Understanding Machine Learning: From Geometry and Probability to Learning Algorithms

AI for Sciences
Joint Winter School - NTU Singapore - Sorbonne Université

Nicolas Baskiotis
nicolas.baskiotis@sorbonne-universite.fr

MLIA Team, ISIR
Sorbonne Université

January 2026

Outline

Outline

What is Machine Learning?

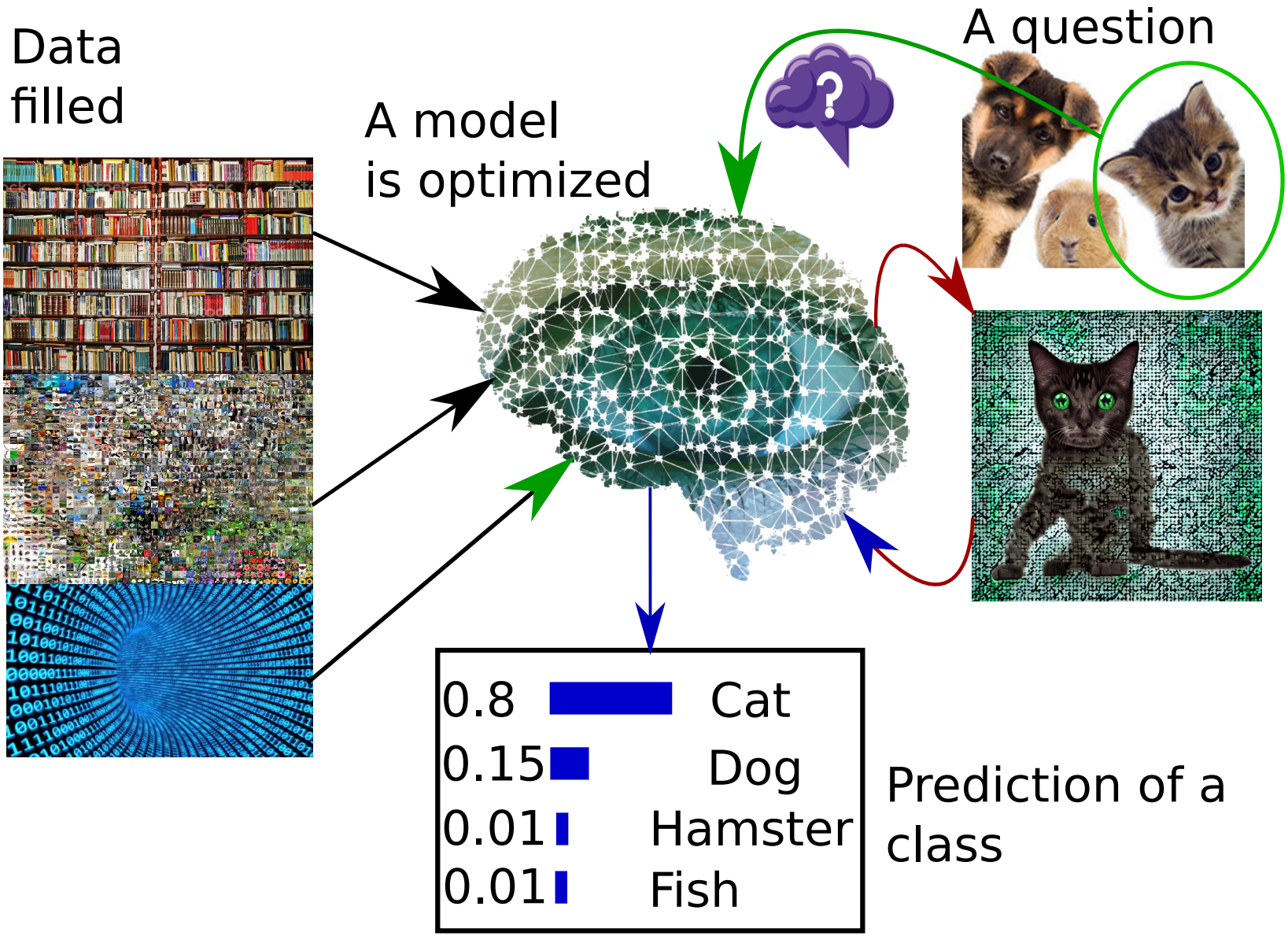
In a few words

- Finding **structures** and **regularities** in observations
- **Predicting** new observations
- Algorithms that **improve** their performance on a task based on experience

Three main families

- **Supervised Learning**: Classification, Regression, Ranking
- **Unsupervised Learning**: Clustering, Representation Learning
- **Reinforcement Learning**: Learning to interact with an environment

The Machine Learning Pipeline



Example: Fruit Classification

We want to recognize which fruit it is:

- apple
- orange
- lemon
- mandarin

We have:

- many examples: apples, oranges, lemons, mandarins of all sizes/colors
- and their labels: which fruit it is

To do this, humans have labeled each fruit in the dataset with its name.

First Step: From Objects to Feature Space

A fruit cannot be directly given to the computer

We need to be able to represent it in terms of data

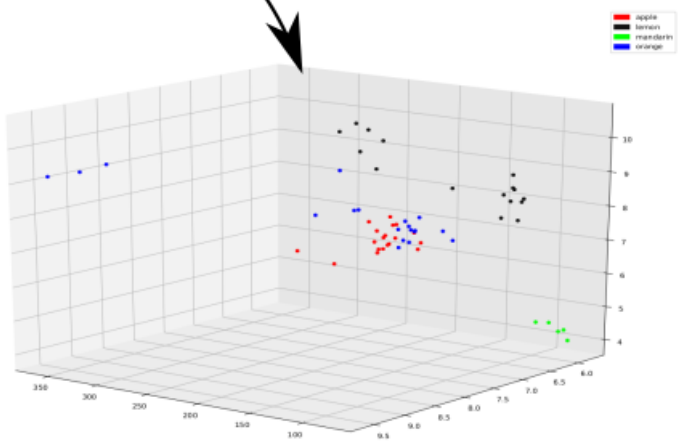
First Step: From Objects to Feature Space

A fruit cannot be directly given to the computer

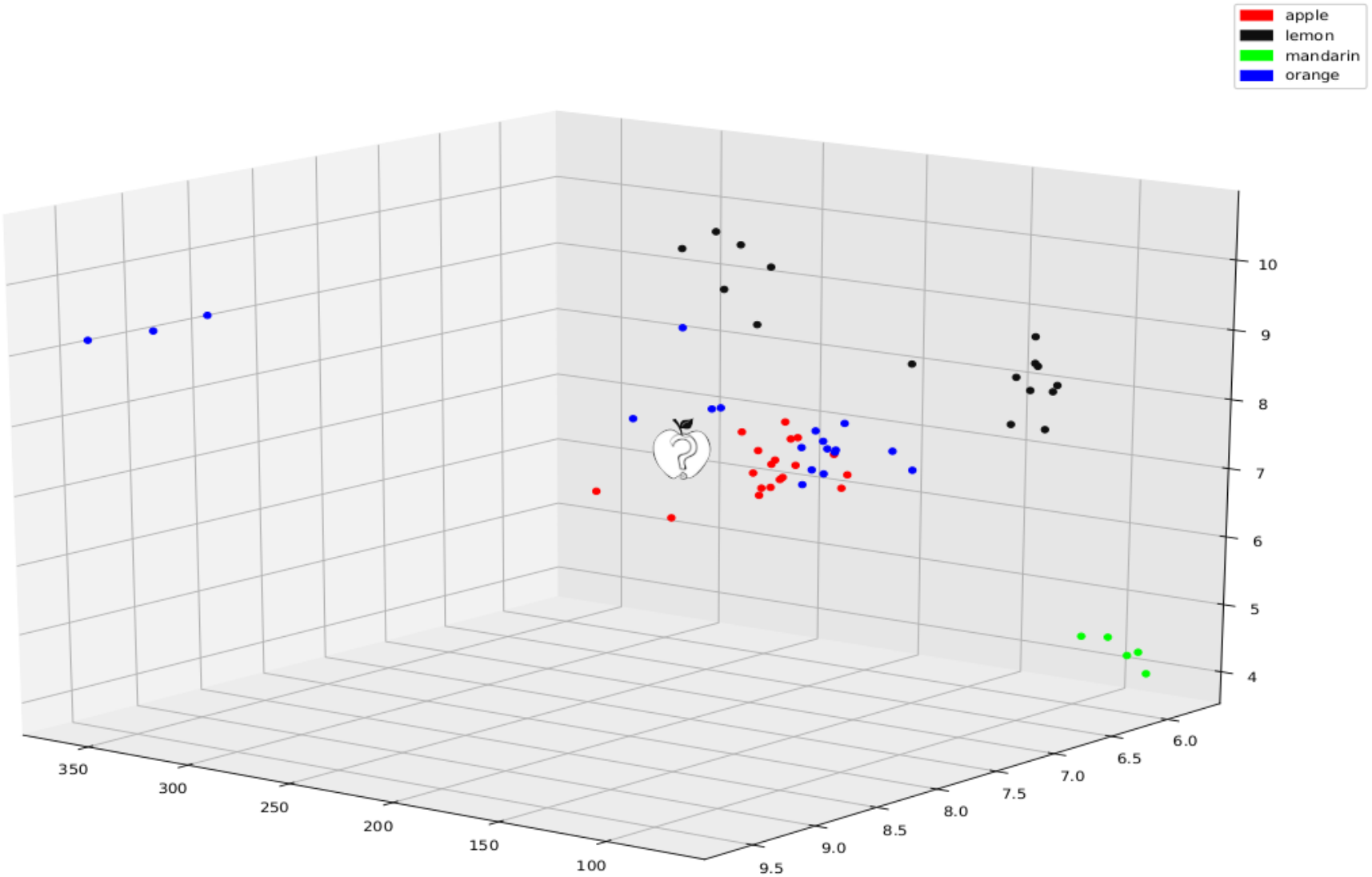
But can be described by features

Which allows to see it as a point in space

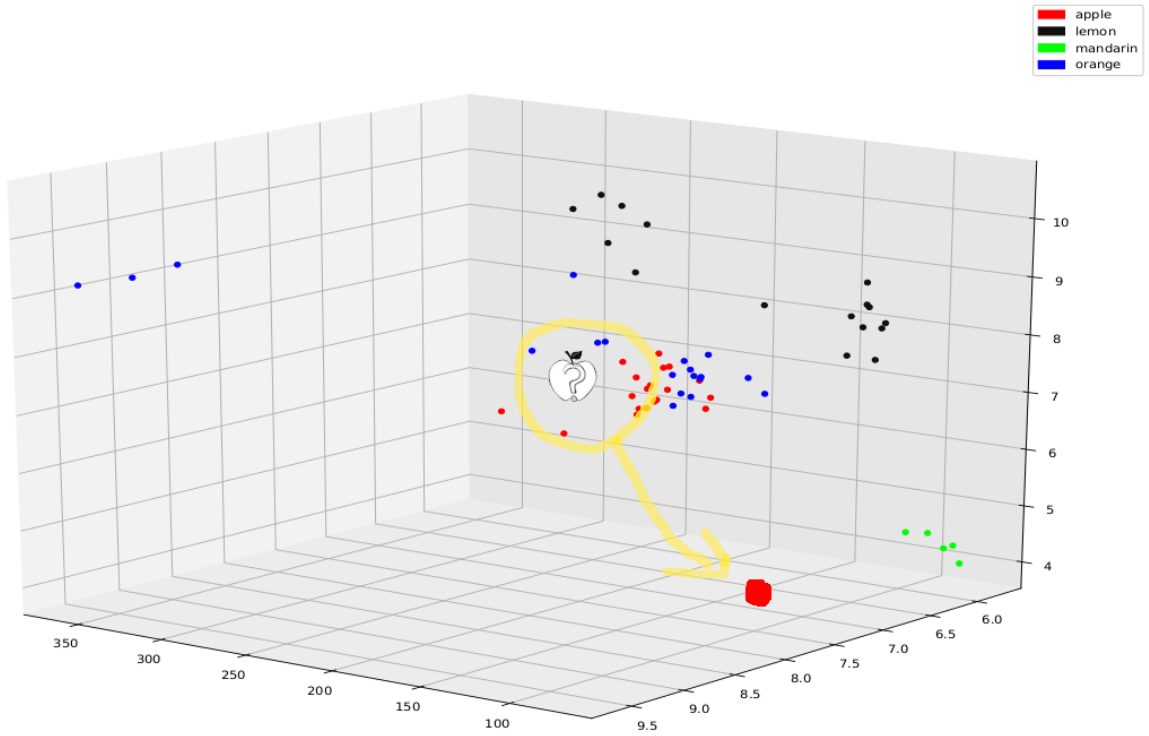
	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89
10	1	apple	braeburn	166	6.9	7.3	0.93
11	1	apple	braeburn	172	7.1	7.6	0.92
12	1	apple	braeburn	154	7.0	7.1	0.88
13	1	apple	golden_delicious	164	7.3	7.7	0.70
14	1	apple	golden_delicious	152	7.6	7.3	0.69
15	1	apple	golden_delicious	156	7.7	7.1	0.69
16	1	apple	golden_delicious	156	7.6	7.5	0.67
17	1	apple	golden_delicious	168	7.5	7.6	0.73
18	1	apple	cripps_pink	162	7.5	7.1	0.83
19	1	apple	cripps_pink	162	7.4	7.2	0.85
20	1	apple	cripps_pink	160	7.5	7.5	0.86
21	1	apple	cripps_pink	156	7.4	7.4	0.84
22	1	apple	cripps_pink	140	7.3	7.1	0.87
23	1	apple	cripps_pink	170	7.6	7.9	0.88
24	3	orange	spanish_jumbo	342	9.0	9.4	0.75
25	3	orange	spanish_jumbo	356	9.2	9.2	0.75
26	3	orange	spanish_jumbo	362	9.6	9.2	0.74
27	3	orange	selected_seconds	204	7.5	9.2	0.77
28	3	orange	selected_seconds	140	6.7	7.1	0.72
29	3	orange	selected_seconds	160	7.0	7.4	0.81
30	3	orange	selected_seconds	158	7.1	7.5	0.79
31	3	orange	selected_seconds	210	7.8	8.0	0.82
32	3	orange	selected_seconds	164	7.2	7.0	0.80



Second Step: Choosing a Model



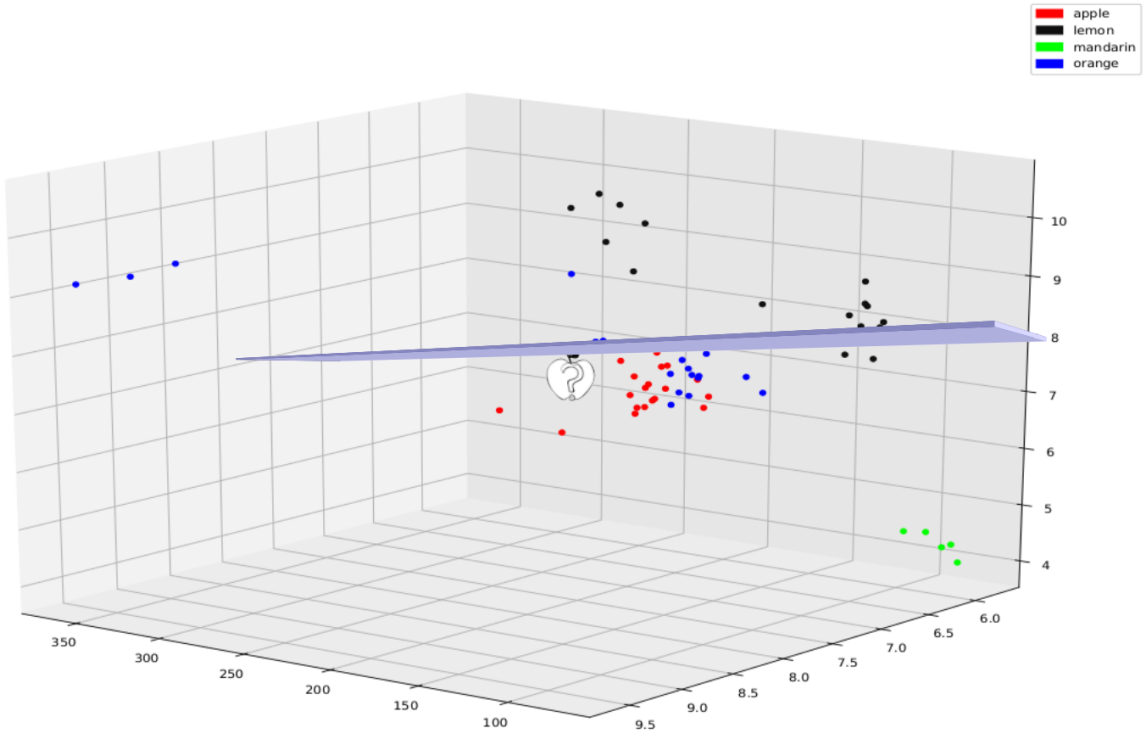
Naive Model: K-Nearest Neighbors



Principle:

- An example is classified based on its nearest neighbors
- Works well but is computationally expensive
- No model is learned; all data must be stored

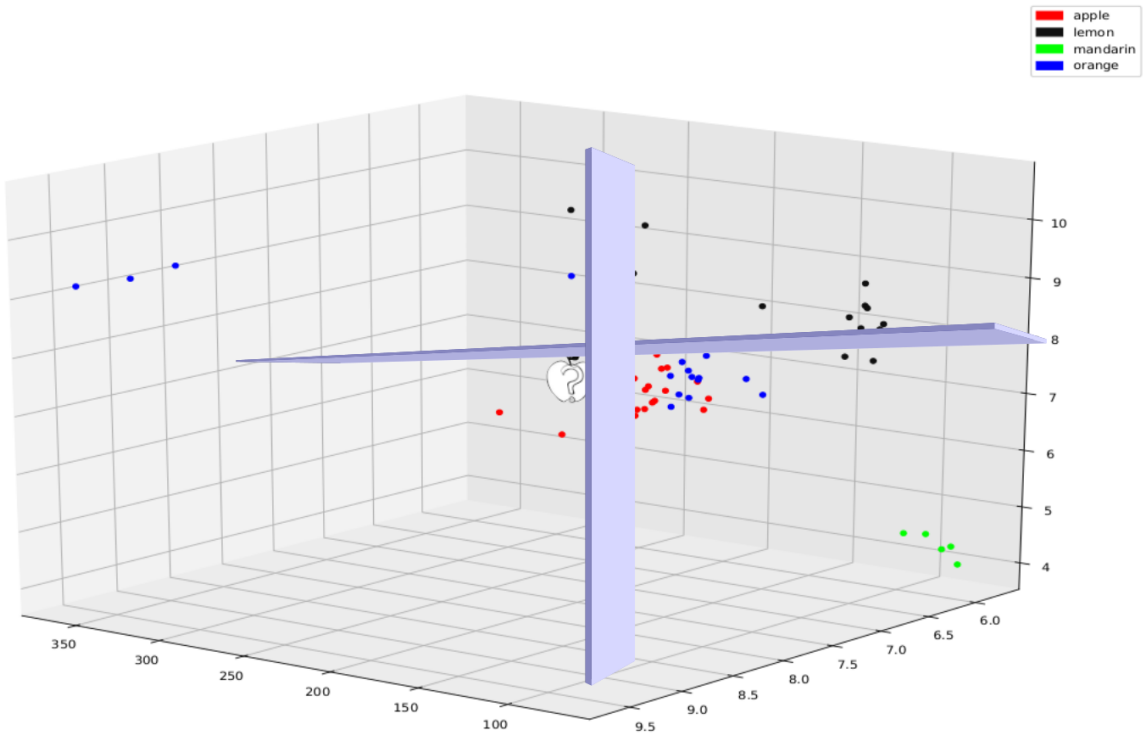
More Complex Model:



Find functions that best separate the fruits

- From the simplest: a line, a plane

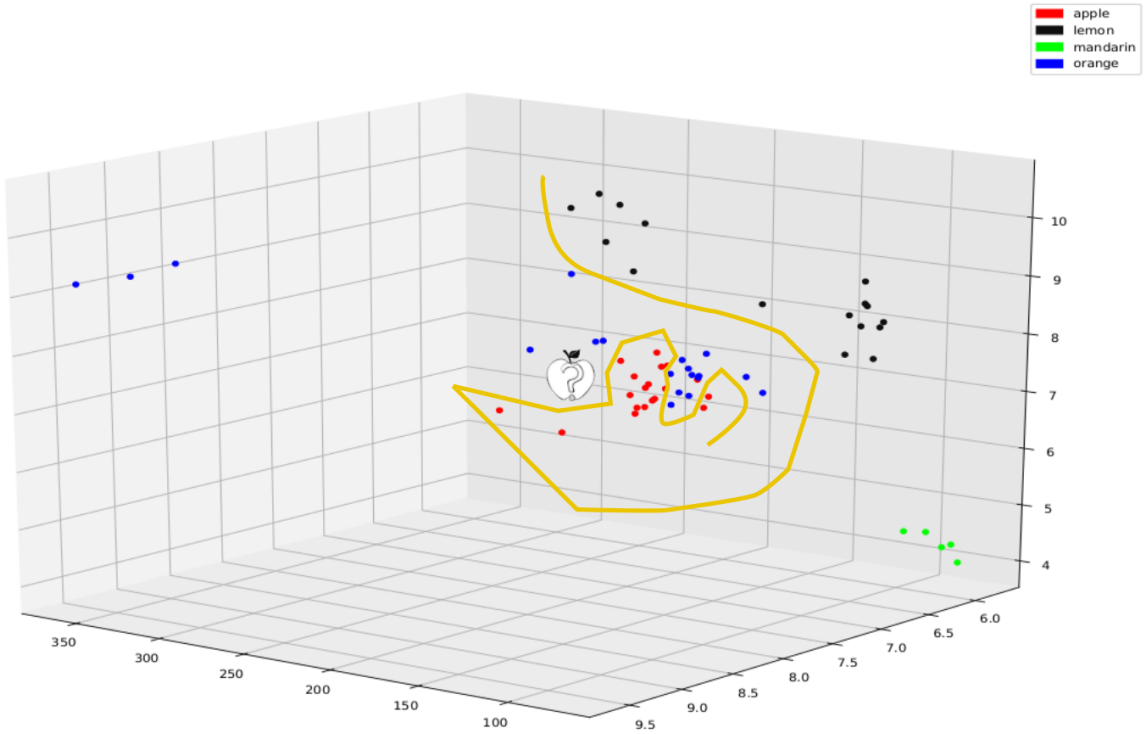
More Complex Model:



Find functions that best separate the fruits

- From the simplest: a line, a plane
- To very complex functions with many parameters

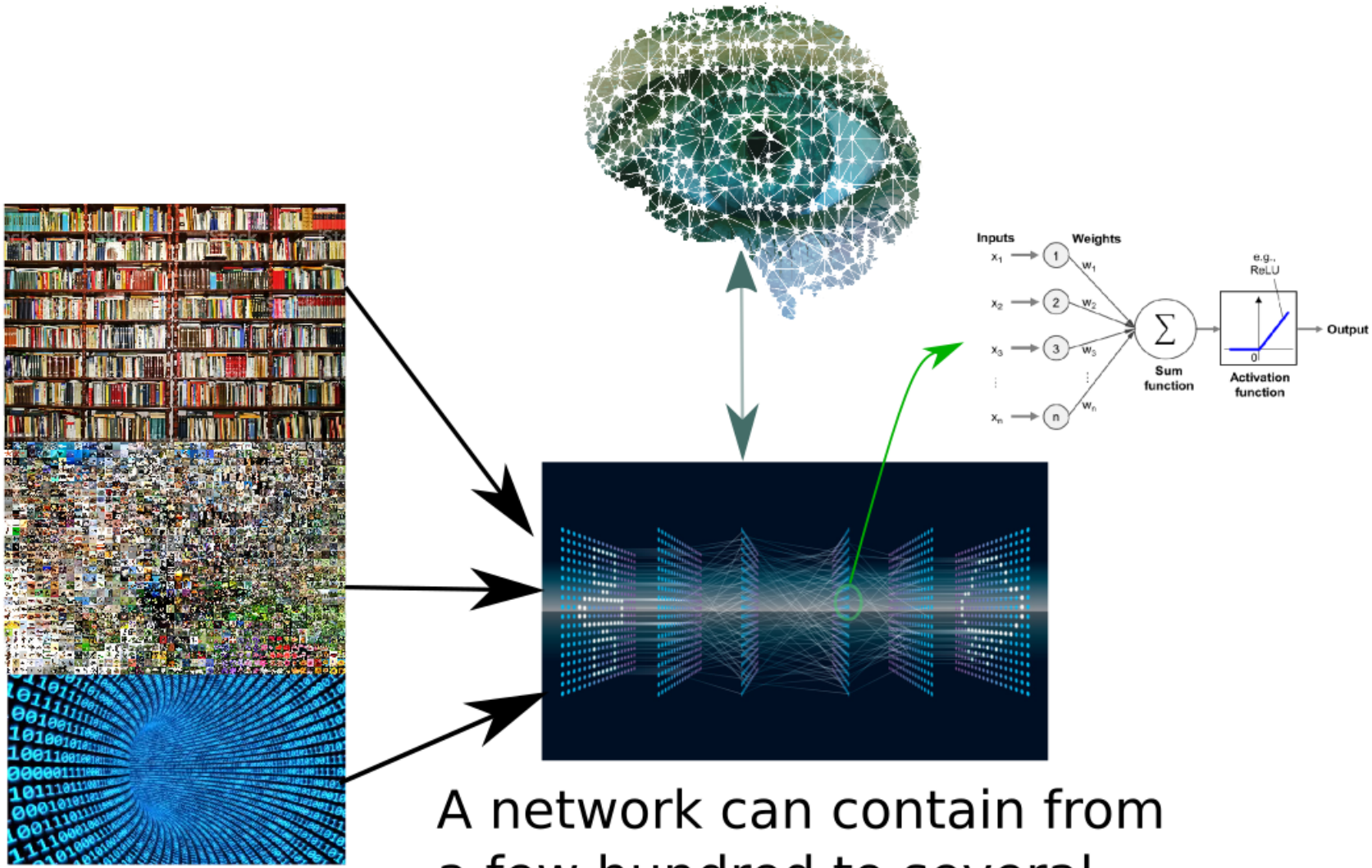
More Complex Model:



Find functions that best separate the fruits

- From the simplest: a line, a plane
- To very complex functions with many parameters
- Goal: find the right parameters for the functions $\Rightarrow$ **Learn** from the data

A Neural Network (or Deep Learning Model)



A network can contain from a few hundred to several billion parameters

Outline

Back to usual Binary Classification

Formalization

- Two classes : $\mathcal{Y} = \{y_+, y_-\}$
- A set $\mathcal{X} \subseteq \mathbb{R}^d$ of examples, with d the dimension of the representation
- An example : $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in \mathcal{X}$
- Goal: Output the class of a new example $\mathbf{x} \in \mathcal{X}$

$\Rightarrow$ we look for a function $f : \mathcal{X} \to \mathcal{Y}$ (the classifier)

- Notation : $\hat{y}$ the predicted class of an example $\mathbf{x}$, $\hat{y} = f(\mathbf{x})$

First Example : Classification of Movies (IMDB Database)

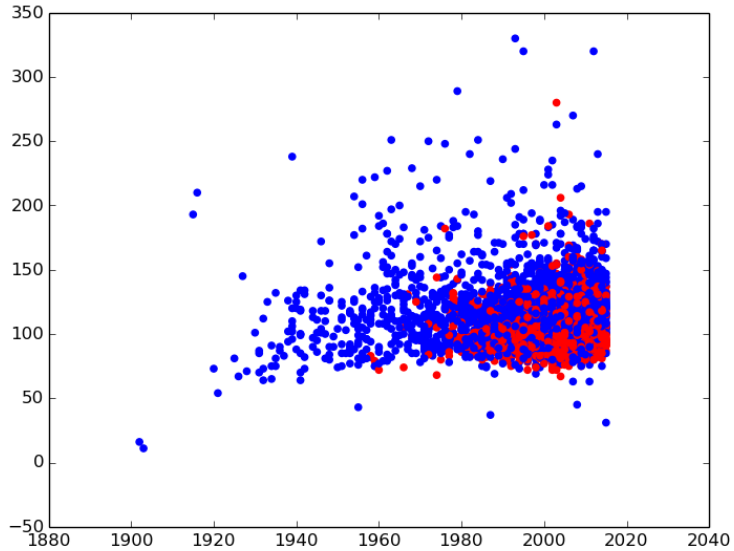
- Two classes : I like (y_+) and I don't like (y_-)
- Simple description of a movie : (year, budget, duration) (3 dimensions, $\mathcal{X} = \mathbb{R}^3$)
- Goal : a classifier such as $f(\mathbf{x}) = y_+$ if $x_1 \geq 2000$ else y_-

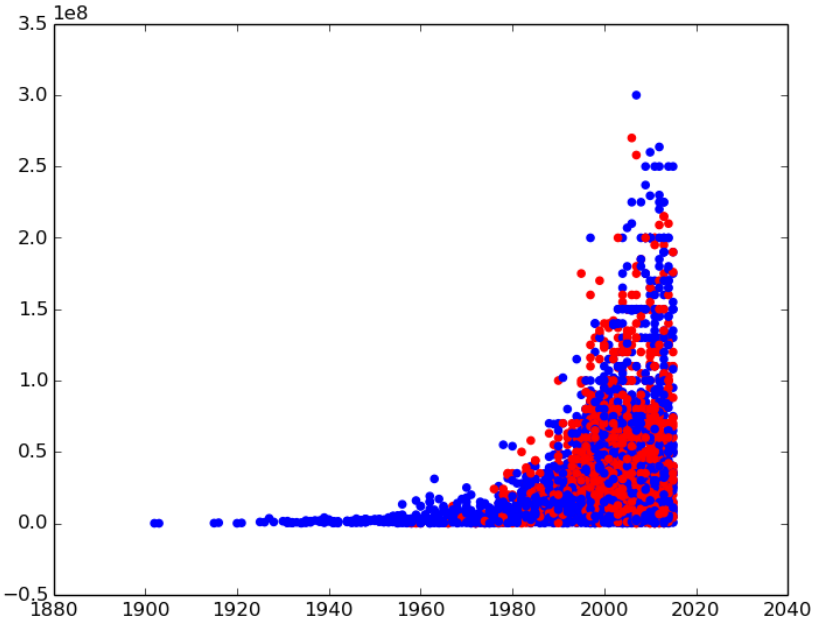
Binary Classification : IMDB

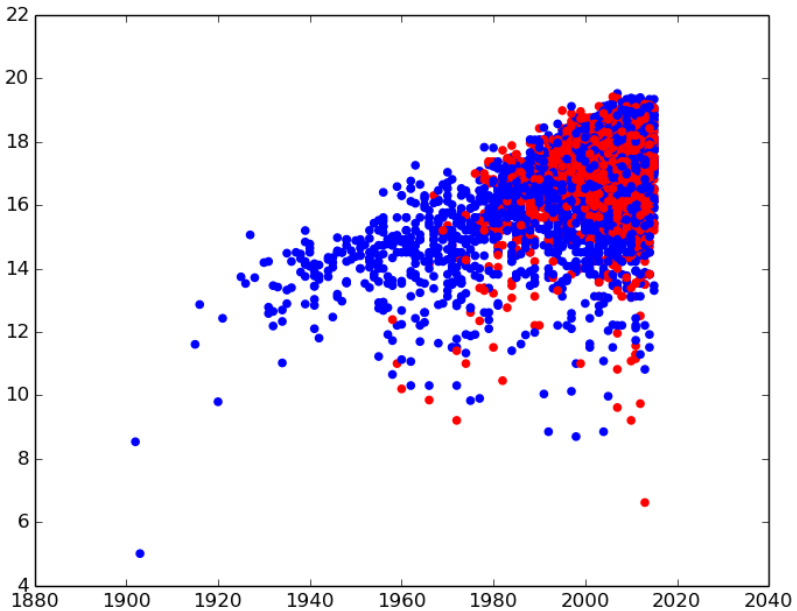
Plot of movies . . . :

- Year vs Duration
- Year vs Budget

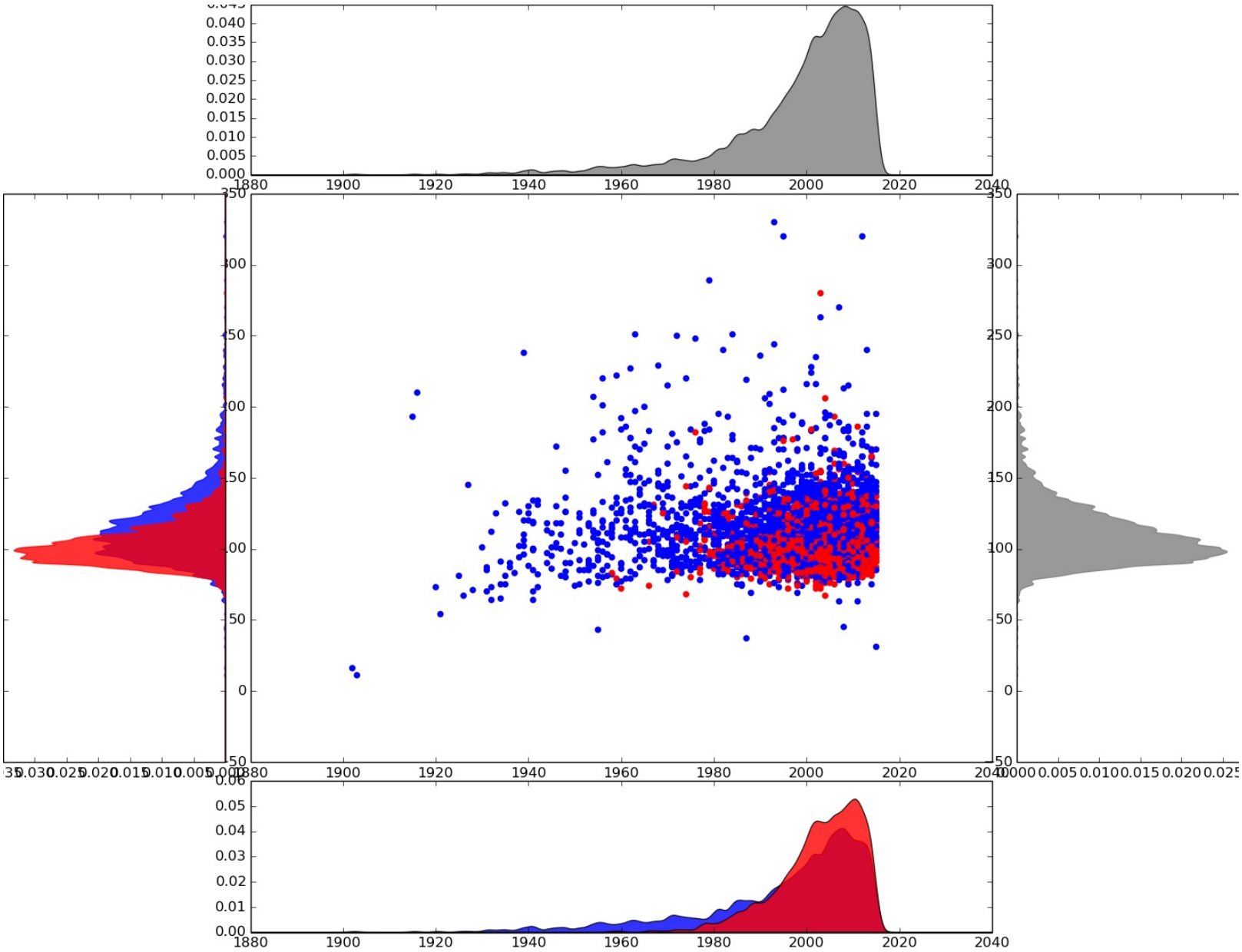
What would *you* do to classify?







IMDB Example : what are $p(\mathbf{x})$, $p(y)$, $p(\mathbf{x}|y)$, $p(y|\mathbf{x})$?



In an ideal world (bayesian)

If we have . . .

$P(y)$ (*a priori* probability) and $p(\mathbf{x}|y)$:

- $p(y, \mathbf{x}) = p(y|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|y)p(y)$
- $p(\mathbf{x}) = p(\mathbf{x}|y_+)p(y_+) + p(\mathbf{x}|y_-)p(y_-)$
- $p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x}|y_+)P(y_+)+p(\mathbf{x}|y_-)P(y_-)}$

In an ideal world (bayesian)

If we have ...

$P(y)$ (*a priori* probability) and $p(\mathbf{x}|y)$:

- $p(y, \mathbf{x}) = p(y|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|y)p(y)$
- $p(\mathbf{x}) = p(\mathbf{x}|y_+)p(y_+) + p(\mathbf{x}|y_-)p(y_-)$
- $p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x}|y_+)P(y_+)+p(\mathbf{x}|y_-)P(y_-)}$

Then

By looking at $\mathbf{x}$, we can study the *a posteriori* probability $p(y|\mathbf{x})$.

- $p(\mathbf{x}|y)$ is the likelihood of $\mathbf{x}$ wrt y.
- Bayesian decision : choose y_+ if $p(y_+|\mathbf{x}) > p(y_-|\mathbf{x})$, the opposite otherwise

$\Rightarrow f(\mathbf{x}) = \text{argmax}_y\, p(y|\mathbf{x}) = \text{argmax}_y\, \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$

- Is $p(\mathbf{x})$ important?

In an ideal world (bayesian)

If we have ...

$P(y)$ (*a priori* probability) and $p(\mathbf{x}|y)$:

- $p(y, \mathbf{x}) = p(y|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|y)p(y)$
- $p(\mathbf{x}) = p(\mathbf{x}|y_+)p(y_+) + p(\mathbf{x}|y_-)p(y_-)$
- $p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x}|y_+)P(y_+)+p(\mathbf{x}|y_-)P(y_-)}$

Then

By looking at $\mathbf{x}$, we can study the *a posteriori* probability $p(y|\mathbf{x})$.

- $p(\mathbf{x}|y)$ is the likelihood of $\mathbf{x}$ wrt y.
- Bayesian decision : choose y_+ if $p(y_+|\mathbf{x}) > p(y_-|\mathbf{x})$, the opposite otherwise

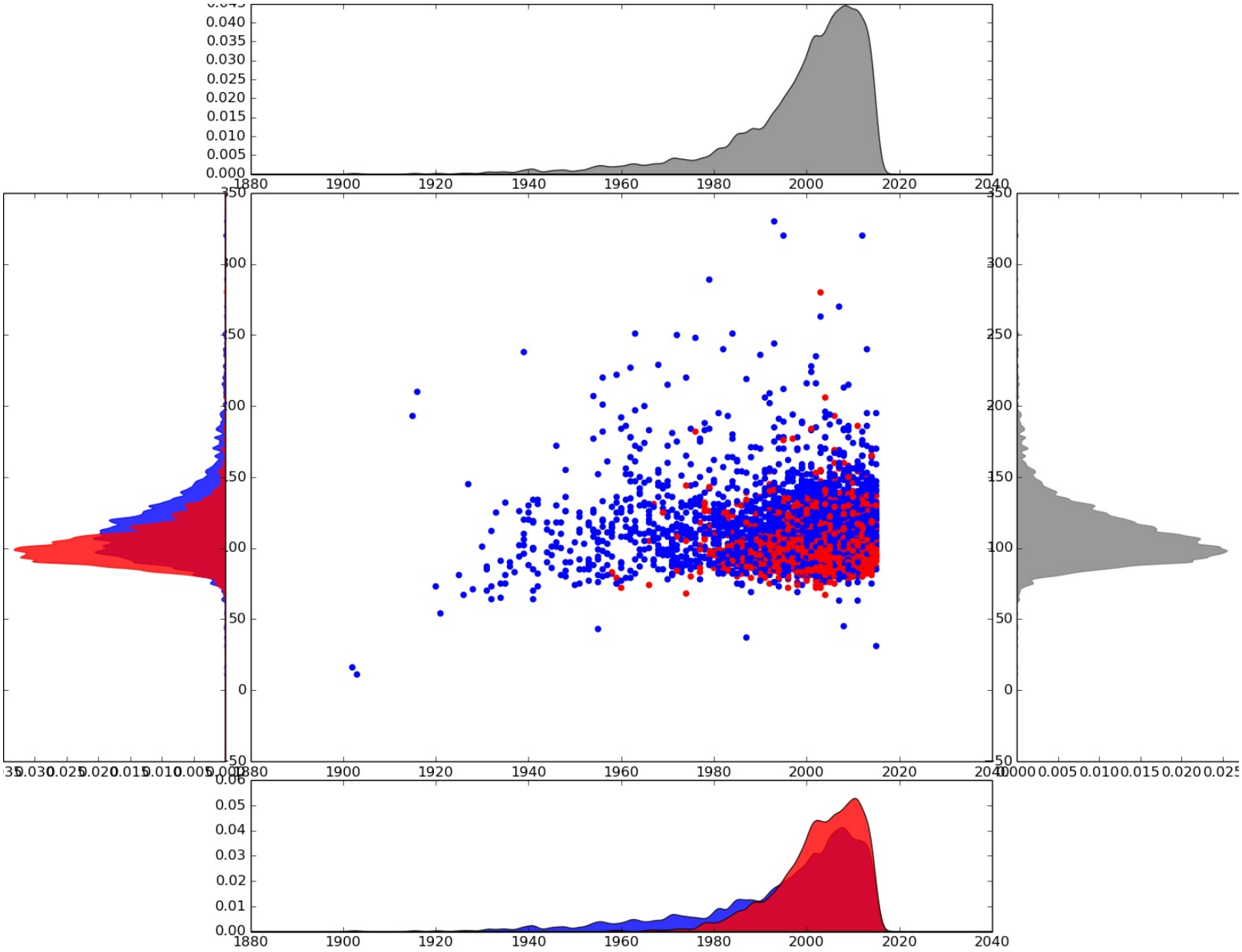
$\Rightarrow f(\mathbf{x}) = \text{argmax}_y\, p(y|\mathbf{x}) = \text{argmax}_y\, \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$

- Is $p(\mathbf{x})$ important?

So it is over ?

Obviously not, $p(\mathbf{x}|y)$ is rarely available ...

IMDB Example

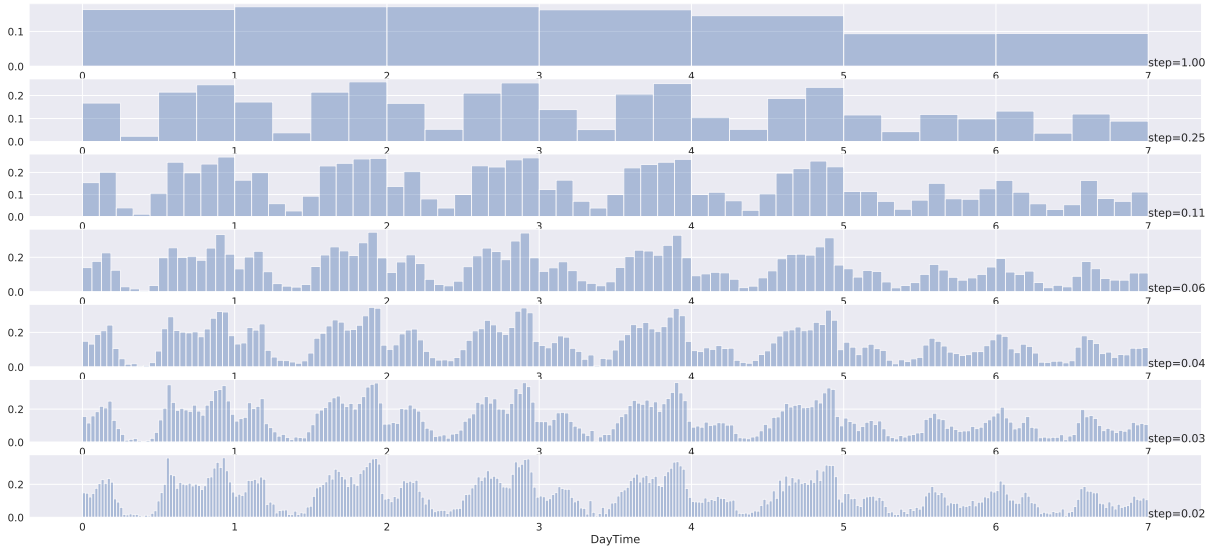


Density Estimation: From Data to Probability

We want to estimate the density of the continuous random variable h.

Histogram :Estimation with a discrete random variable H

- The continuous values of the random variable are first discretized with a collection $\{T_j\}$ of intervals $\Rightarrow$ selection of a discretization step
- The histogram method provides us an estimation of a discrete variable H:
  $P(H \in T_j) = \frac{|\{\mathbf{x}_i | x_i^t \in T_j\}|}{N}$
- The associated probability density is then : $p(h \in T_j) = \frac{P(H \in T_j)}{\Delta^d}$ (d the dimension).
- The discretization step is very important : smaller $\rightarrow$ over-fitting, higher $\rightarrow$ under-fitting

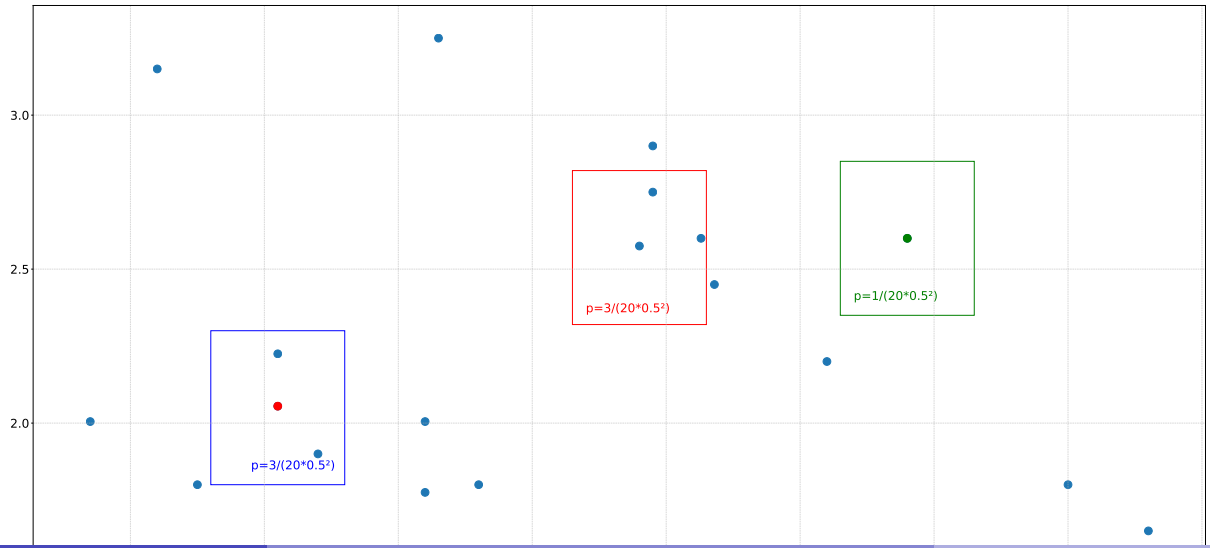


Continuous approximation : Kernel Density Estimation

Intuition : histogram centered at the estimation point

- Rather to decide the a priori discretization, the estimation is done by centering a window at the point of interest $\mathbf{x}_0$
- Let $\mathcal{R}_{\mathbf{x}_0}$ be the hypercube centered at $\mathbf{x}_0$ with a lenght r and $p(\mathbf{x}_0)$ the (constant) density to estimate
- Discrete probability of a point being in the hypercube : $P_{\mathcal{R}_{\mathbf{x}_0}} = \int_{\mathcal{R}_{\mathbf{x}_0}} p(\mathbf{x}_0) d\mathbf{x} = r^d p(\mathbf{x}_0)$

$\Rightarrow p(\mathbf{x}_0) = \frac{P_{\mathcal{R}_{\mathbf{x}_0}}}{r^d}$



Generalization : Parzen Window

For a sample of size N

- $\mathcal{R}$ is a hypercube, each side of length r
- $V = r^d$, d the dimension of the representation space
- $\phi(\mathbf{x}) = \begin{cases} 1 & \text{if } |x^i| \leq 1/2 \\ 0 & \text{otherwise} \end{cases}$ Indicator function of the unit hypercube
- ϕ defines a unit hypercube centered at the origin.

$\Rightarrow$ $\phi(\frac{\mathbf{x}_0 - x}{r}) = 1$ iff $\mathbf{x}$ is inside the hypercube of volume V centered at $\mathbf{x}_0$.

Consequently

- Number of samples inside the hypercube: $k = \sum_{i=1}^{N} \phi(\frac{(\mathbf{x}_0 - \mathbf{x}_i)}{r})$
- Estimate density:

$$p(\mathbf{x}_0) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{V} \phi(\frac{\mathbf{x}_0 - \mathbf{x}_i}{r})$$

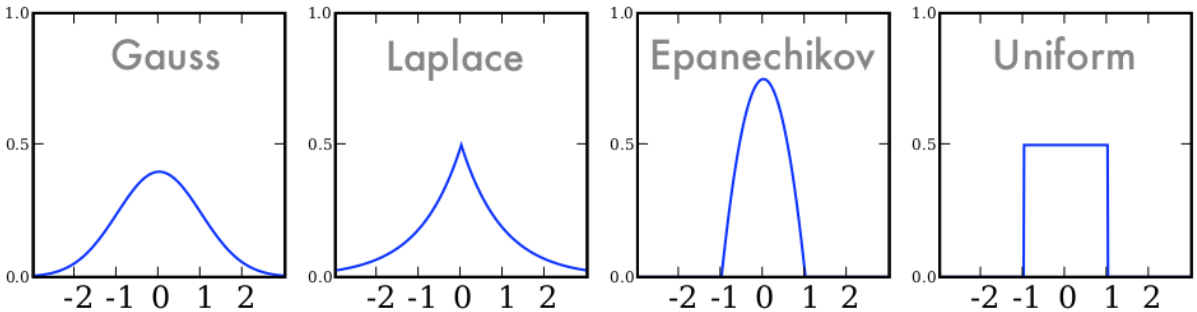
- With $\delta(\mathbf{x}) = \frac{1}{V}\phi(\frac{\mathbf{x}}{r})$, then

$$p(\mathbf{x}_0) = \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{x}_0 - \mathbf{x}_i)$$

Discussion

Beyond hypercubes

- ϕ can be more generic (kernel)
- can weight samples wrt the distance of the estimation point
- Necessary conditions:
  - $\phi(x) \geq 0$
  - $\int \phi(x)dx = 1$
  - $\phi(x)$ symmetric
  - $\phi(x)$ maximal at 0 and doubly monotonic.



$(2\pi)^{-\frac{1}{2}} e^{-\frac{1}{2}x^2}$ $\frac{1}{2}e^{-|x|}$ $\frac{3}{4}\max(0, 1-x^2)$ $\frac{1}{2}\chi_{[-1,1]}(x)$

From density estimation to classification

Nadaraya-Watson estimate

Let

- a label y_i for each $\mathbf{x}_i$, $y_i \in \{-1, 1\}$
- n_+ the number of positive examples, n_- the number of negative examples.
- a kernel δ for the density estimation.

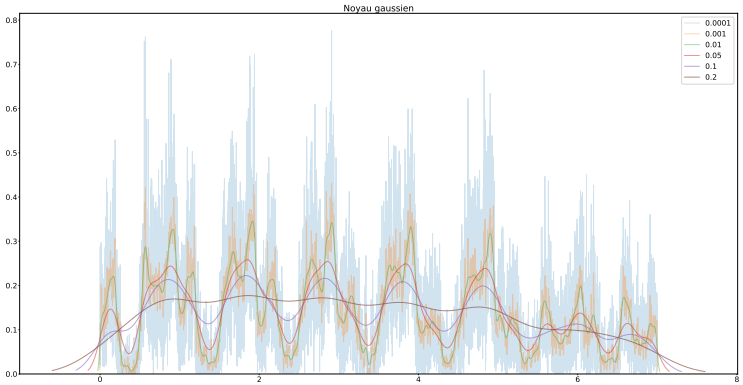
The goal of binary classification is to estimate $p(\mathbf{x}|y=1)$ and $p(\mathbf{x}|y=-1)$

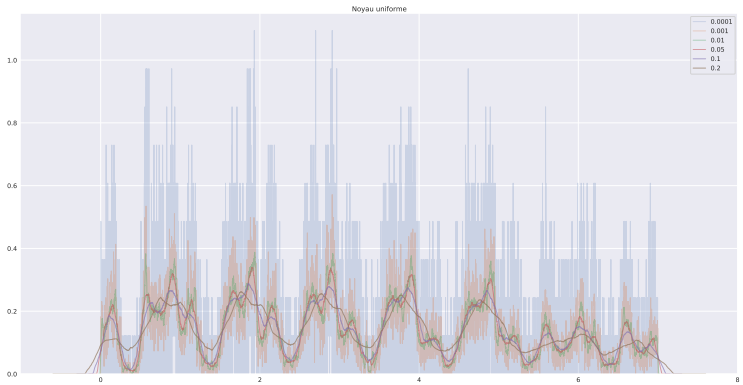
- $p(\mathbf{x}|y=1) = \frac{1}{n_+} \sum_{i|y_i=1} \delta(\mathbf{x}-\mathbf{x}_i)$, $p(\mathbf{x}|y=-1) = \frac{1}{n_-} \sum_{i|y_i=-1} \delta(\mathbf{x}-\mathbf{x}_i)$
- $p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} = \frac{\frac{1}{n_y} \sum_{i|y_i=y} \delta(\mathbf{x}-\mathbf{x}_i) \frac{n_y}{n}}{\frac{1}{n} \sum_i \delta(\mathbf{x}-\mathbf{x}_i)} = \frac{\sum_{i|y_i=y} \delta(\mathbf{x}-\mathbf{x}_i)}{\sum_i \delta(\mathbf{x}-\mathbf{x}_i)}$

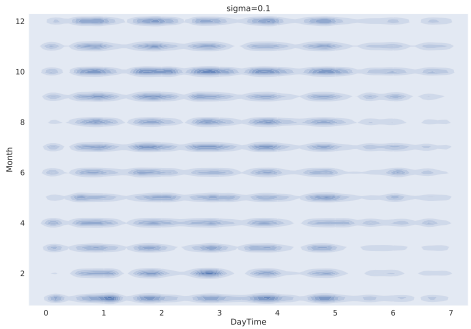
$\Rightarrow p(y_+|\mathbf{x}) - p(y_-|\mathbf{x}) = \frac{\sum_{j=1}^N y_j \delta(\mathbf{x}-\mathbf{x}_j)}{\sum_{i=1}^N \delta(\mathbf{x}-\mathbf{x}_i)} = \frac{1}{\sum_{i=1}^N \delta(\mathbf{x}-\mathbf{x}_i)} \sum_{j=1}^N y_j \delta(\mathbf{x}-\mathbf{x}_j)$

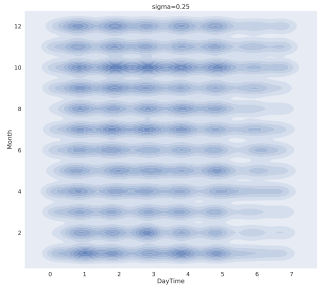
- It is more generic than just classification ! If y is continuous, this approach solves the regression problem
- Intuition : local averaging around the point of interest, with a weighting depending on the distance.

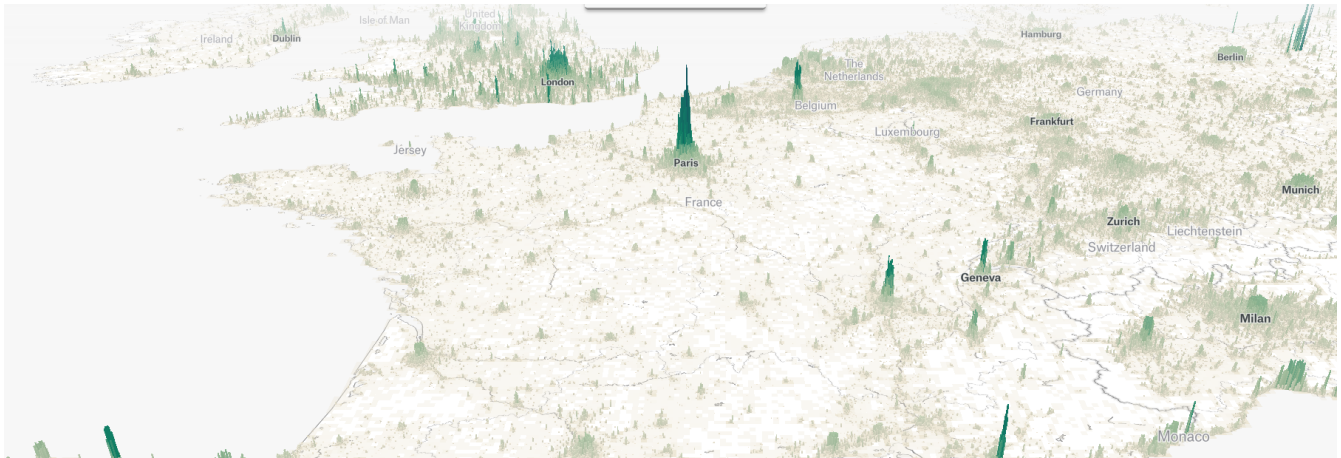
From density estimation to classification









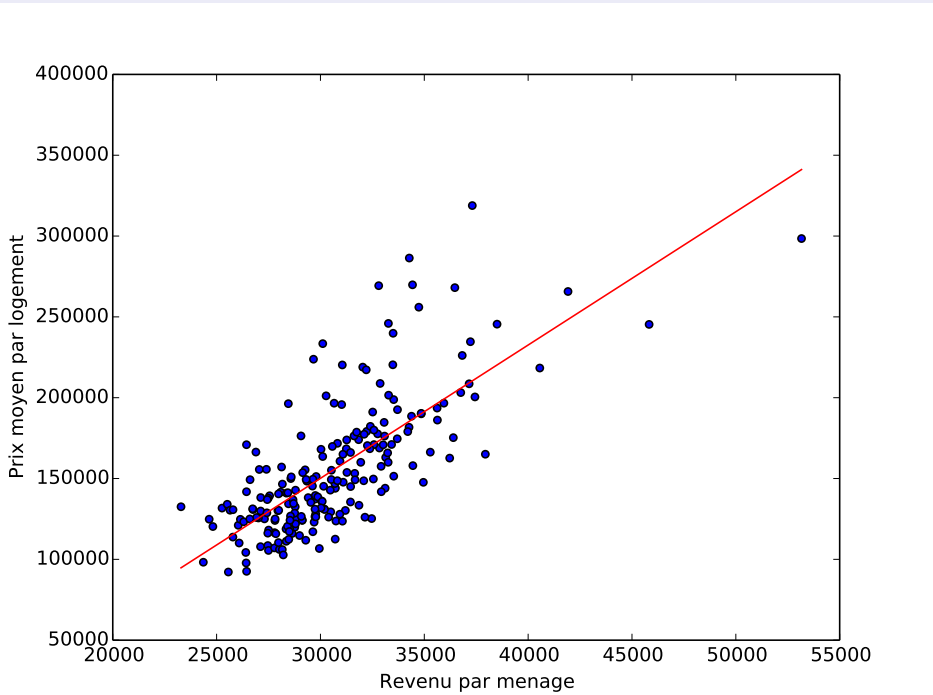


Outline

Introduction

Linear Regression

- Goal: predict a continuous real output y from a number of input variables
- Many applications, widely used in all fields
- Very flexible (input transformations)



Formalization

Goal

Given a dataset $\{(\mathbf{x}^j, y^j) \in \mathbb{R}^d \times \mathbb{R}\}_{j=1}^N$, $\mathbf{x}^j = (x_1^j, x_2^j, \ldots, x_d^j)$

- Hypothesis: linear variation of output with respect to inputs

$$\mathbb{E}[y|\mathbf{x}] = w_0 + \sum_{i=1}^{d} w_i x_i$$

$\Rightarrow$ We are looking for:

  - a function $f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i$
  - that minimizes errors: $f(\mathbf{x}^i)$ should be close to y^i
  - under the condition that error is independent of $\mathbf{x}$, with constant variance σ^2, following a normal distribution.

  $\Rightarrow$ $y|\mathbf{x} \sim \mathcal{N}(f(\mathbf{x}), \sigma^2)$ (link with Bayesian learning)

- Error measure: least squares cost (Mean Squared Error - MSE)

$$\ell(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$$

Logistic Regression

Binary Classification

- Two classes: $Y = \{-1, +1\}$, and a training set $\{(\mathbf{x}^i, y^i) \in \mathbb{R}^d \times Y\}$
- Can we use a quadratic cost in this case?
  2D case:
  - $f_{\mathbf{w}}(\mathbf{x}) = 1 \Leftrightarrow w_0 + x_1 w_1 + x_2 w_2 = 1$
  - $f_{\mathbf{w}}(\mathbf{x}) = -1 \Leftrightarrow w_0 + x_1 w_1 + x_2 w_2 = -1$

Logistic Regression

Binary Classification

- Two classes: $Y = \{-1, +1\}$, and a training set $\{(\mathbf{x}^i, y^i) \in \mathbb{R}^d \times Y\}$
- Can we use a quadratic cost in this case?
  2D case:
  - $f_{\mathbf{w}}(\mathbf{x}) = 1 \Leftrightarrow w_0 + x_1 w_1 + x_2 w_2 = 1$
  - $f_{\mathbf{w}}(\mathbf{x}) = -1 \Leftrightarrow w_0 + x_1 w_1 + x_2 w_2 = -1$
- What if $f_{\mathbf{w}}(\mathbf{x}) >> 1$?

$\Rightarrow$ the cost will be very large!

- Same if $f_{\mathbf{w}}(\mathbf{x}) << -1$

$\Rightarrow$ The cost is not suited for classification!

Adapting the Formalism

In each region of space, we assume that:

- the label $+1$ follows a Bernoulli distribution parameterized by a function depending on $\mathbf{x}$:

$$p(y = 1|\mathbf{x}) = \mu(\mathbf{x})$$

- the label -1 as well: $p(y = -1|\mathbf{x}) = 1 - \mu(\mathbf{x})$

$\Rightarrow$ $p(y|\mathbf{x}) = \mu(\mathbf{x})^{\frac{y+1}{2}}(1 - \mu(\mathbf{x}))^{\frac{1-y}{2}}$

- How to represent $\mu(\mathbf{x})$? Linear function?

Adapting the Formalism

In each region of space, we assume that:

- the label $+1$ follows a Bernoulli distribution parameterized by a function depending on $\mathbf{x}$:

$$p(y = 1|\mathbf{x}) = \mu(\mathbf{x})$$

- the label -1 as well: $p(y = -1|\mathbf{x}) = 1 - \mu(\mathbf{x})$

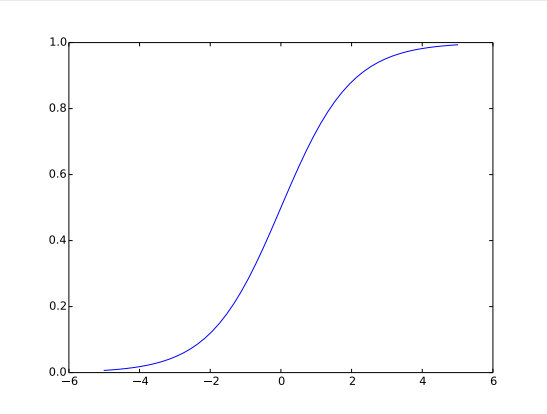
$\Rightarrow$ $p(y|\mathbf{x}) = \mu(\mathbf{x})^{\frac{y+1}{2}}(1 - \mu(\mathbf{x}))^{\frac{1-y}{2}}$

- How to represent $\mu(\mathbf{x})$? Linear function?

$\Rightarrow$ Problem! Not between 0 and 1!

- Transform a linear function: the sigmoid function.

$$\mu(\mathbf{x}) = \sigma(f_{\mathbf{w}}(\mathbf{x})) = \frac{1}{1 + e^{-f_{\mathbf{w}}(\mathbf{x})}}$$

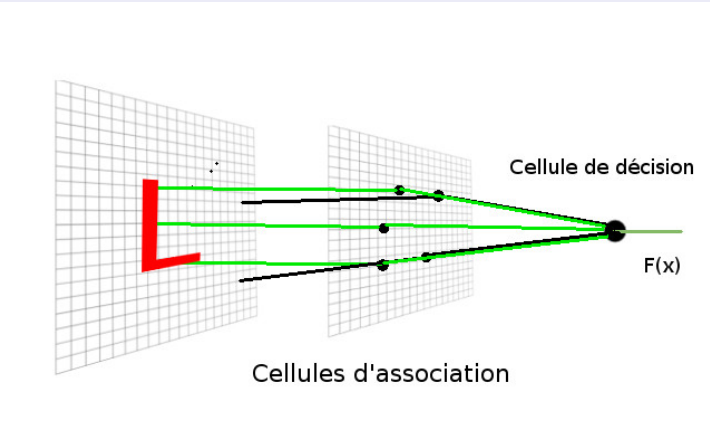


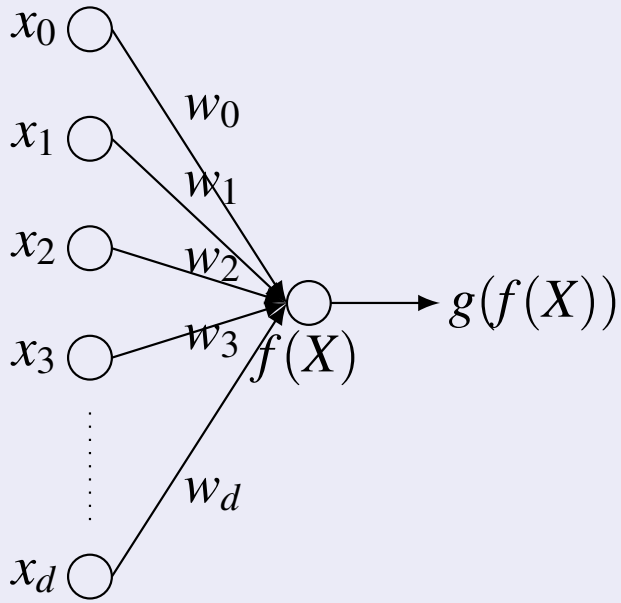
- Use likelihood as cost function (maximize $\prod_{i=1}^{N} P(y^i|\mathbf{x}^i)$)

Rosenblatt's Perceptron (1960)

The Idea

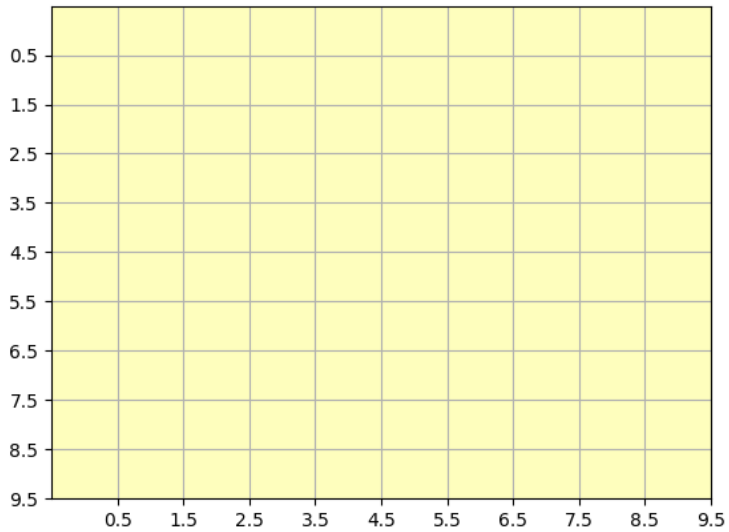
- Pattern recognition between two classes
- Inspired by the visual cortex





- Each association cell produces an output $x_i(S)$ based on a stimulus
- The decision cell responds according to a threshold function:
  $\sum w_i x_i(S) > \theta \Rightarrow +1$ otherwise -1

Learning Intuition



Initial weights state

Learning Intuition

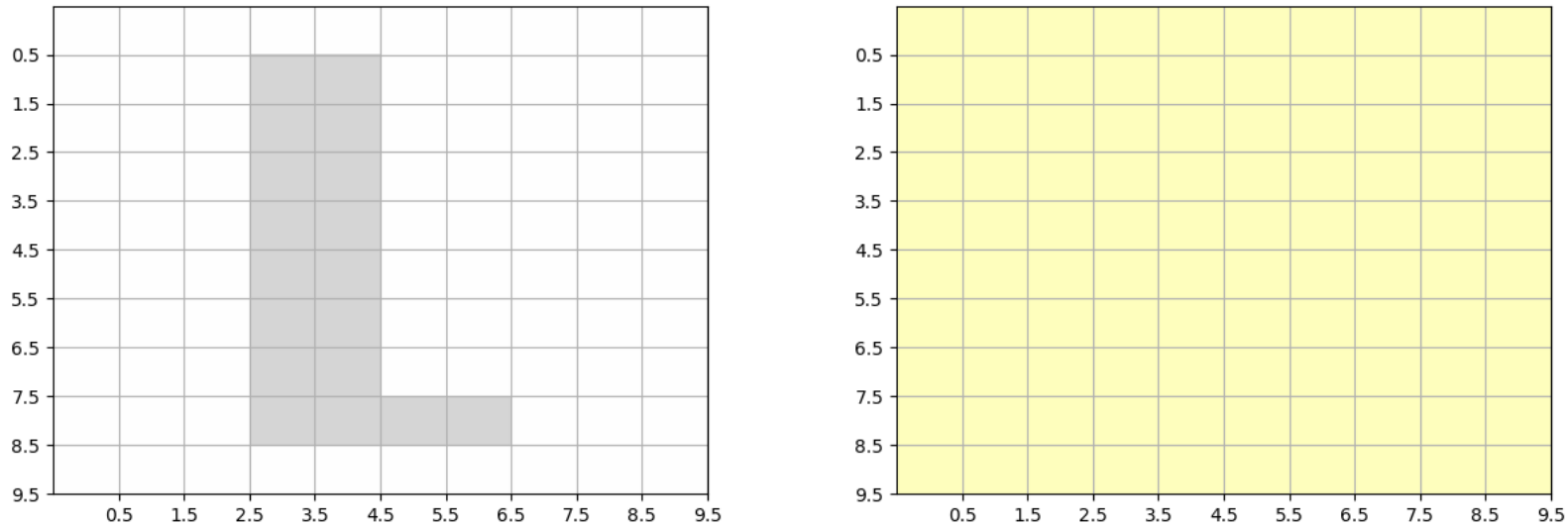


Image 1, Initial weights

Learning Intuition

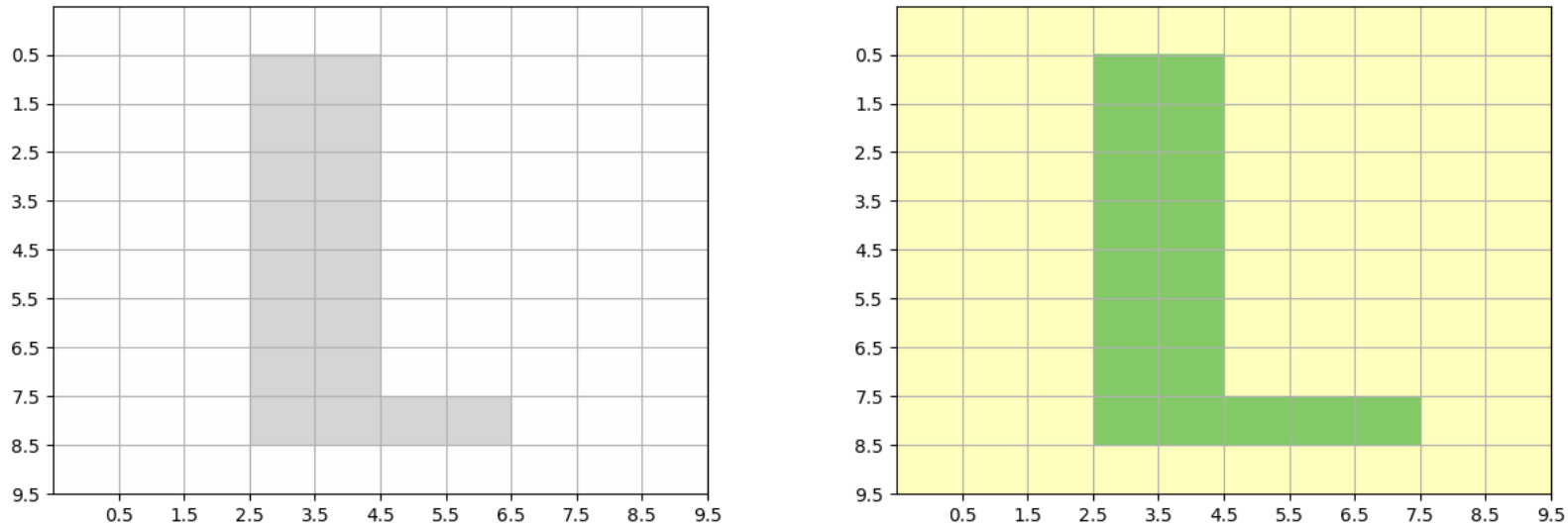


Image 1, Weights 1

Learning Intuition

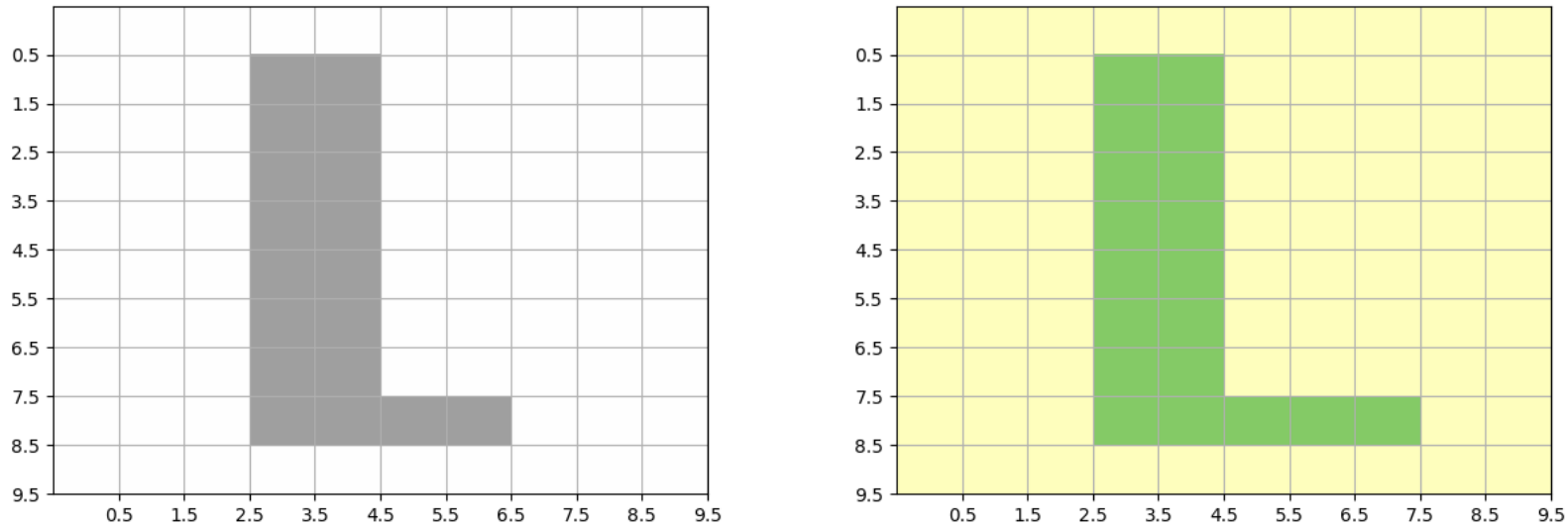


Image 2, Weights 1

Learning Intuition

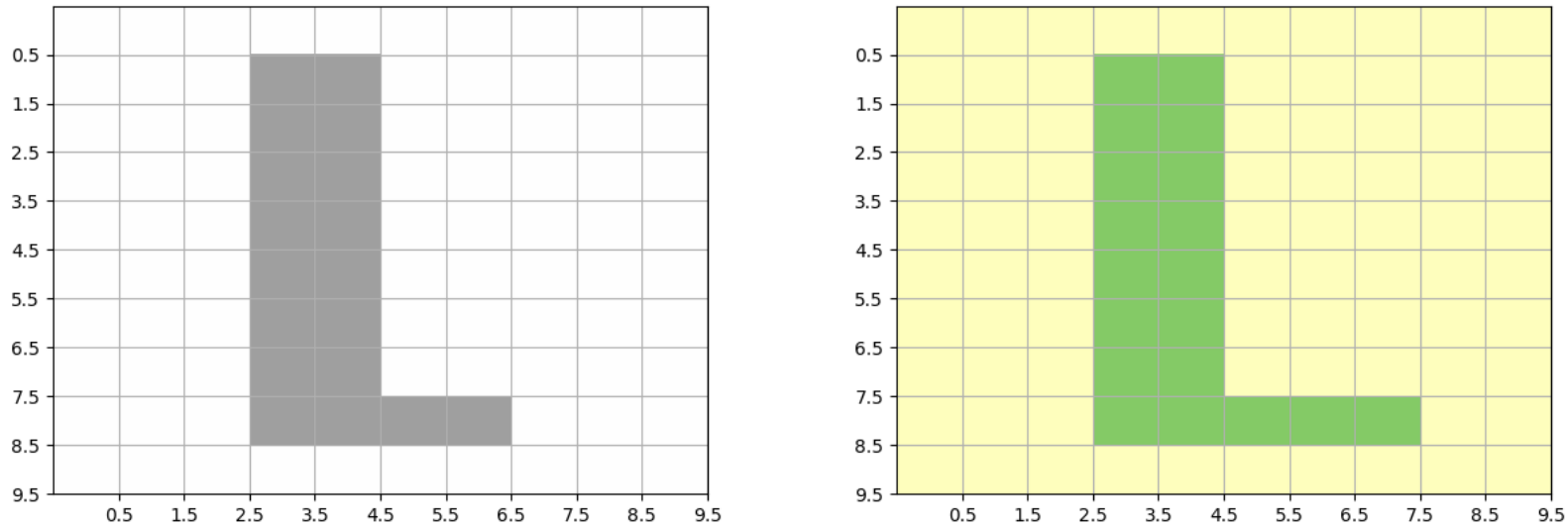
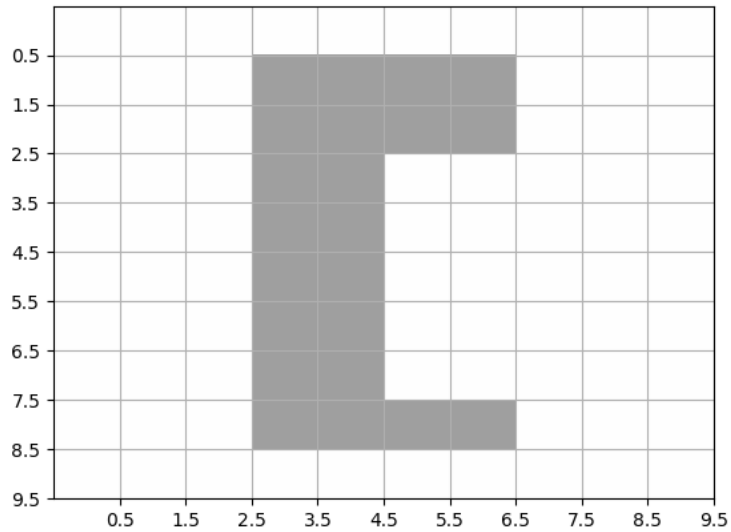


Image 2, Weights 2

Learning Intuition



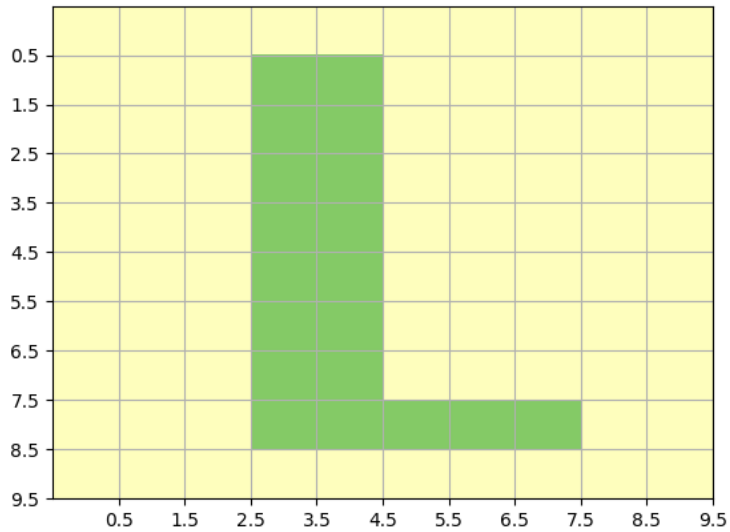
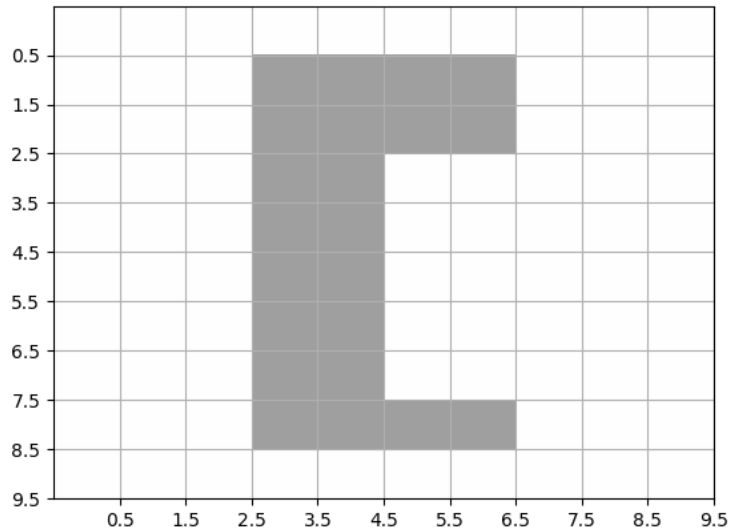


Image 3, Weights 2

Learning Intuition



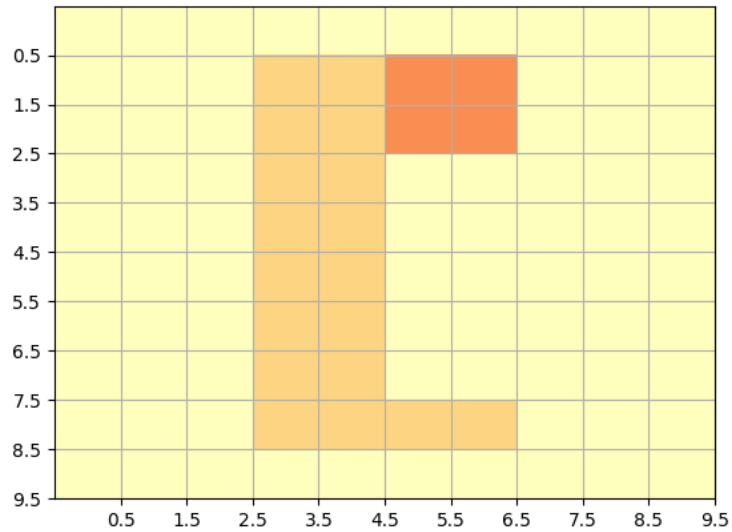


Image 3, Weights 3

Learning Intuition

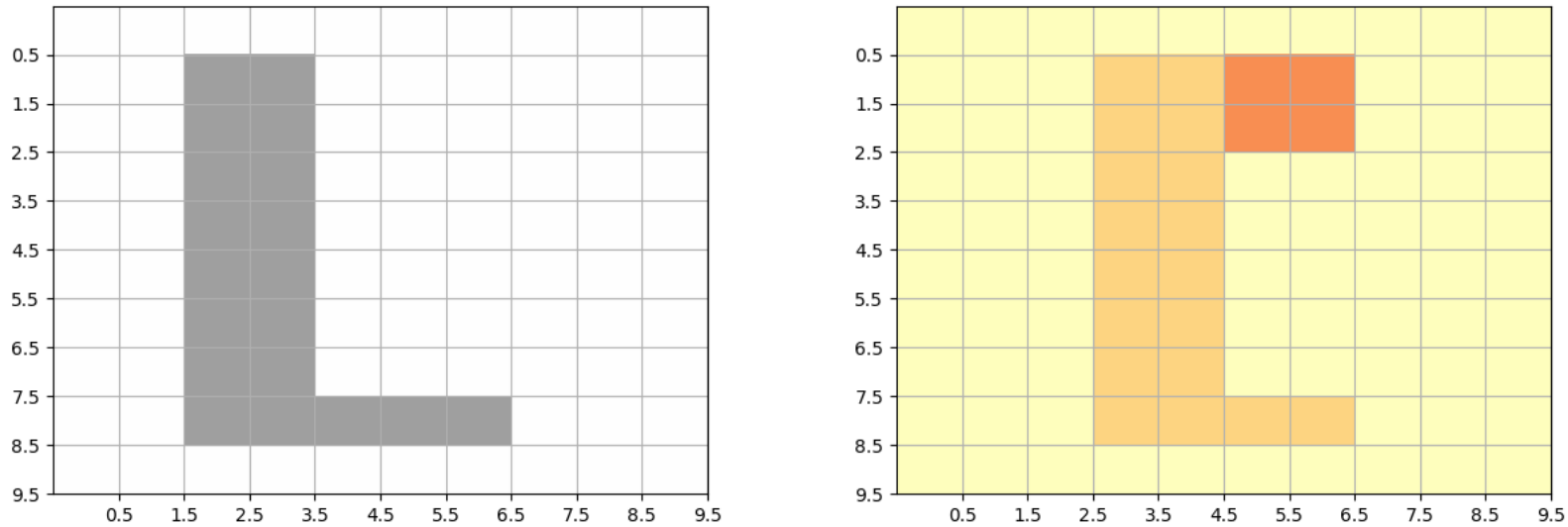


Image 4, Weights 3

Learning Intuition

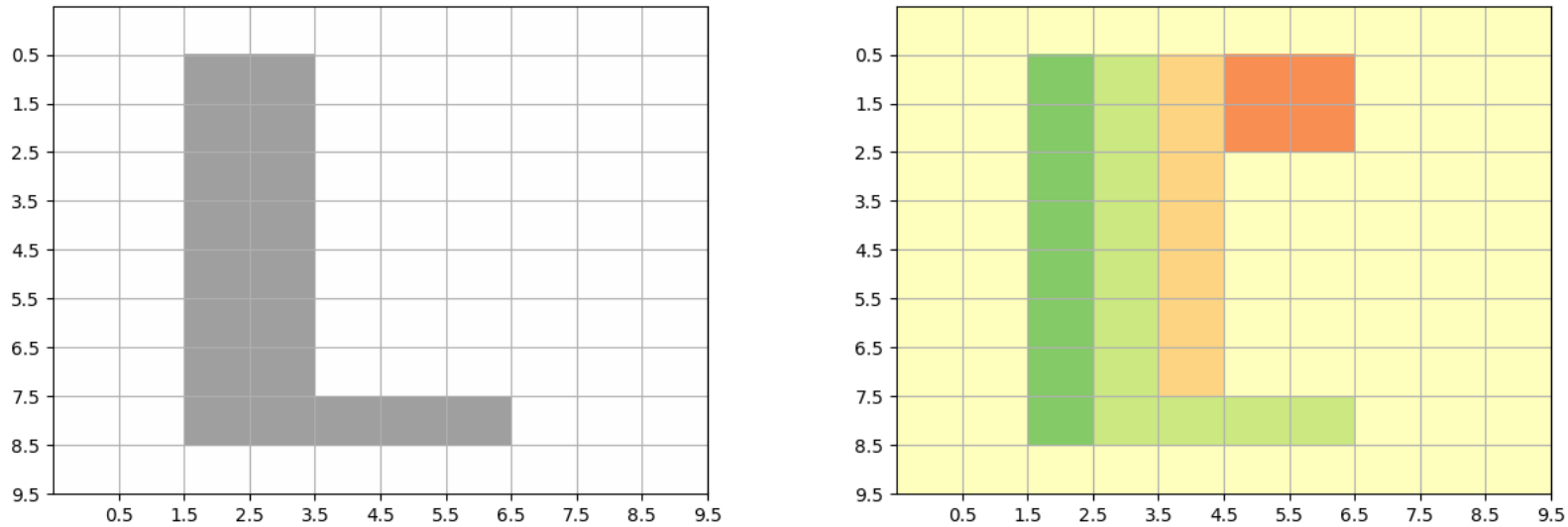


Image 4, Weights 4

Learning Algorithm

Perceptron Algorithm

- Initialize $\mathbf{w}$ randomly
- While not converged:
  - for all examples (x^i, y^i):
    - if $(y^i \times < \mathbf{w}.\mathbf{x}^i >) < 0$ then $\mathbf{w} = \mathbf{w} + \epsilon y^i \mathbf{x}^i$
- Decision: $f(x) = sign(< \mathbf{wx} >)$

What does the update rule correspond to?

Learning Algorithm

Perceptron Algorithm

- Initialize $\mathbf{w}$ randomly
- While not converged:
  - for all examples (x^i, y^i):
    - if $(y^i \times < \mathbf{w}.\mathbf{x}^i >) < 0$ then $\mathbf{w} = \mathbf{w} + \epsilon y^i \mathbf{x}^i$
- Decision: $f(x) = sign(< \mathbf{wx} >)$

What does the update rule correspond to?

- Hinge Loss: $l(f(x), y) = max(0, \alpha - yf(x)) = max(0, \alpha - y < w.\mathbf{x} >)$ with $\alpha = 0$
- $\nabla_w l(f_w(x), y) = \begin{cases} 0 & \text{if } (y < w.\mathbf{x} >) > \alpha \\ -yx_i & \text{otherwise} \end{cases}$

$\Rightarrow$ Gradient descent

The Perceptron: A Geometric View

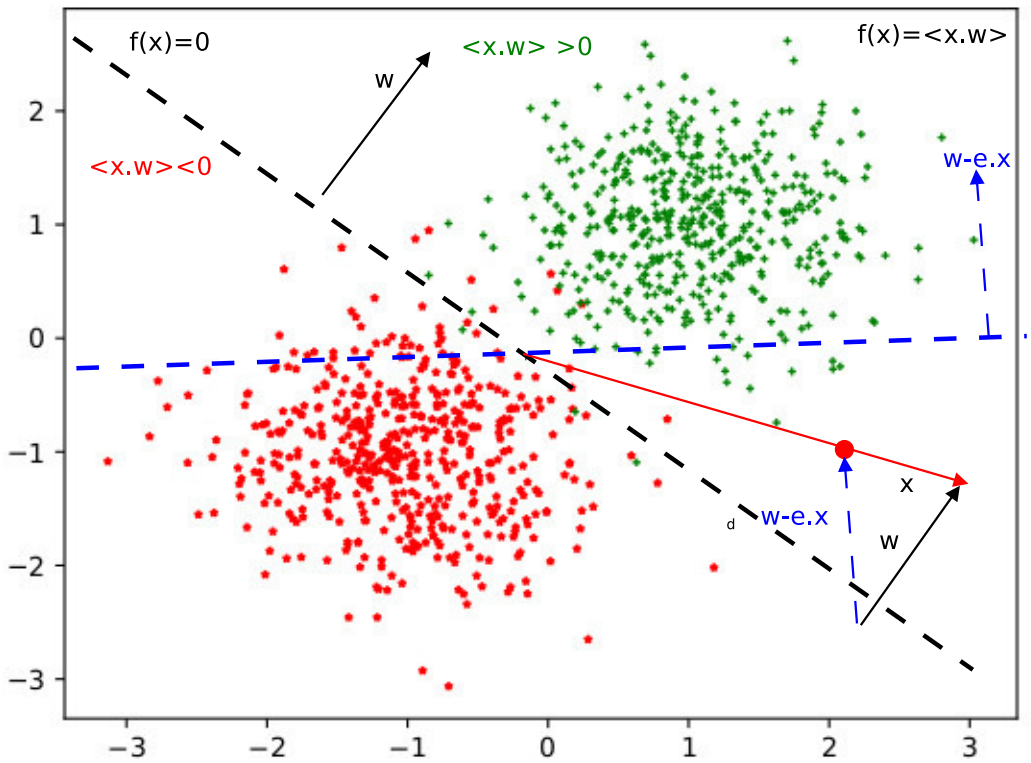
Algorithm

- Linear separator: $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$
- Decision: $\hat{y} = \text{sign}(f(\mathbf{x}))$
- Update rule (if misclassified):

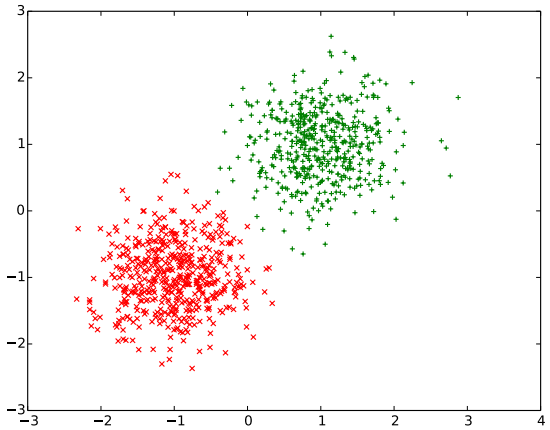
$$\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$$

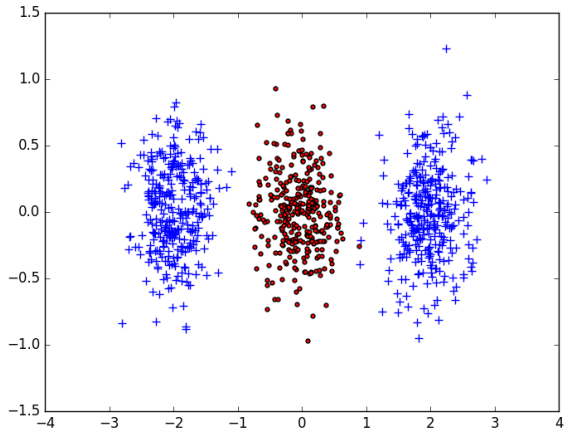
Geometric Interpretation

- $\mathbf{w}$ is normal to the decision boundary
- $\langle \mathbf{w}, \mathbf{x} \rangle$: signed distance to boundary
- Update: rotate $\mathbf{w}$ towards misclassified point



The Limits of Linearity



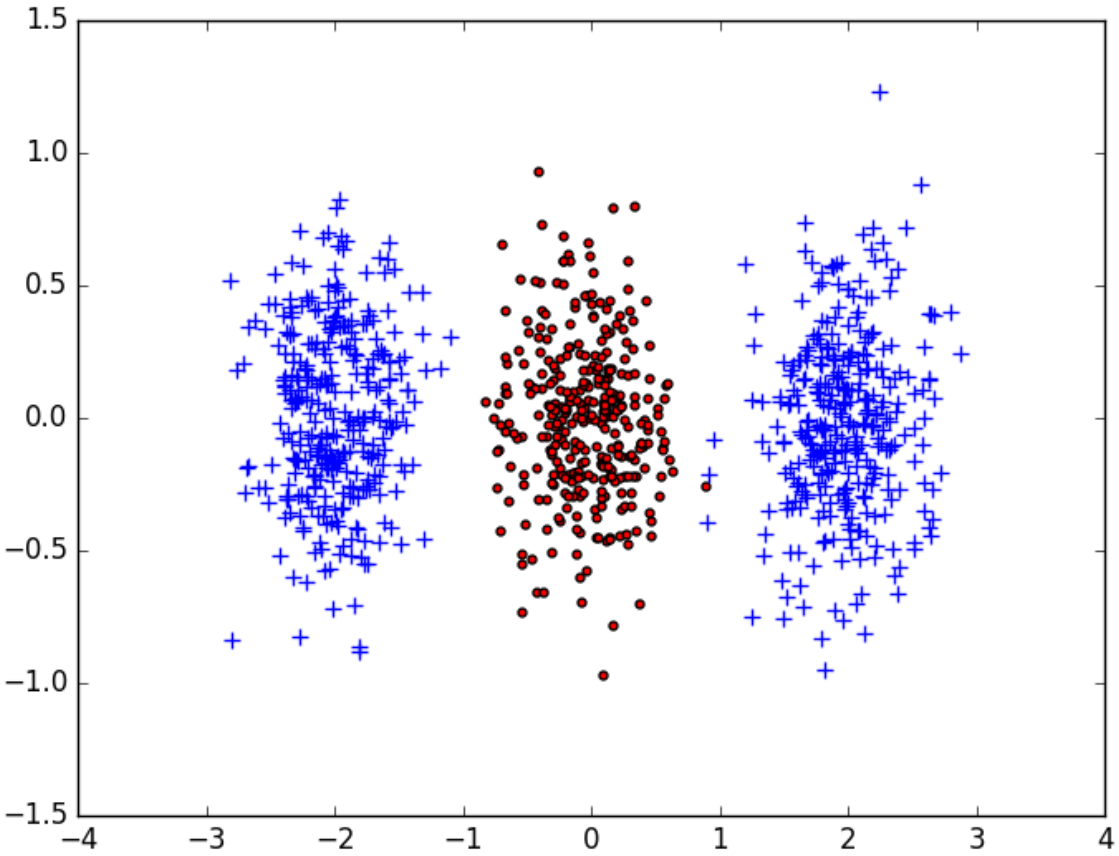


What if data is not linearly separable?

1. **Feature projection**: Map to higher dimensions
2. **Non-linear models**: Neural networks

Outline

Non-Linearly Separable Data

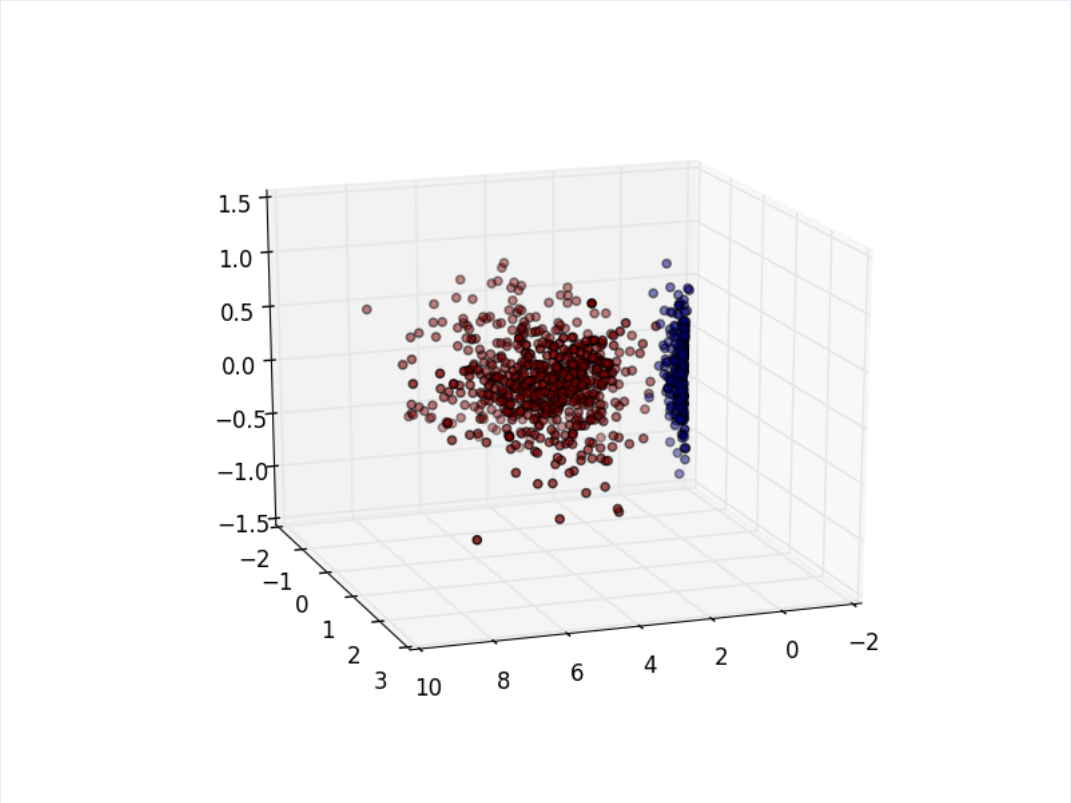


What to do?

Non-Linearly Separable Data

Representation Transformation

- We increase the dimension: $(x_1, x_2) \rightarrow (x_1^2, x_1 x_2, x_2)$



- The problem is linearly separable again!

Data Projection

Express data in a higher dimensional space

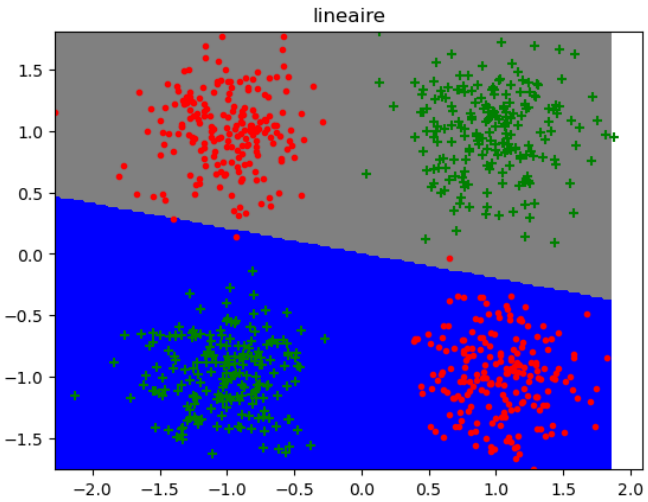
- We consider a function $\phi(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$, typically $d' >> d$.
- New training set: $\{(\boldsymbol{z}^k, y^k)\}_{k=1}^N$ with $\boldsymbol{z}^i = \phi(\mathbf{x}^i)$
- Linear model: $\sum_{i=1}^{d'} w_i z_i = \sum_{i=1}^{d'} w_i \phi(x_i)$
- Same optimization algorithm: gradient descent with perceptron cost

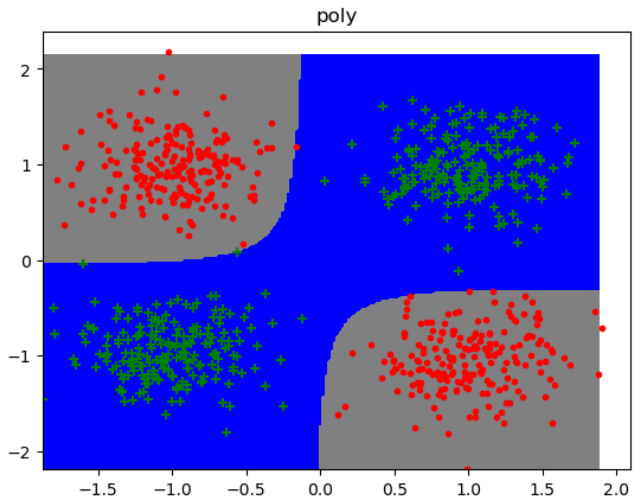
Example: Polynomial Projection

Consider $\mathbf{x} \in \mathbb{R}^3$, for a degree 2 projection:

- $\phi(\mathbf{x}) = (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2) \in \mathbb{R}^9$
- New form of decision boundary:
  $w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1x_2 + w_5x_1x_3 + w_6x_2x_3 + w_7x_1^2 + w_8x_2^2 + w_9x_3^2 = 0$

Data Projection





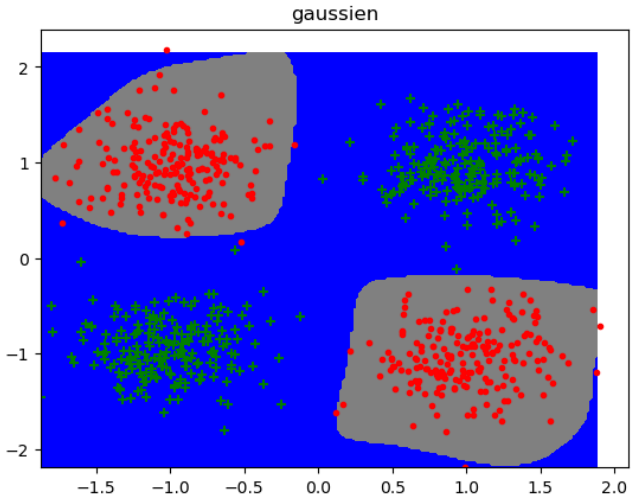
Example: Polynomial Projection

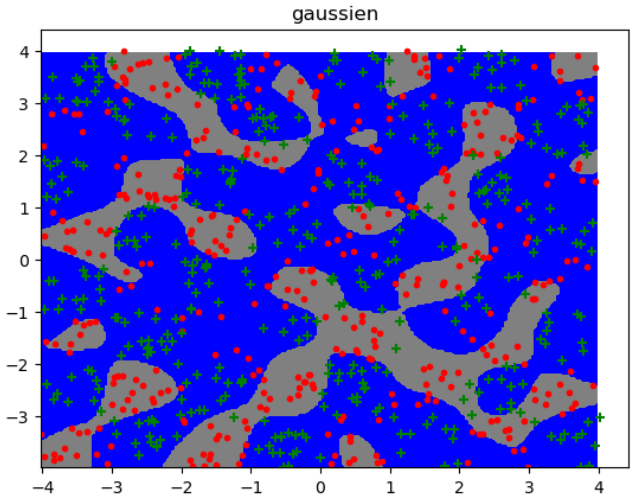
Consider $\mathbf{x} \in \mathbb{R}^3$, for a degree 2 projection:

- $\phi(\mathbf{x}) = (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2) \in \mathbb{R}^9$
- New form of decision boundary:

$$w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1x_2 + w_5x_1x_3 + w_6x_2x_3 + w_7x_1^2 + w_8x_2^2 + w_9x_3^2 = 0$$

Data Projection



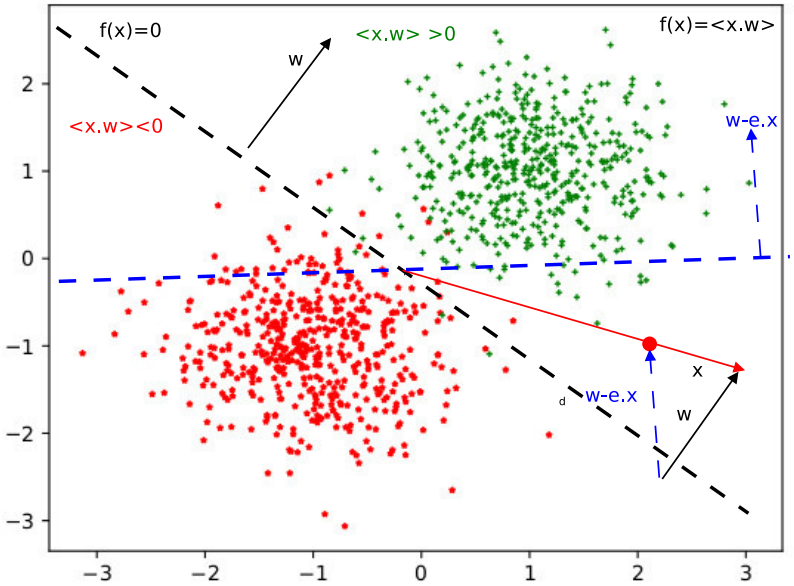


Example: Gaussian Projection on a basis

Consider a basis $\{\mathbf{x}'^j\}_{j=1}^M$ of points in $\mathbb{R}^d$, the projection of an example $\mathbf{x}$ consists of the M Gaussian distances to basis points $\mathbf{x}'^j$:

- $\phi(\mathbf{x}) = (e^{\|\mathbf{x}-\mathbf{x}'^1\|^2/2\sigma}, e^{\|\mathbf{x}-\mathbf{x}'^2\|^2/2\sigma}, \ldots, e^{\|\mathbf{x}-\mathbf{x}'^M\|^2/2\sigma})$
- New form of decision boundary:
  $w_1 e^{\|\mathbf{x}-\mathbf{x}'^1\|^2/2\sigma} + w_2 e^{\|\mathbf{x}-\mathbf{x}'^2\|^2/2\sigma} + \ldots + w_M e^{\|\mathbf{x}-\mathbf{x}'^M\|^2/2\sigma} = 0$
- Only points $\mathbf{x}'$ close to $\mathbf{x}$ play a role (and thus only the associated weights).
- Very similar to k-NN with distance weighting, **but** weights are learned automatically!

Back to the Perceptron



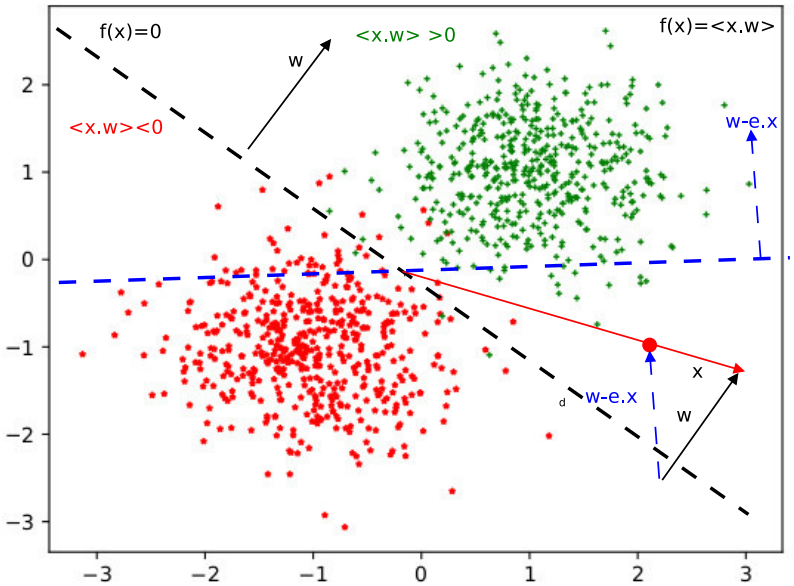
Geometric Considerations

What does it represent:

- $\mathbf{w}$ with respect to the separator? $\Rightarrow$ the normal to the hyperplane
- $\langle \mathbf{w}, \mathbf{x} \rangle = \cos(\mathbf{w}, \mathbf{x}) \|\mathbf{x}\| \|\mathbf{w}\|$, angle between the two vectors.
- the update rule: if $(y\langle w.\mathbf{x} \rangle) < 0$ correct $\mathbf{w} = \mathbf{w} + y\mathbf{x}$?

$\Rightarrow$ $\langle \mathbf{w} + y\mathbf{x}, \mathbf{x} \rangle = \langle \mathbf{w}, \mathbf{x} \rangle + y\|\mathbf{x}\|^2$, thus allows to increase or decrease the angle

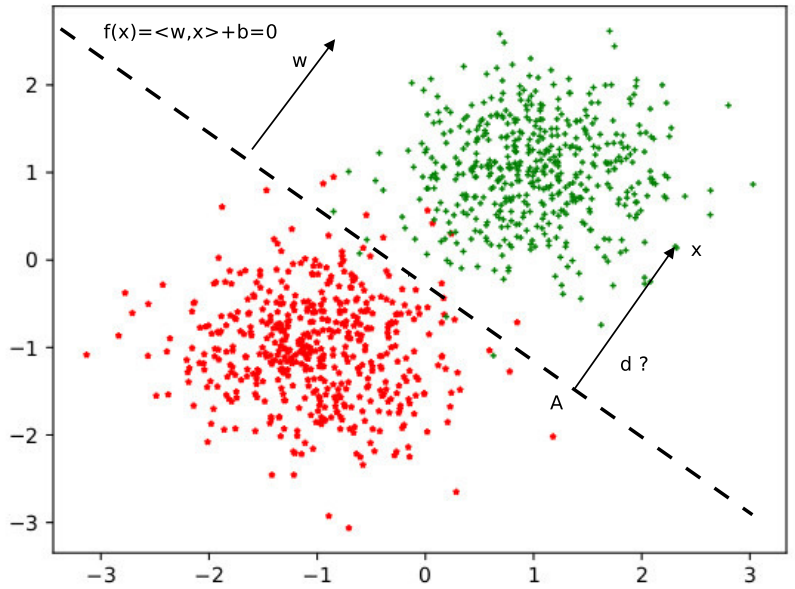
Back to the Perceptron



Questions

- Unique solution?
- Some solutions better than others?

Geometric Considerations

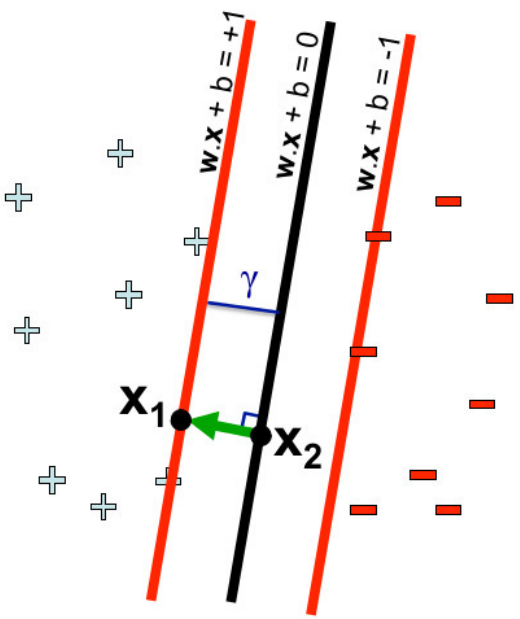


Geometric Distance to the Separator

- $d = |AX|$, where $X = A + d\frac{\mathbf{w}}{\|\mathbf{w}\|}$ (for an example above the separator)
- Moreover, $f(A) =< \mathbf{w}, A > +b = 0$, thus $\langle \mathbf{w}, X - d\frac{\mathbf{w}}{\|\mathbf{w}\|} \rangle + b = 0$
- Hence $d = \frac{\langle \mathbf{w}, X \rangle + b}{\|\mathbf{w}\|} = \frac{f(\mathbf{x})}{\|\mathbf{w}\|}$.
- $f(\mathbf{x})$ is the *functional* distance to the separator

Note: $\mathbf{w}$ is defined up to a multiplicative constant...

Uniqueness of the Solution



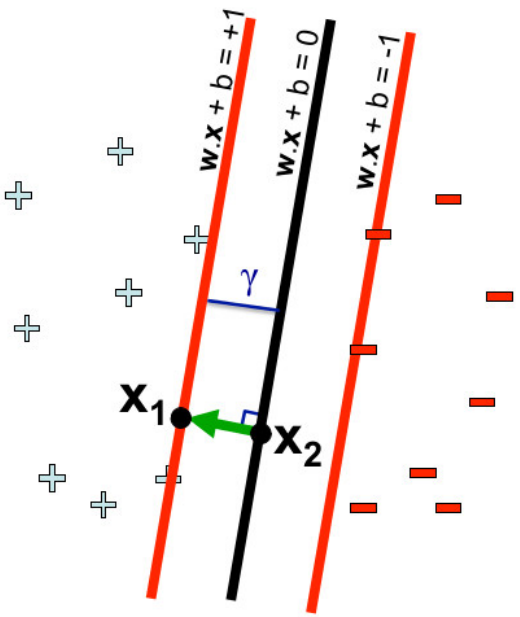
If separable

- Introduce a constraint: the functional distance to the closest points is fixed at 1

$\Rightarrow f(\mathbf{x}) = \mathbf{w}\mathbf{x}_1 + b = \pm 1$

- All points satisfy: $yf(\mathbf{x}) \geq 1$.
- Let γ denote the Euclidean distance: $\mathbf{x}_1 - \mathbf{x}_2 = \gamma \frac{\mathbf{w}}{\|\mathbf{w}\|}$
- γ: distance between the hyperplane (boundary) and the closest point $\mathbf{x}_1$
- γ is called the margin (symmetric? why?)

Maximizing the Margin



If separable

- $f(\mathbf{x}) = \mathbf{w}\mathbf{x}_1 + b = \pm 1$ and $\mathbf{x}_1 - \mathbf{x}_2 = \gamma \frac{\mathbf{w}}{\|\mathbf{w}\|}$
- We have
  $f(\mathbf{x}_1) - f(\mathbf{x}_2) = 1 = (\mathbf{w}\mathbf{x}_1 + b) - (\mathbf{w}\mathbf{x}_2 + b) = \mathbf{w}(\mathbf{x}_1 - \mathbf{x}_2)$
- Thus $\gamma \frac{\mathbf{w}\mathbf{w}}{\|\mathbf{w}\|} = 1$, so $\gamma \|\mathbf{w}\| = 1$, hence $\gamma = \frac{1}{\|\mathbf{w}\|}$

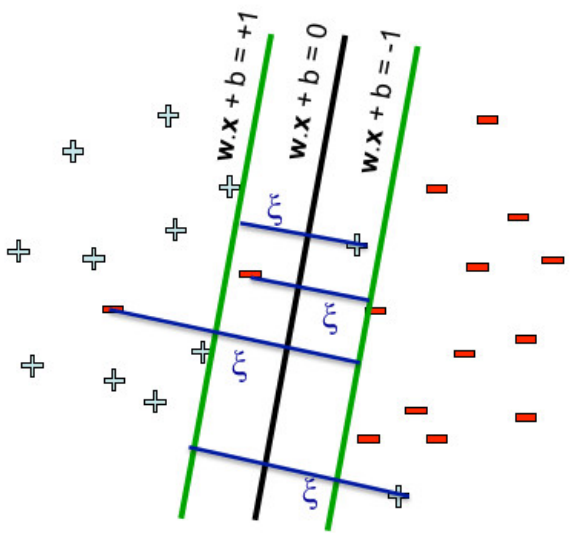
$\Rightarrow$ Maximize the margin $\Leftrightarrow$ minimize $\|\mathbf{w}\|$!

- New formulation:
  minimize $\|\mathbf{w}\|^2$ such that

$$\forall i, (\mathbf{w}\mathbf{x}^i + b) y^i \geq 1$$

Convex quadratic optimization problem

What if Data is Noisy?



Support Vector Machine Approach

- Introduce slack variables ξ^i: we tolerate an "overflow" $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 - \xi_i$, with $\xi_i \geq 0$.
- This "overflow" should be as small as possible

$\Rightarrow$ Minimize $\|\mathbf{w}\|^2 + K\sum \xi_j$
s.t. $(\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 - \xi_i$ and $\xi_i \geq 0$

- If margin is greater than 1 $\rightarrow$ no overflow cost, otherwise linear cost:

$$\begin{cases} \xi_i = 0 & \text{if } (\mathbf{w}\mathbf{x}^i + b)y^i \geq 1 \\ \xi_i = 1 - (\mathbf{w}\mathbf{x}^i + b)y^i & \text{if } (\mathbf{w}\mathbf{x}^i + b)y^i < 1 \end{cases}$$

$\Rightarrow$ $\xi_i = max(0, 1 - (\mathbf{w}\mathbf{x}^i + b)y^i)$ (hinge-loss!)

Formulation

- Minimize: $\|\mathbf{w}\|^2 + K\sum \ell(y^i, \mathbf{w}\mathbf{x}^i + b)$, with $\ell(y, \hat{y}) = max(0, 1 - y\hat{y})$
- $\|\mathbf{w}\|^2 \rightarrow$ regularization term to control overfitting.

Resolution

Optimize the Lagrangian

- $L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 + K\sum_i \xi_i - \sum_i \alpha_i(y^i(\mathbf{w}\mathbf{x}^i + b) + \xi_i - 1) - \sum_i \eta_i\xi_i$

Important Remarks at Optimum

- $\mathbf{w} = \sum_i \alpha_i y^i \mathbf{x}^i$: the weight vector is a linear combination of examples
- There are (even many) α_i that are zero $\Rightarrow$ examples not taken into account (normal?)
- The resulting decision function:

$$f(\mathbf{x}) = \langle \mathbf{w}\mathbf{x} \rangle + b = \sum_i \alpha_i y^i \langle \mathbf{x}^i \mathbf{x} \rangle + b$$

only involves dot products between training examples.

In all formulations, what matters is the dot product!

- minimize$_{\mathbf{w},b}$ $\frac{1}{2}\|\mathbf{w}\|^2 + K\sum_i \xi_i$ s.t. $y^i(\mathbf{w}\mathbf{x}^i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

$\Leftrightarrow$ maximize$_{\alpha}$ $-\frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y^i y^j < \mathbf{x}^i, \mathbf{x}^j > + \sum_i \alpha_i$
s.t. $\sum \alpha_i y^i = 0$ and $\alpha_i \in [0, K]$.

The Kernel Trick

Non-linearity $\rightarrow$ projection

- We want to consider a projection $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^{n'}, \quad n << n'$
- For example, $\phi(\mathbf{x}) = (x_1, x_2, \ldots, x_d, x_1x_1, x_1x_2, \ldots, x_dx_d)$
- We can denote: $K(\mathbf{x}^i, \mathbf{x}^j) = \langle \phi(\mathbf{x}^i), \phi(\mathbf{x}^j) \rangle$
- $\mathbf{w}^* = \sum_i \alpha_i y^i \phi(\mathbf{x}^i)$

SVM involves the following quantities:

Original space	Projected space
$\langle \mathbf{x}^1, \mathbf{x}^2 \rangle$	$\langle \phi(\mathbf{x}^1), \phi(\mathbf{x}^2) \rangle = K(\mathbf{x}^1, \mathbf{x}^2)$
$\langle \mathbf{w}, \mathbf{x} \rangle$	$\langle \mathbf{w}, \phi(\mathbf{x}) \rangle = \sum_i \alpha_i y^i \langle \phi(\mathbf{x}^i) \phi(\mathbf{x}) \rangle = \sum_i \alpha_i y^i K(\mathbf{x}^i, \mathbf{x})$
$f(\mathbf{x}) = \sum_i \alpha_i y^i \langle \mathbf{x}^i, \mathbf{x} \rangle + b$	$f(\mathbf{x}) = \sum_i \alpha_i y^i \langle \phi(\mathbf{x}^i), \phi(\mathbf{x}) \rangle + b = \sum_i \alpha_i y^i K(\mathbf{x}^i, \mathbf{x}) + b$

Projection and Dot Product

Example: Polynomial Projection

Let $\phi_2(\mathbf{x}) = (x_0x_0, x_0x_1, \ldots, x_0x_d, x_1x_0, x_1x_1, \ldots, x_dx_d)$ with $x_0 = 1$:

- $\langle \phi_2(\mathbf{x})\phi_2(\mathbf{x}') \rangle = \sum_{i=0}^{d} \sum_{j=0}^{d} (x_i x_j)(x'_i x'_j)$

$= \sum_{i=0}^{d} \sum_{j=0}^{d} (x_i x'_i)(x_j x'_j)$

$= (x_0x'_0 + x_1x'_1 + \ldots x_dx'_d)^2$

$= (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^2$

$K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^2$ much cheaper to compute (linear in D).

- Kernel Trick: replace the costly (and not always possible) computation of $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ by a cheaper function $K(\mathbf{x}, \mathbf{x}')$

An Infinite Projection: The Gaussian Kernel

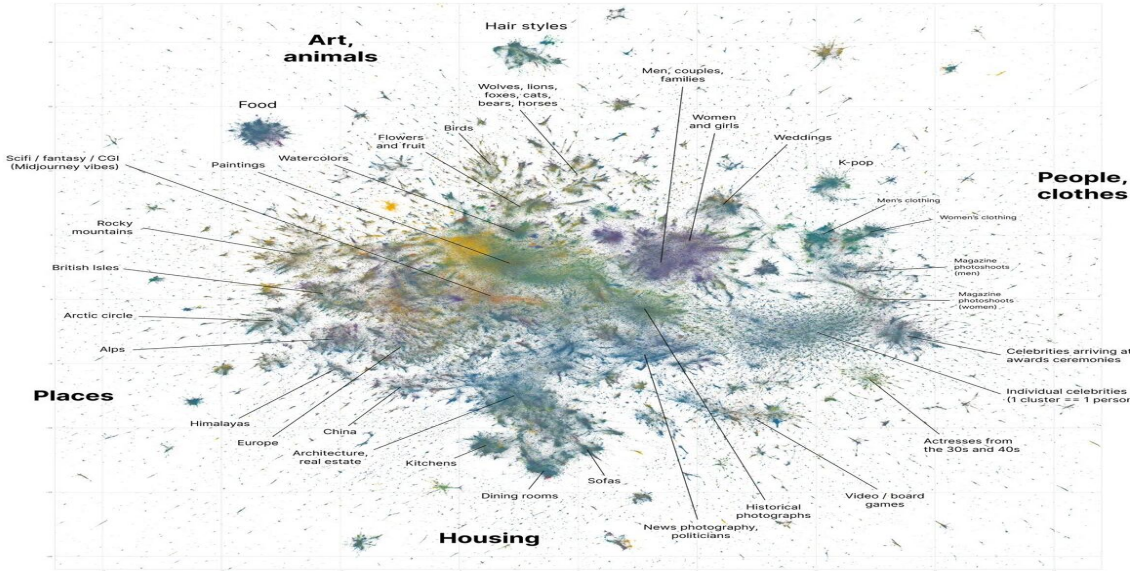
Let $K(\mathbf{x}, \mathbf{x}') = e^{-\gamma\|\mathbf{x}-\mathbf{x}'\|^2}$:

$$
\begin{aligned}
K(\mathbf{x}, \mathbf{x}') &= e^{-\gamma\|\mathbf{x}-\mathbf{x}'\|^2} \\
&= e^{-\gamma(\|\mathbf{x}\|^2+\|\mathbf{x}'\|^2-2\mathbf{x}\mathbf{x}')} \\
&= e^{-\gamma\|\mathbf{x}\|^2} e^{-\gamma\|\mathbf{x}'\|^2} e^{2\gamma\mathbf{x}\mathbf{x}'} \\
&= e^{-\gamma\|\mathbf{x}\|^2} e^{-\gamma\|\mathbf{x}'\|^2} \sum_{n=0}^{\infty} \frac{(2\gamma\mathbf{x}\mathbf{x}')^n}{n!} \\
&= \sum_{n=0}^{\infty} \left(\sqrt{\frac{(2\gamma)^n}{n!}} e^{-\gamma\|\mathbf{x}\|^2} \phi_n(\mathbf{x}) \right) \left(\sqrt{\frac{(2\gamma)^n}{n!}} e^{-\gamma\|\mathbf{x}'\|^2} \phi_n(\mathbf{x}') \right)
\end{aligned}
$$

With ϕ_n such that $\langle \phi_n(\mathbf{x}) \phi_n(\mathbf{x}') \rangle = (\langle \mathbf{x}\mathbf{x}' \rangle)^n$
$\Rightarrow K$ corresponds to the dot product of an infinite concatenation of projections

Outline

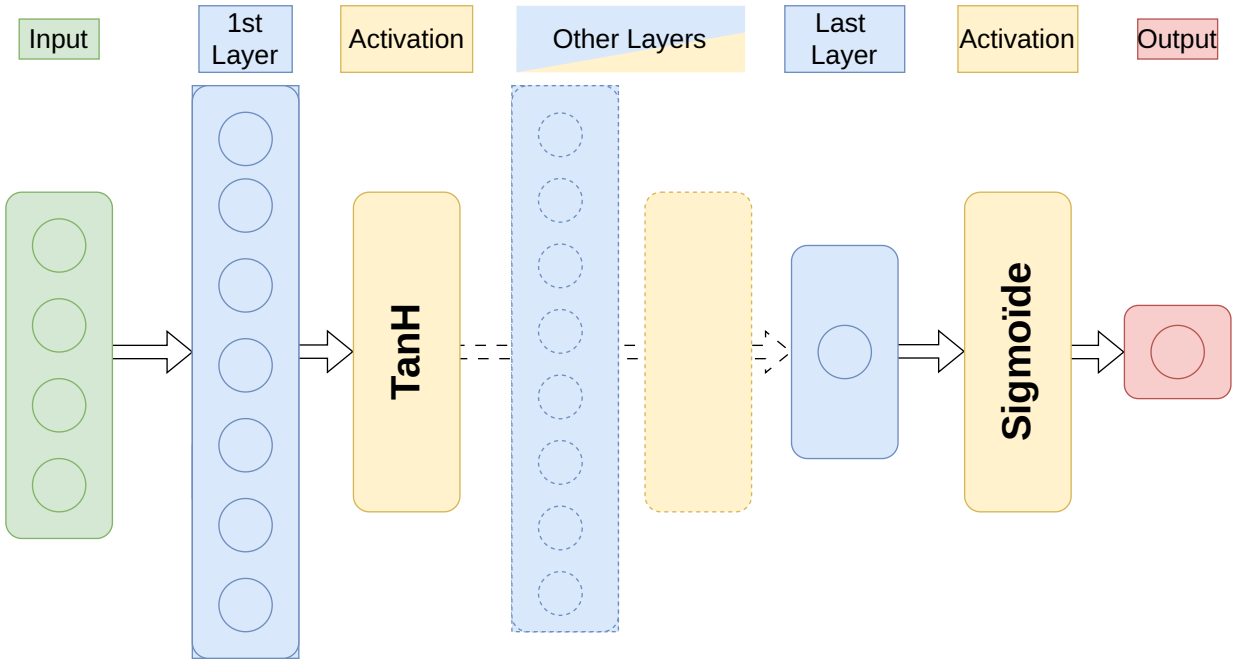
When There Are No Features . . .



We need to project discrete objects into a continuous space

- Feature engineering is tedious and domain-specific
- Random projection: existed (feature hashing), not very effective
- No description of objects, but there are relationships between them!
- A ”good” representation space must be able to express the latent semantics of these relationships
- Can be guided by the expert (before Deep Learning), the task (end-to-end learning) or by data without supervision (self-supervised for example)

Neural Networks as Feature Learners



Multi-Layer Perceptron (MLP)

- Stack of linear layers + non-linear activations
- Hidden layers = **learned features**
- Output layer = task-specific (classification, regression)
- Training: backpropagation (chain rule + gradient descent)

Cost Functions for Different Tasks

Regression: Mean Squared Error

$$L = \frac{1}{N}\sum_i (y^i - \hat{y}^i)^2$$

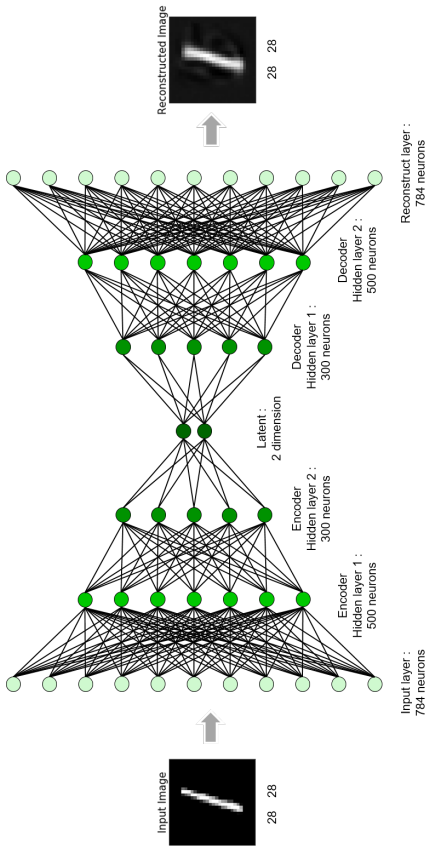
Binary Classification: Binary Cross-Entropy

$$L = -\frac{1}{N}\sum_i [y^i \log(\hat{y}^i) + (1 - y^i)\log(1 - \hat{y}^i)]$$

Multi-class Classification: Cross-Entropy + Softmax

$$\text{Softmax}(z)_k = \frac{e^{z_k}}{\sum_j e^{z_j}} \qquad L = -\log(\hat{y}_{\text{true class}})$$

Auto-encoders: Learning Representations



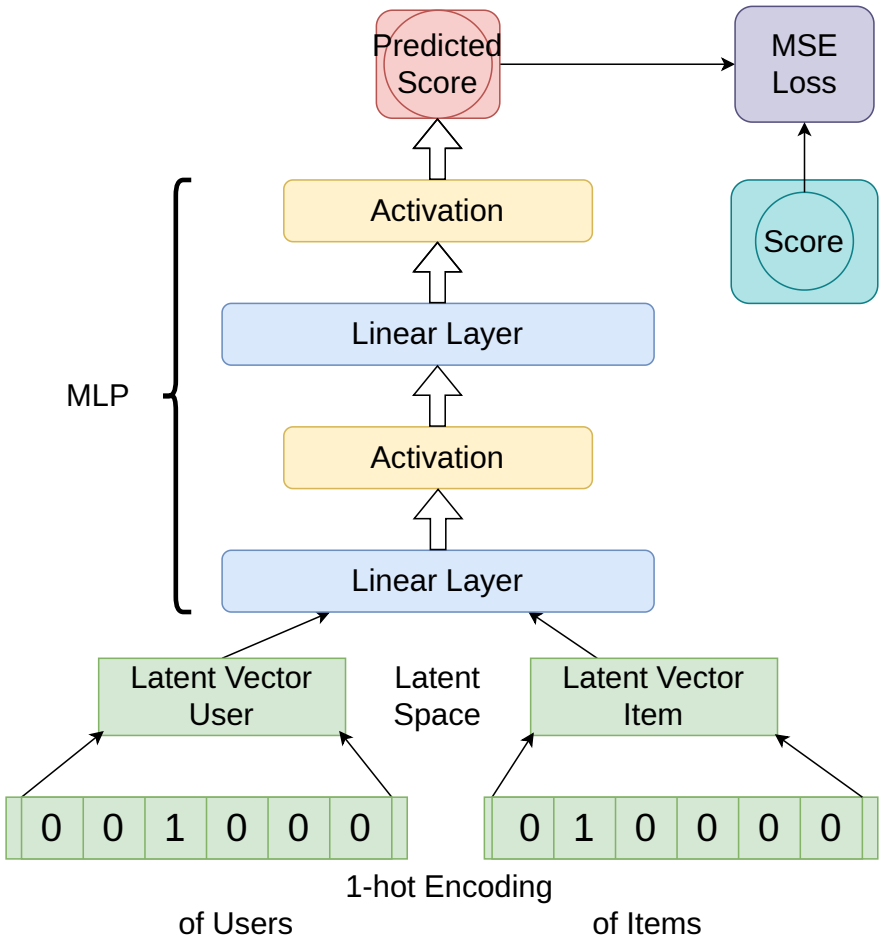
Unsupervised Learning

- Input = Output target
- Bottleneck forces compression
- Learn useful representations

Applications

- Dimensionality reduction
- Denoising
- Anomaly detection
- Pre-training

Representation and Recommendation



Data

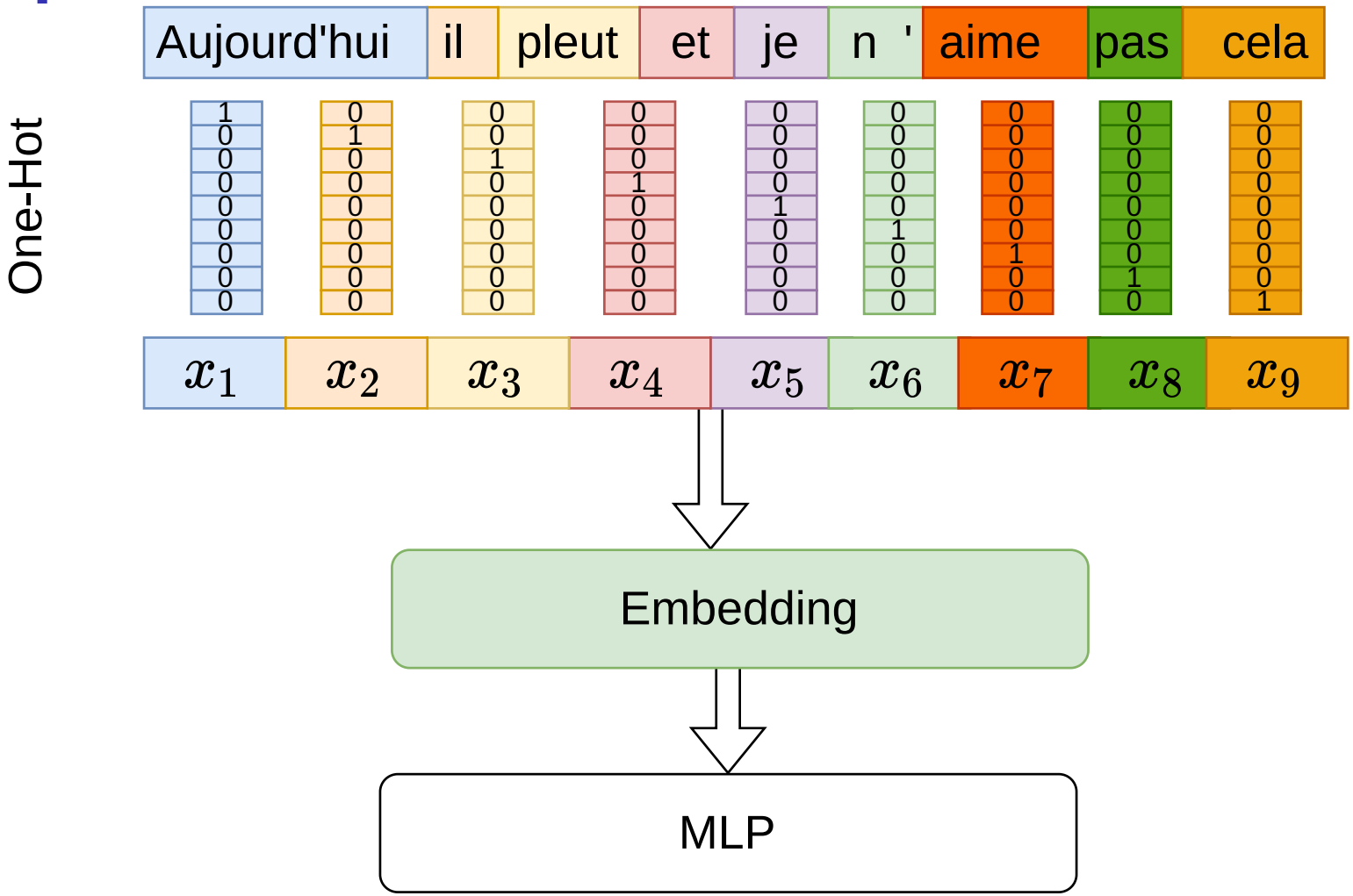
- Products (without description)
- Users (without description)
- User ratings on products

Goal: predict user opinions

Procedure:

- One-hot encoding of products and users (one product and one user per dimension)
- Random representation initially
- Backpropagation corrects the MLP and user/product representations

Representation and Text



Same goal:

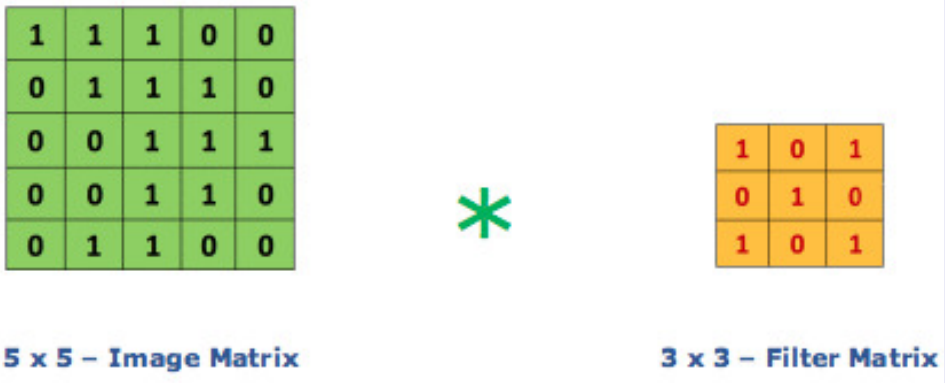
- Find a ”useful” representation for atomic elements (tokens)
- Tokenization problem

Specialized Architectures

Convolutional Networks (CNN)

For images and signals:

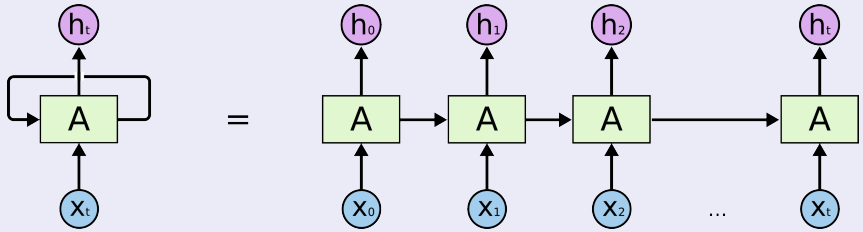
- Local filters detect patterns
- Translation invariance
- Hierarchical features



Recurrent Networks (RNN)

For sequences:

- Process variable-length input
- Memory of past elements
- LSTM/GRU for long-range



Outline

Summary: The ML Journey

1. Probabilistic Foundations

Bayes theorem, density estimation, k-NN

2. Geometric Foundations

Linear models, decision boundaries, gradient descent

3. Feature Engineering

Projections, kernels, SVM margin maximization

4. Representation Learning

Neural networks learn features automatically

Common Thread

Finding the right representation of data

Thank you!

Questions?

Nicolas Baskiotis
nicolas.baskiotis@sorbonne-universite.fr

MLIA Team, ISIR
Sorbonne Université