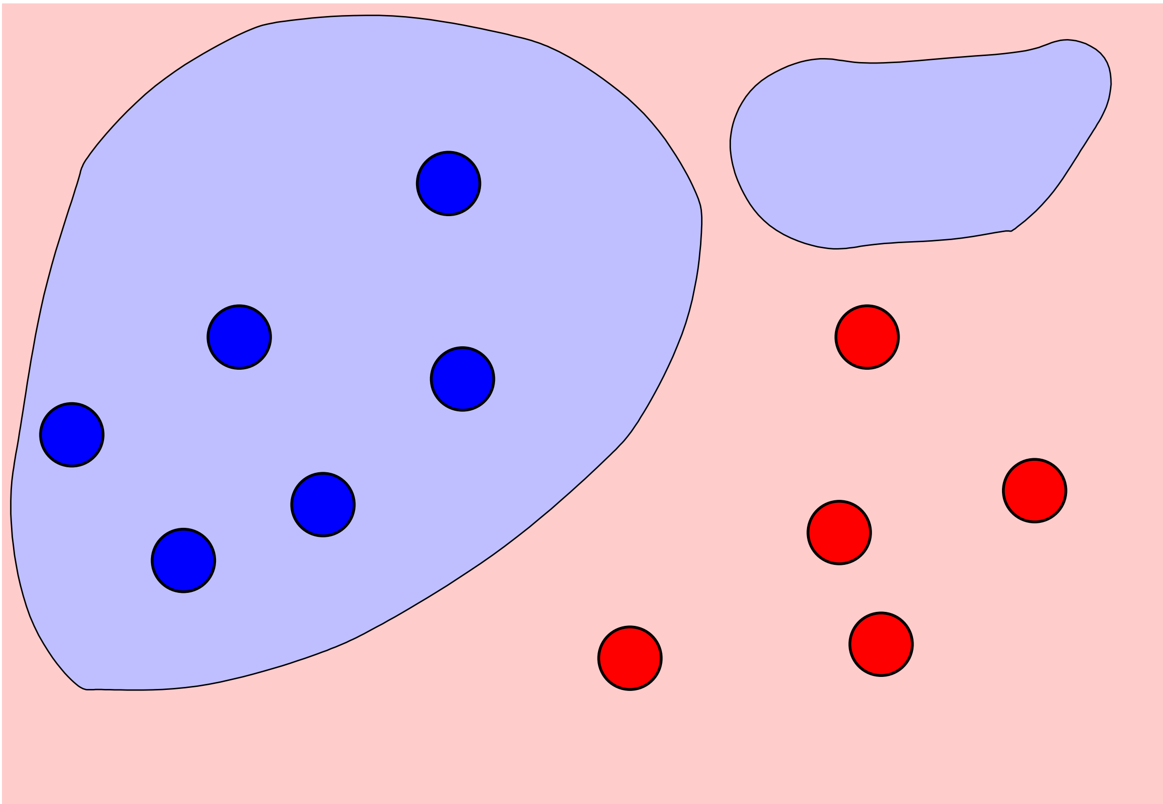
What is a classification rule?

Binary Classification Separation Surfaces for Vectors



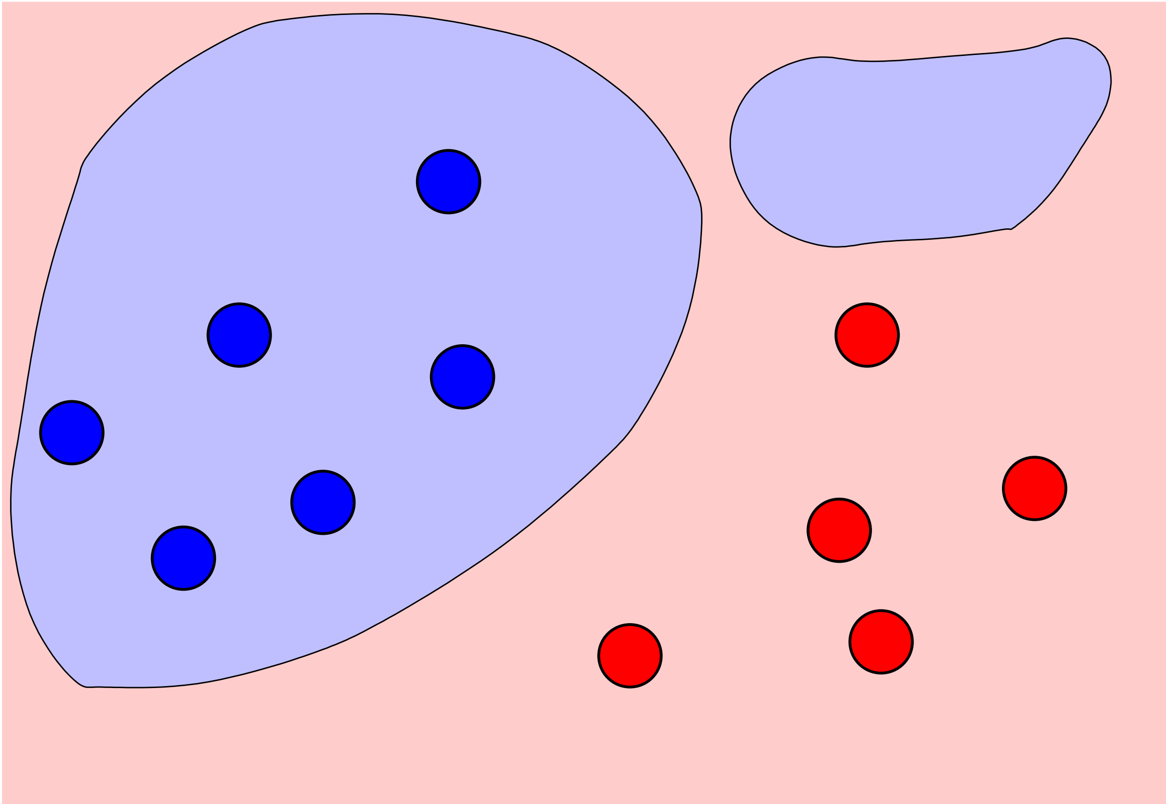
Classification rule = a partition of $\mathbb{R}^d$ into two sets

Binary Classification Separation Surfaces for Vectors



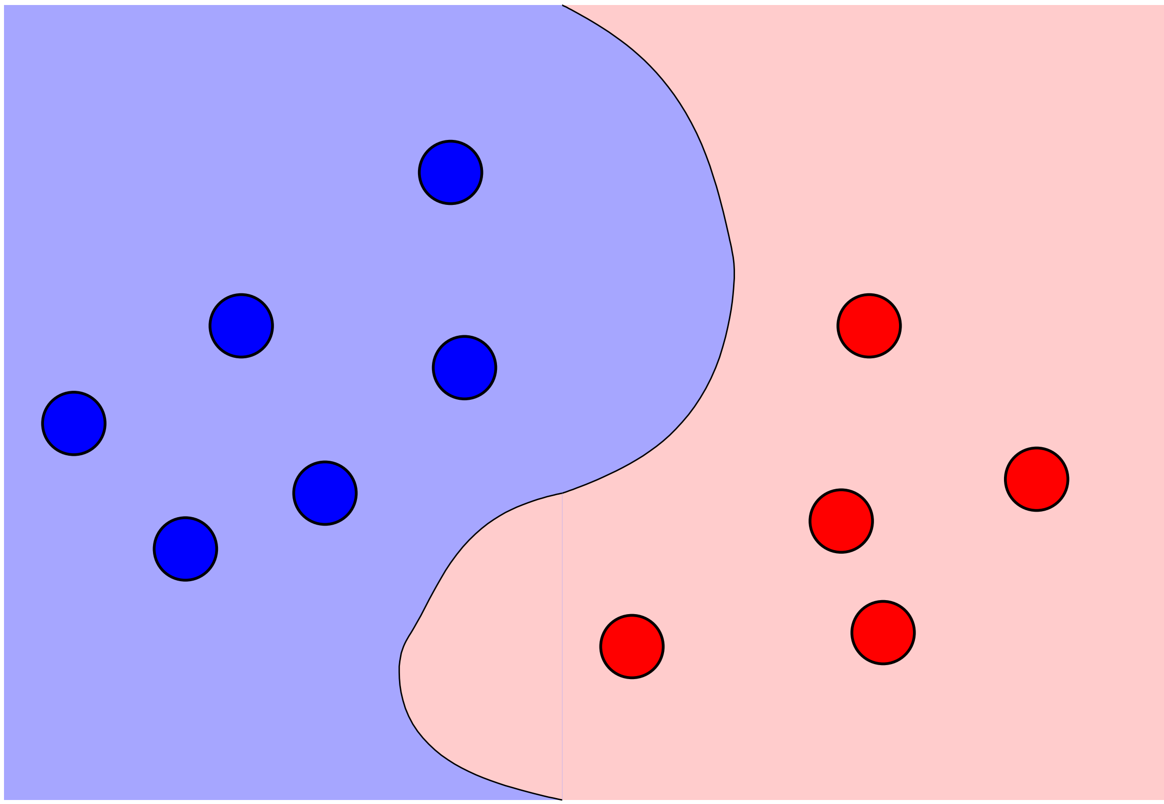
This partition is usually interpreted as the level set of function on $\mathbb{R}^d$

Binary Classification Separation Surfaces for Vectors



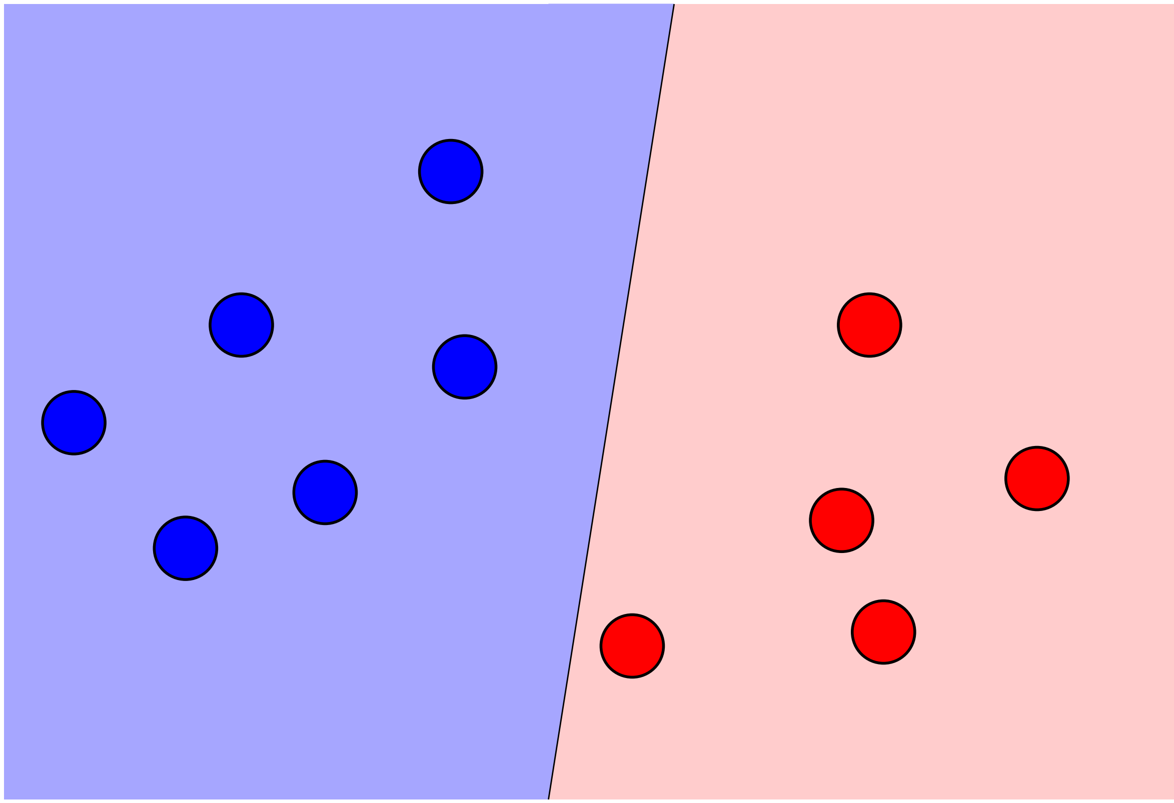
Typically, $\{\mathbf{x} \in \mathbb{R}^d | \boldsymbol{f}(\mathbf{x}) > 0\}$ and $\{\mathbf{x} \in \mathbb{R}^d | \boldsymbol{f}(\mathbf{x}) \leq 0\}$

Classification Separation Surfaces for Vectors



Can be defined by a single surface, *e.g.* a curved line

Classification Separation Surfaces for Vectors



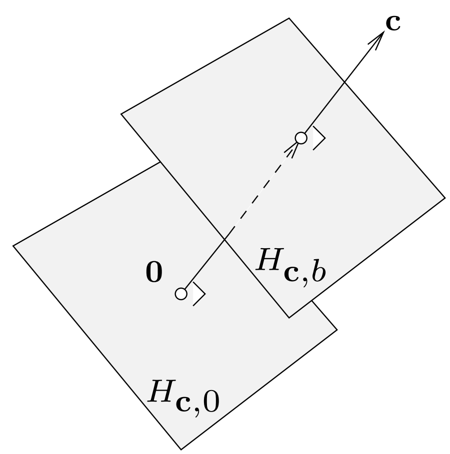
Even more **simple**: using **straight lines** and halfspaces.

Linear Classifiers

- **Straight lines** (hyperplanes when $d > 2$) are **the simplest type** of classifiers.
- A hyperplane $H_{\mathbf{c},b}$ is a set in $\mathbb{R}^d$ defined by
  - a normal vector $\mathbf{c} \in \mathbb{R}^d$
  - a constant $b \in \mathbb{R}$. as

$$H_{\mathbf{c},b} = \{\mathbf{x} \in \mathbb{R}^d \,|\, \mathbf{c}^T\mathbf{x} = b\}$$

- Letting b vary we can "slide" the hyperplane across $\mathbb{R}^d$

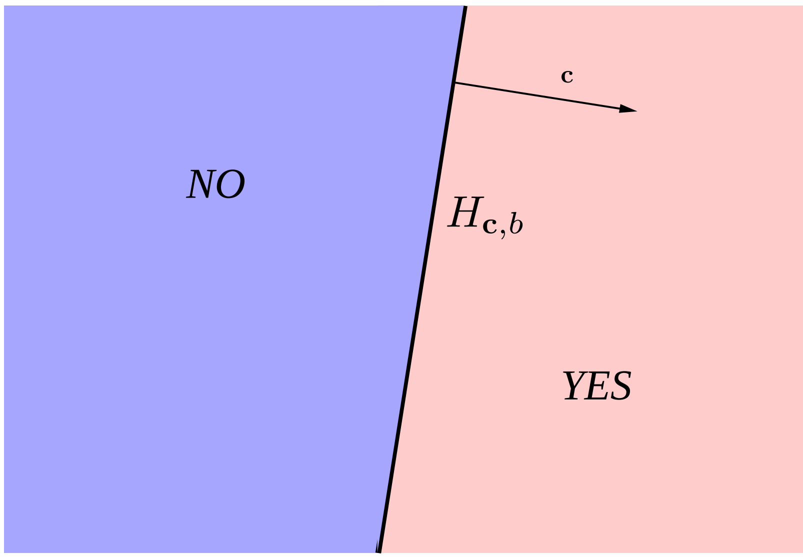


Linear Classifiers

- Exactly like lines in the plane, hypersurfaces **divide** $\mathbb{R}^d$ into **two** halfspaces,

$$\{\mathbf{x} \in \mathbb{R}^d \,|\, \mathbf{c}^T\mathbf{x} < b\} \cup \{\mathbf{x} \in \mathbb{R}^d \,|\, \mathbf{c}^T\mathbf{x} \geq b\} = \mathbb{R}^d$$

- Linear classifiers attribute the "yes" and "no" answers given arbitrary $\mathbf{c}$ and b.



- Assuming we only look at halfspaces for the decision surface...
  ...*how to* ***choose*** *the* ***"best"*** $(\mathbf{c}^\star, b^\star)$ *given a* ***training sample****?*

Linear Classifiers

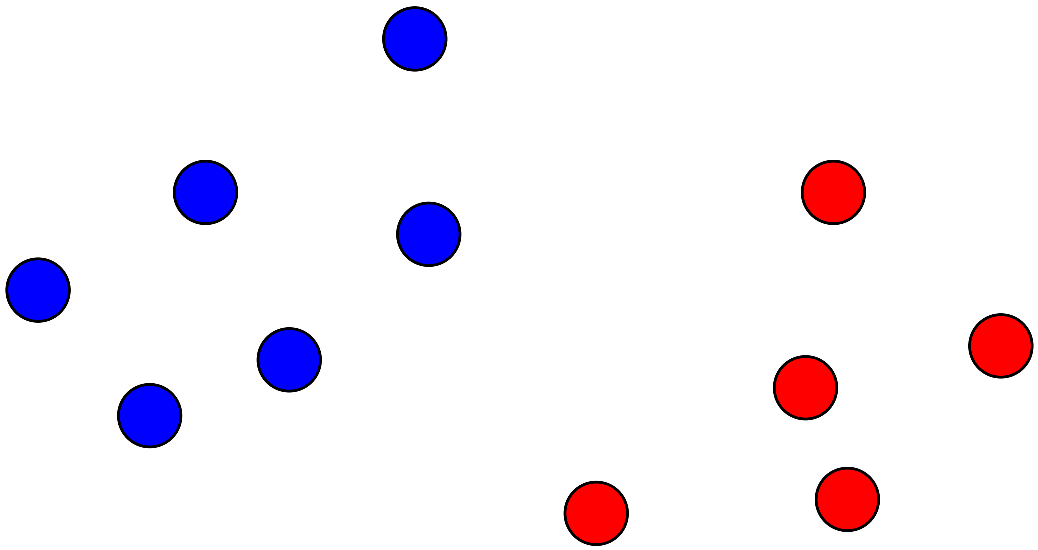
- This specific question,

$$\text{“training set”} \ \left\{\left(\mathbf{x}_i \in \mathbb{R}^d,\ \mathbf{y}_i \in \{0,1\}\right)_{i=1..N}\right\} \overset{????}{\Longrightarrow} \text{“best”}\, \mathbf{c}^\star,\ b^\star$$

  has different answers. Depends on the meaning of “best” **?**:

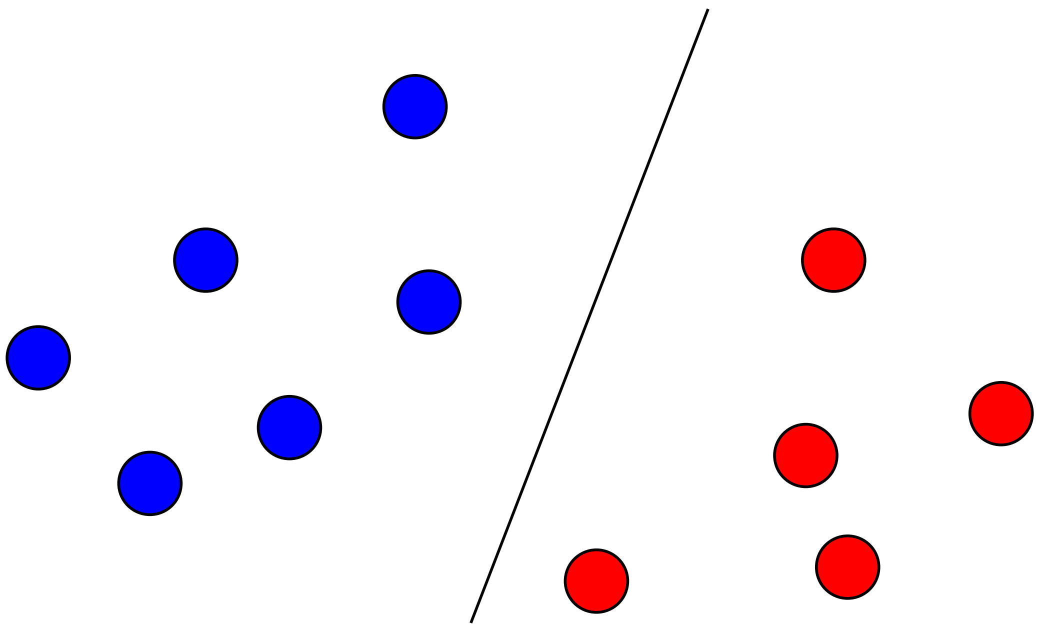
- **Linear Discriminant Analysis** (or Fisher's Linear Discriminant);
- **Logistic regression** maximum likelihood estimation;
- **Perceptron**, a one-layer neural network;
- **Support Vector Machine**, the result of a convex program
- *etc.*

Classification Separation Surfaces for Vectors



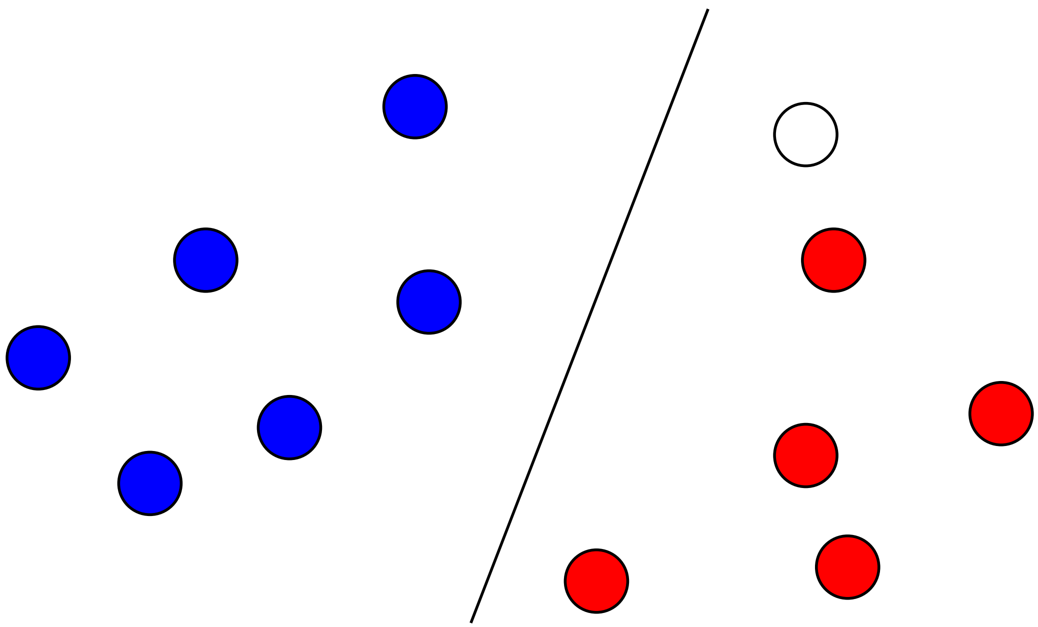
Given two sets of points...

Classification Separation Surfaces for Vectors



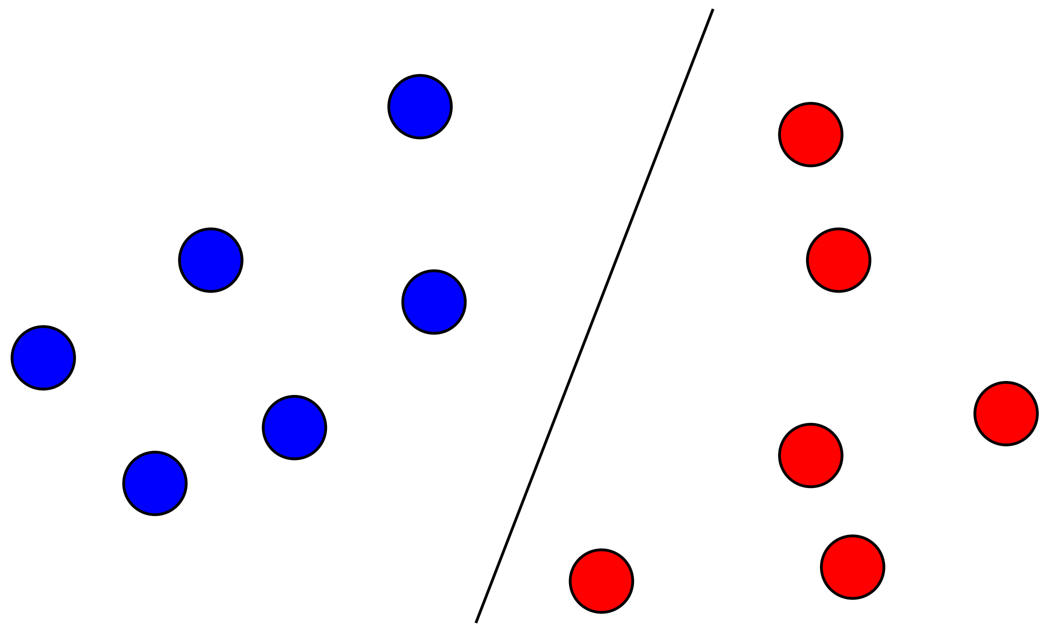
It is sometimes possible to separate them perfectly

Classification Separation Surfaces for Vectors



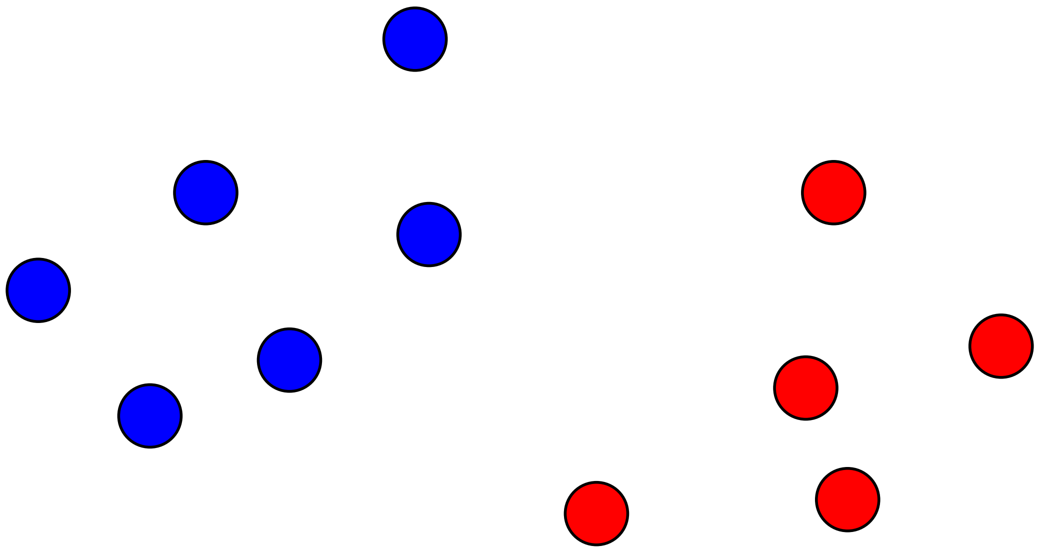
Each choice might look equivalently good on the training set...

Classification Separation Surfaces for Vectors



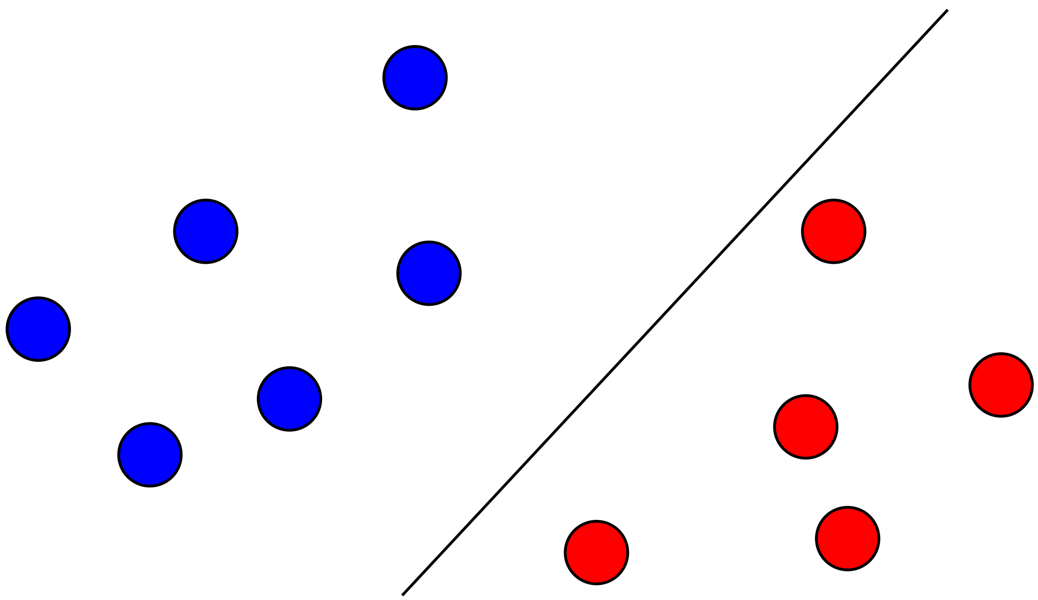
...but it will have obvious impact on new points

Linear classifier, some degrees of freedom



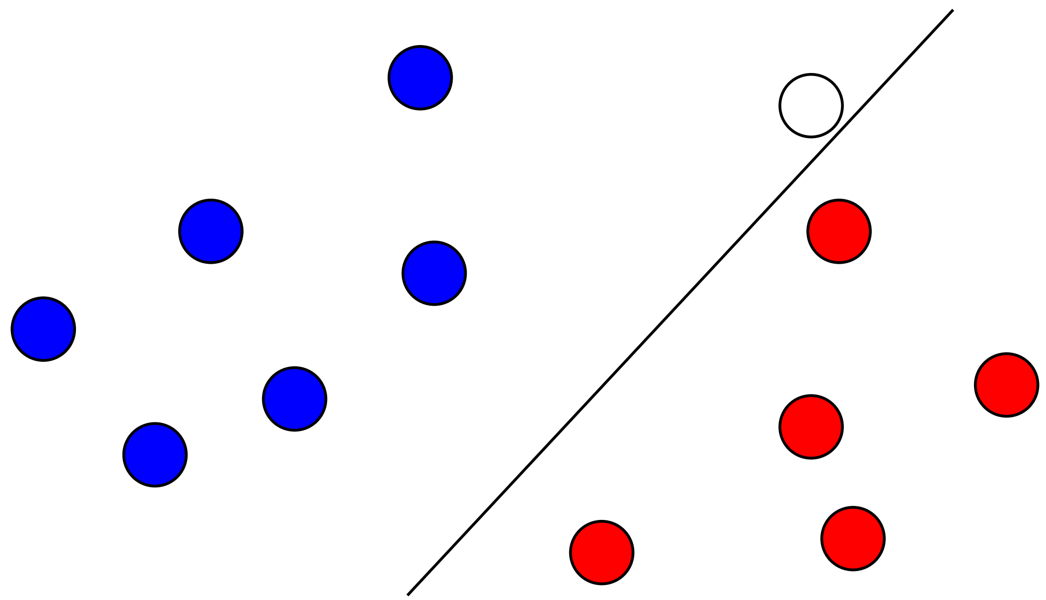
...but it will have obvious impact on new points

Linear classifier, some degrees of freedom



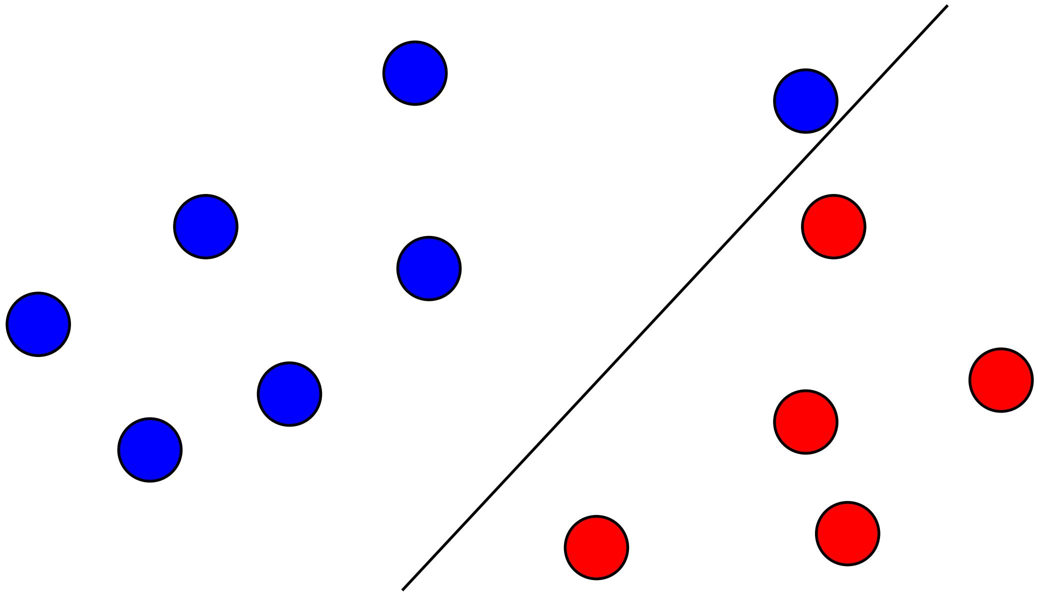
...but it will have obvious impact on new points

Linear classifier, some degrees of freedom



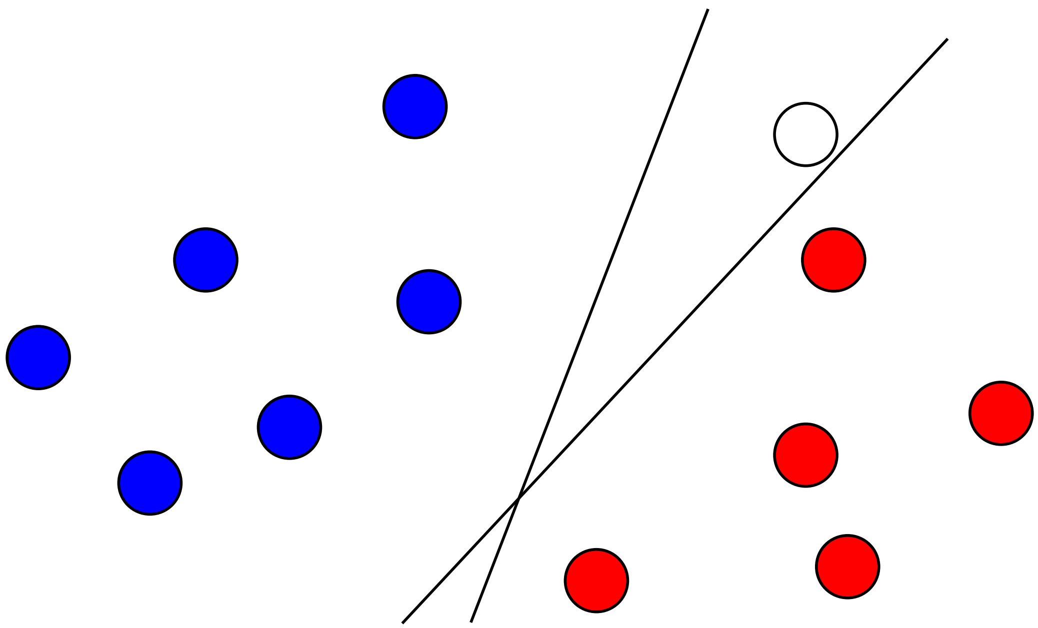
Specially close to the border of the classifier

Linear classifier, some degrees of freedom



Specially close to the border of the classifier

Linear classifier, some degrees of freedom



For each hyperplane, different results, different performance.

Perceptron: an iterative algorithm to compute $\mathbf{c}$ and b

- Iterative algorithm that considers each point successively.
- Here we consider $\mathcal{S} = \{-1, 1\}$
- Start from any arbitrary estimate $\omega = \left[\begin{smallmatrix} b \\ \mathbf{c} \end{smallmatrix}\right]$.
- Loop over j until ω does not change for a while...
  - ○ Consider a point $\left[\begin{smallmatrix} 1 \\ \mathbf{x}_j \end{smallmatrix}\right]$ and his label y_j.
  - ○ Do $u_j = \text{sign}(\omega^T \left[\begin{smallmatrix} 1 \\ \mathbf{x}_j \end{smallmatrix}\right])$ and y_j match?
  - ○ it not, set $\omega \leftarrow \omega + \rho(y_j - u_j) \left[\begin{smallmatrix} 1 \\ \mathbf{x}_j \end{smallmatrix}\right]$.
- Not much more to add, better see in practice.