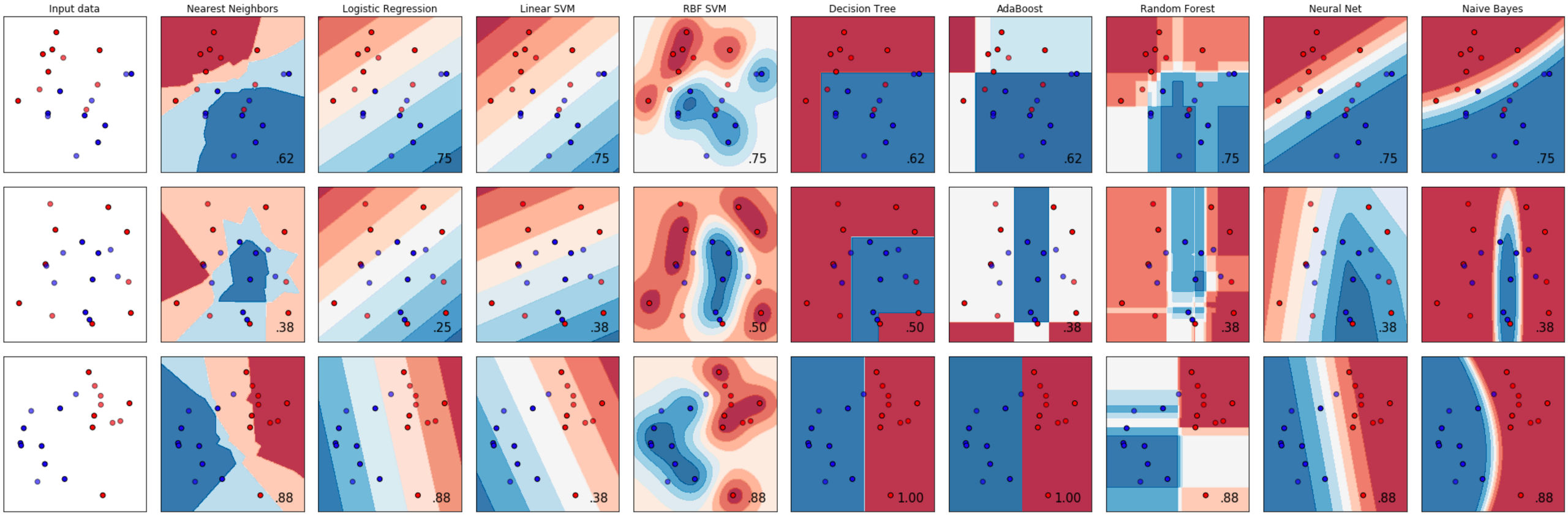
Machine Learning
Part 2

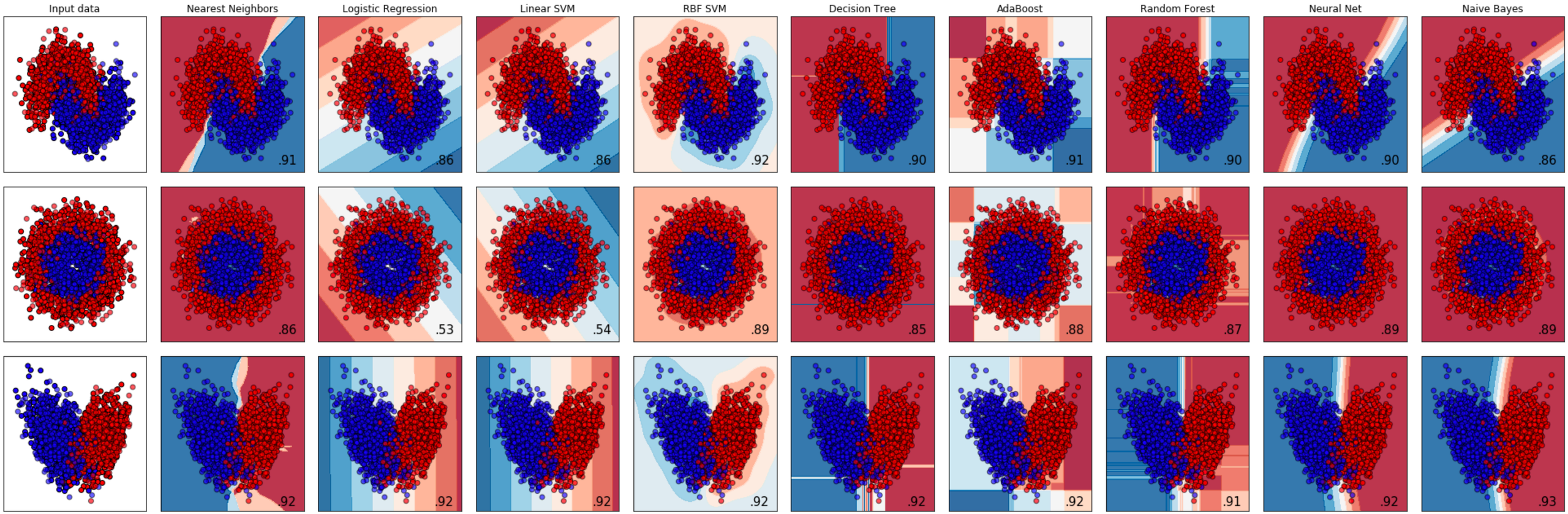
VHIO course

UNIVERSITAT DE BARCELONA

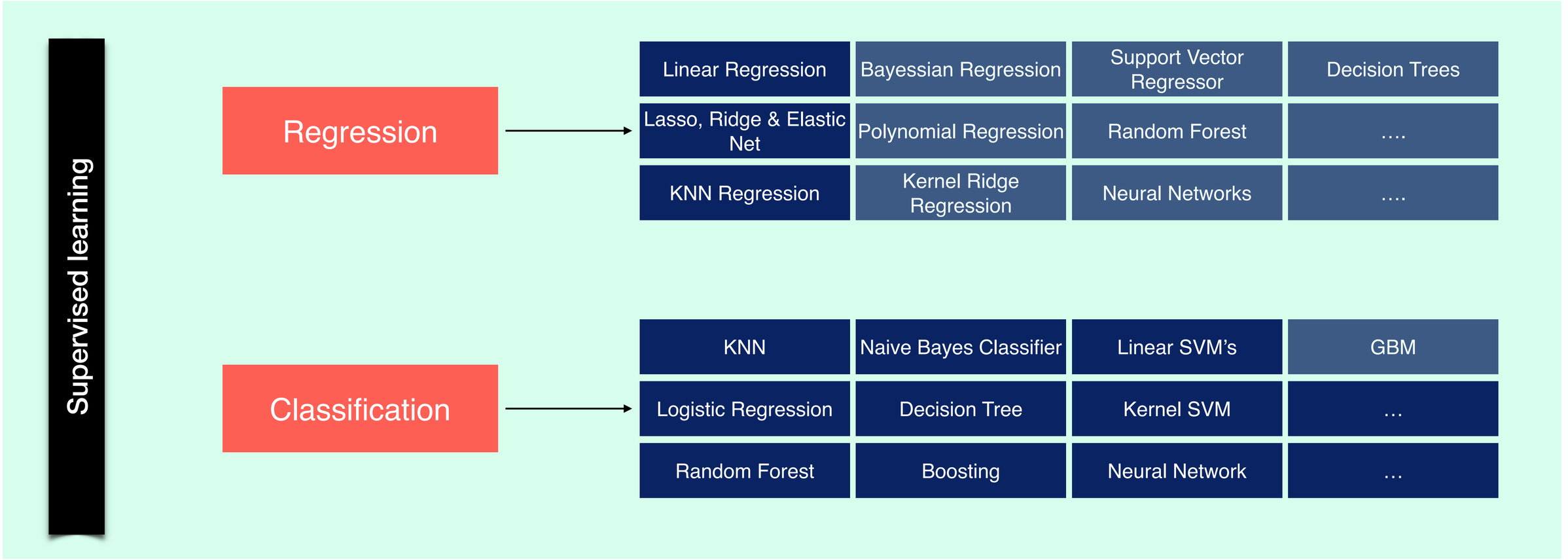
Classification Methods



Classification Methods



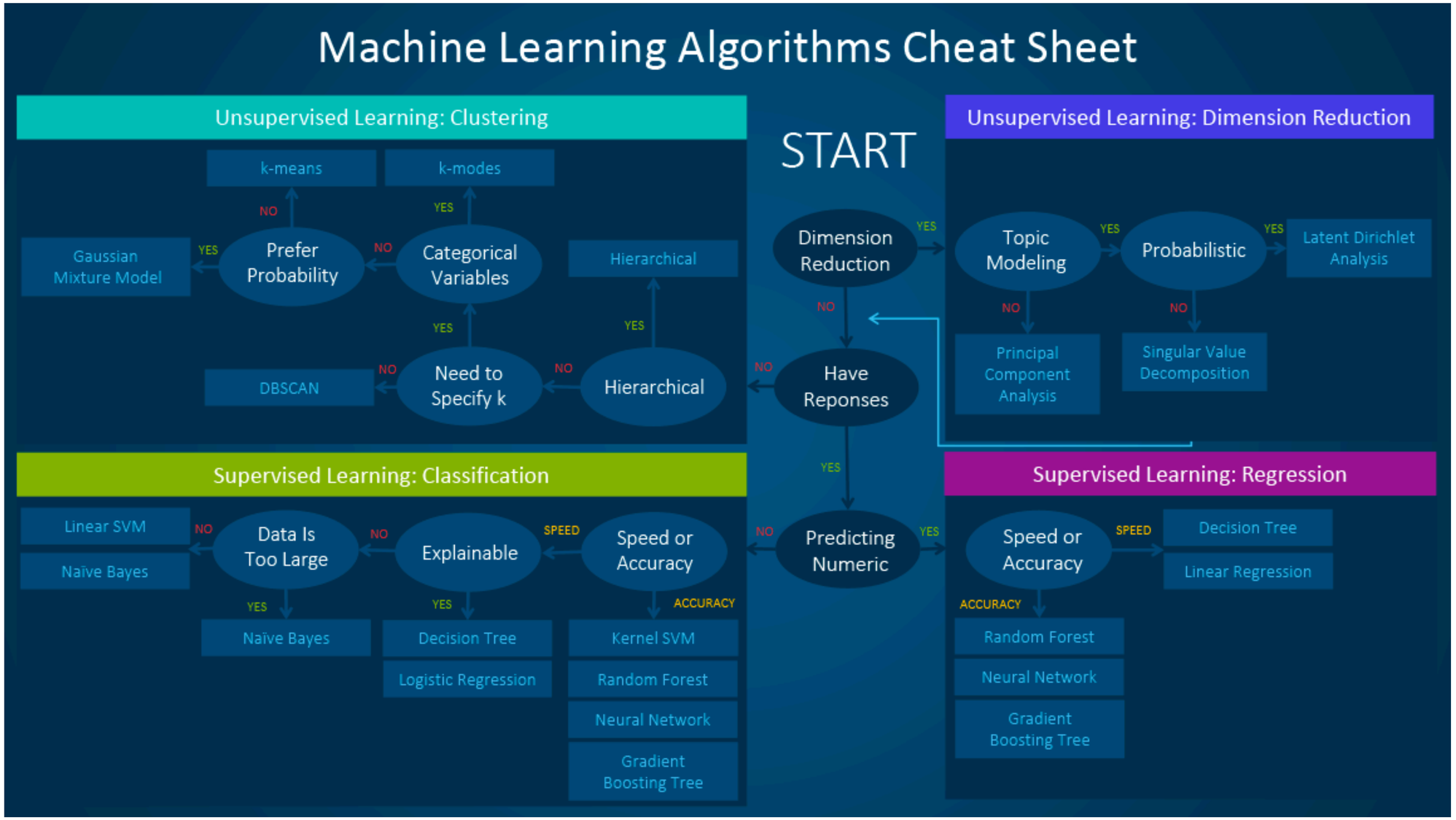
Which one to use?



Which one to use?

- It always **depends on the data**:
  - Distribution
  - Number of training examples
  - Number of features
- But also **depends on the application**.
  - Interpretability vs. Black Box

Which one to use?



Example: Patient cost estimation

Task: Can you accurately predict insurance costs?

Experience:

1338 examples with the following features:

sex: insurance contractor gender, female, male

bmi: Body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight (kg / m ^ 2) using the ratio of height to weight, ideally 18.5 to 24.9

children: Number of children covered by health insurance / Number of dependents

smoker: Smoking

region: the beneficiary's residential area in the US, northeast, southeast, southwest, northwest.

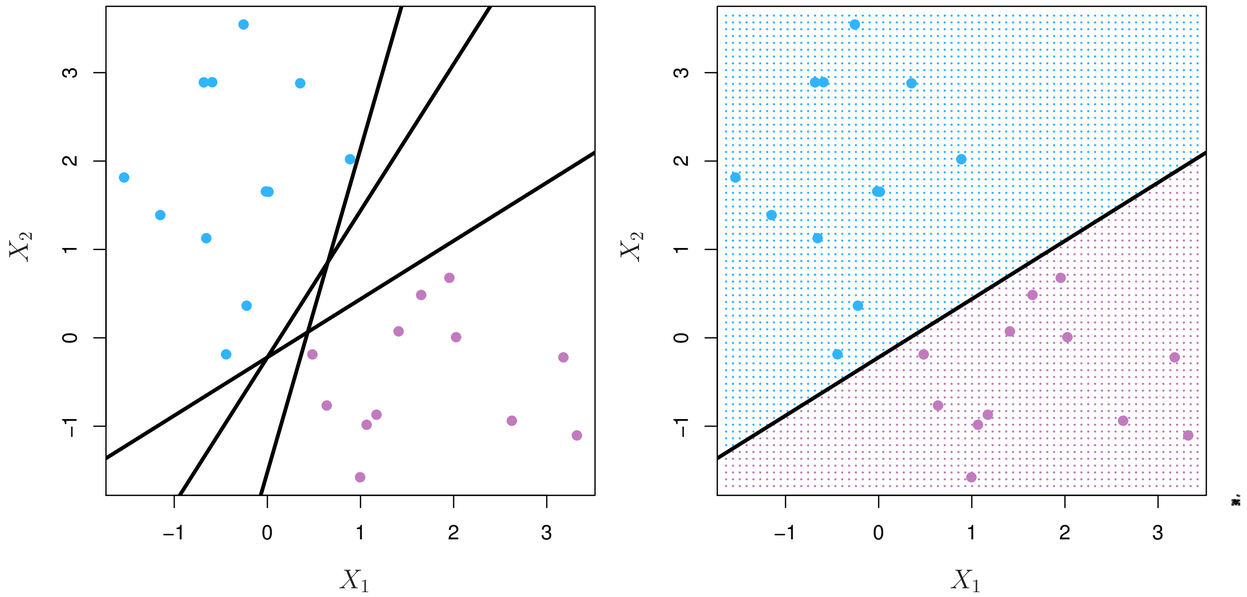
charges: Individual medical costs billed by health insurance

ML MODELS

SVM

Nivell del cos u
Nivell del cos dos
Nivell del cos tres
Nivell del cos quatre
Nivell del cos cinc

Separating Hyperplanes

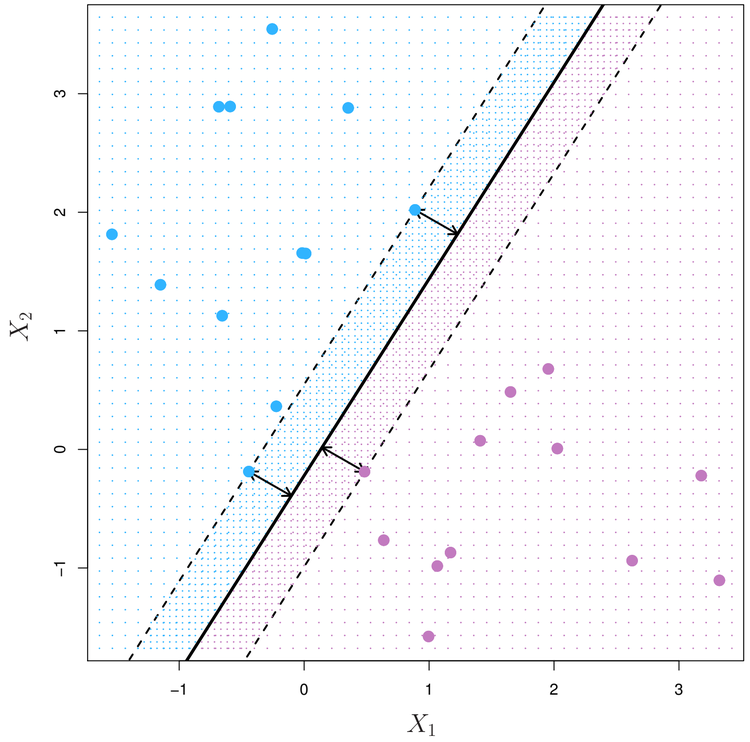


- If $f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.

- If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for magenta, then if $Y_i \cdot f(X_i) > 0$ for all i, $f(X) = 0$ defines a separating hyperplane.

Maximal Margin Classifier

- Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



Constrained optimization problem

$$maximize_{\beta_0,\ldots,\beta_p} M$$

$$subject\ to \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

$$for\ all \quad i = 1,\ldots,N$$

Maximal Margin Classifier

- The previous formulation is equivalent to:

$$minimize_{\beta_0,\ldots,\beta_p} \|\beta\|^2$$

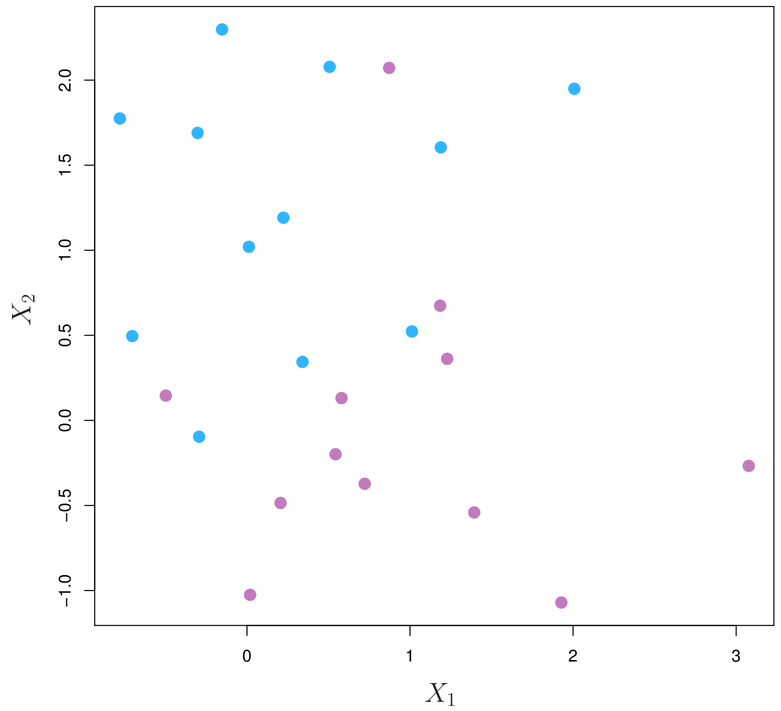
$$subject\ to\ y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq 1$$

$$for\ all\ \ i = 1,\ldots,N$$

- This is an easy problem (the objective function is differentiable and convex) that can be solved using standard software.

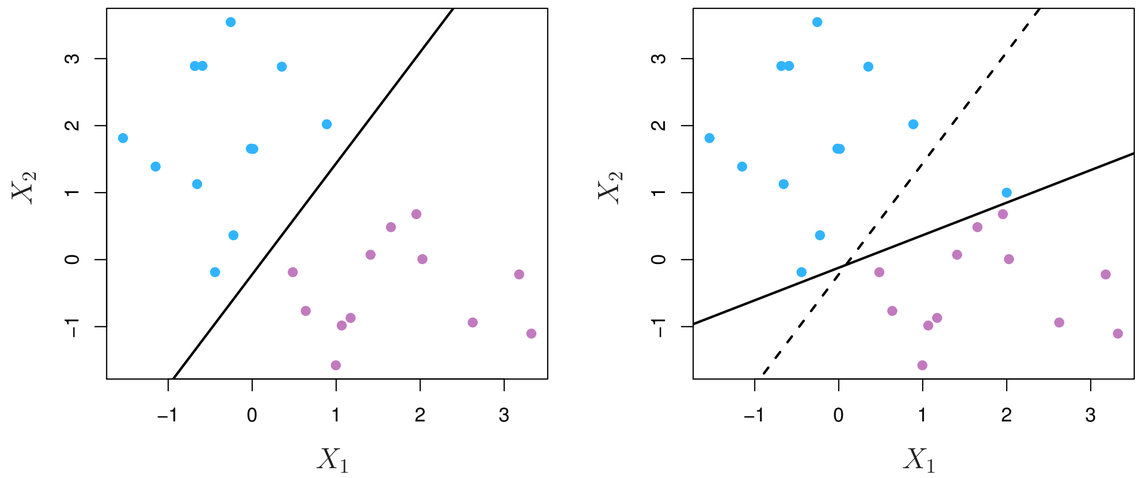
Non-separable Data

- Sometimes the data is not separable by a linear boundary



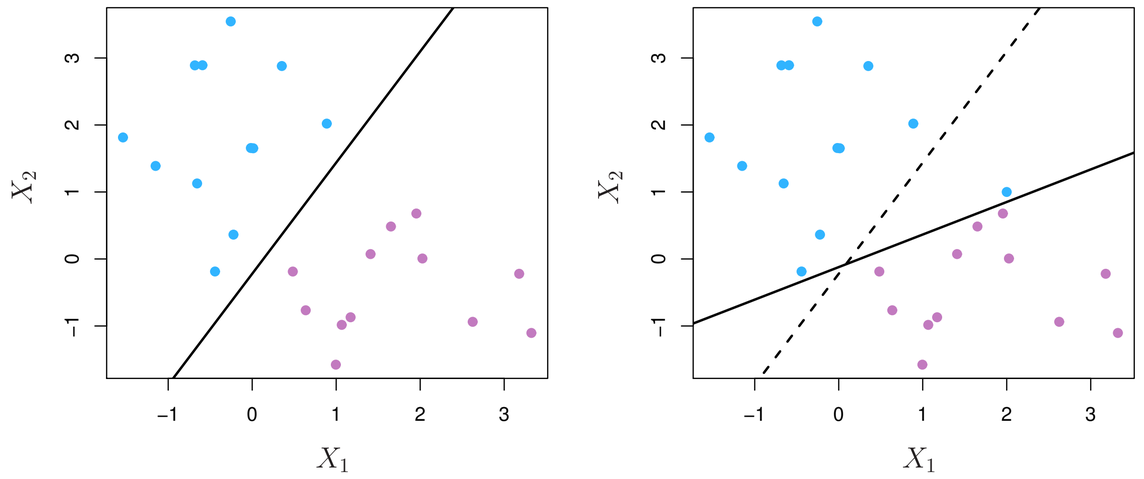
- This is often the case, unless $N < p$

Noisy Data



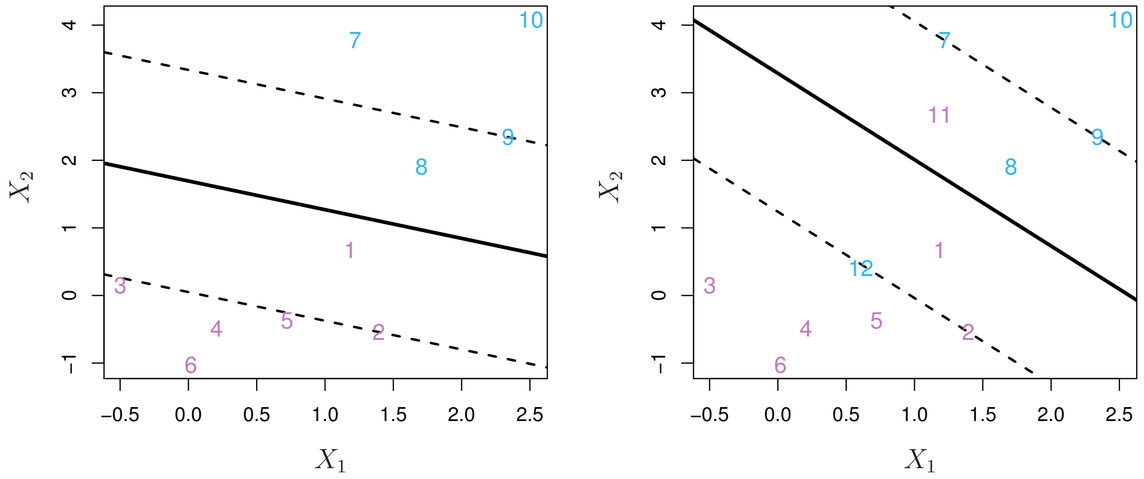
- Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

Noisy Data



- Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier. The **Support Vector Classifier** maximizes a **soft margin**.

Support Vector Classifier



$$maximize_{\beta_0,\ldots,\beta_p,e_1,\ldots,e_n} M \quad subject\ to \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

$$e_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$$

The case of non separable data

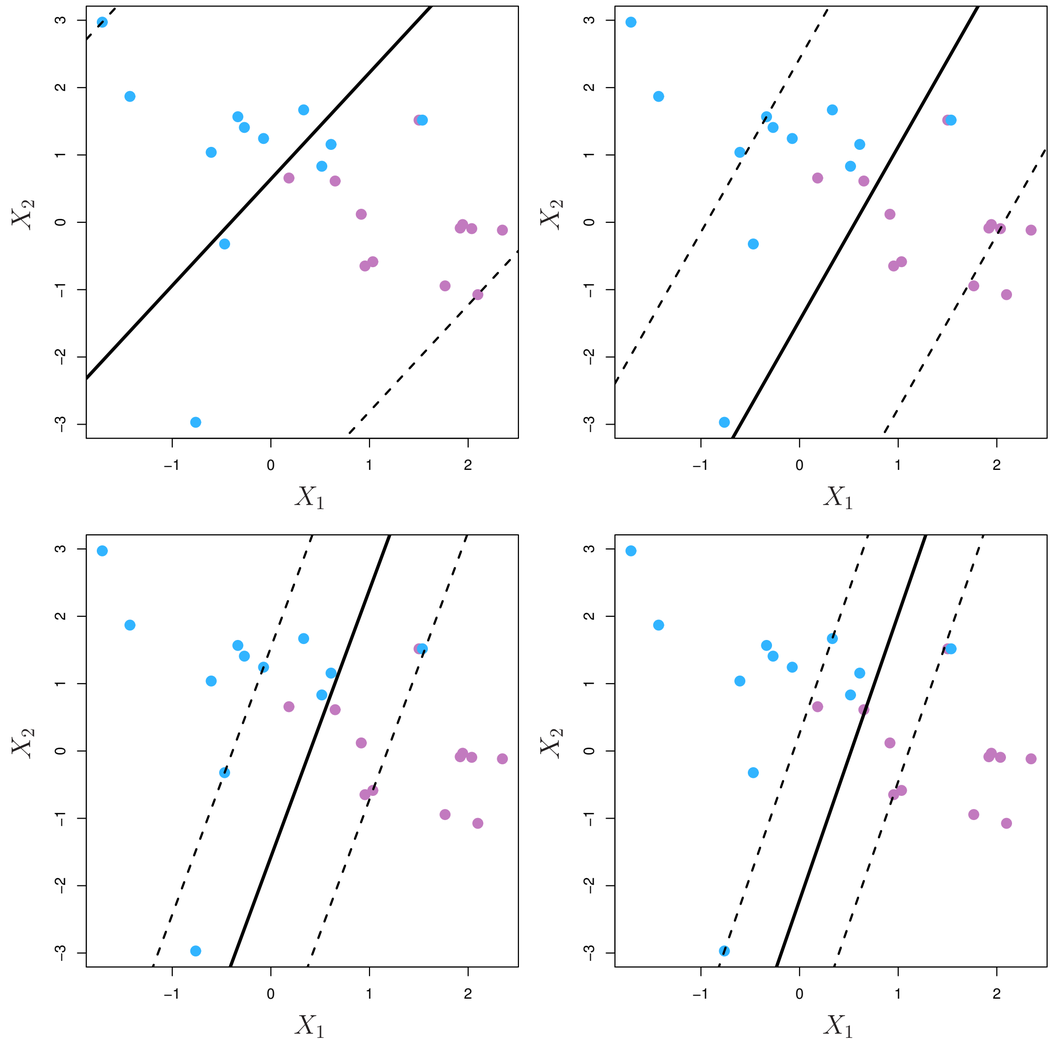
- This SVM variant takes a parameter C that determines how hard points violating the constraint should be penalized. This parameter appears in the objective function of the problem, which now is formulated as:

$$minimize_{\beta_0..\beta_p,\xi} \frac{1}{2}\|\beta\|^2 + \frac{C}{N}\sum_i \xi$$

$$s.t.\ y_i(\beta \cdot x_i) \geq 1 - \xi_i, \xi \geq 0, \forall i \in \{1,...,N\}$$

- Large C means high penalty, and in the limit $C \to \infty$ we obtain the separable case.

C is a regularization parameter

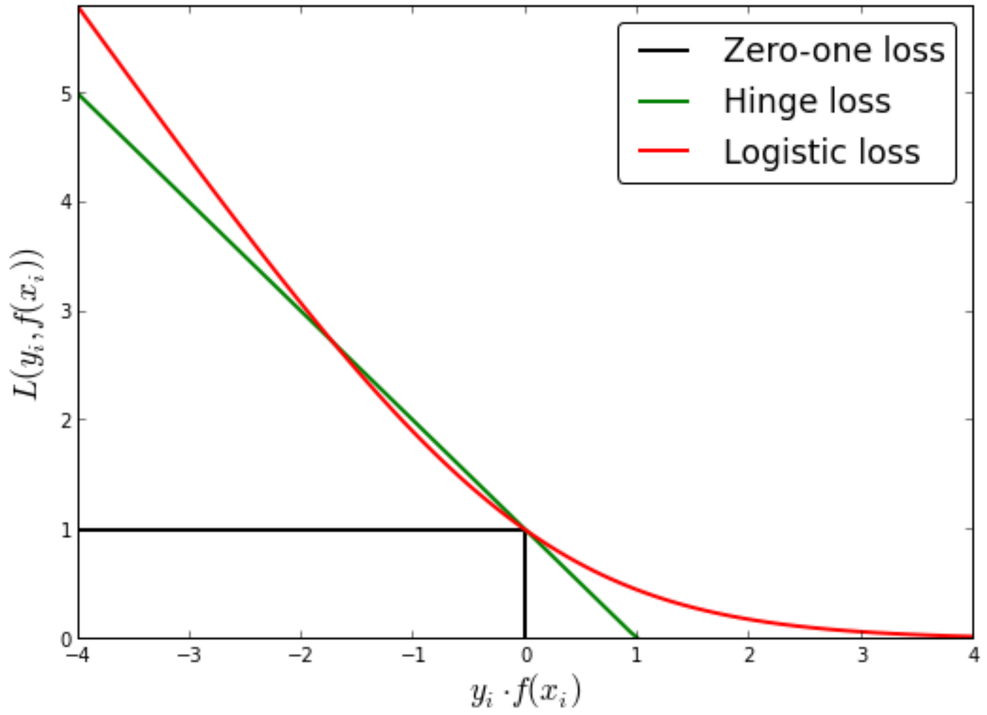


The case of non separable data

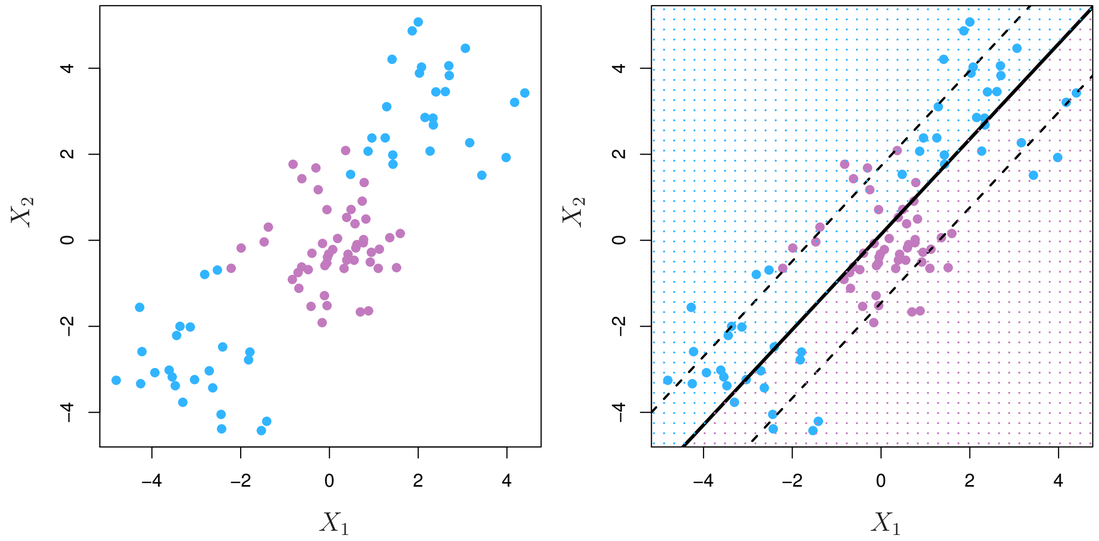
- Observe that if $y_j(\beta \cdot \mathrm{x}_j) \geq 1$, then $\xi_j = 0$ and in this case there is no contribution to the penalty term, but if margin $y_j(\beta \cdot \mathrm{x}_j) < 1$, $\xi_j > 0$ and the penalty terms increases $\frac{C}{N}\xi_j$:

$$\xi_j = max(0, 1 - y_i(\beta \cdot \mathrm{x}_j))$$

Loss Functions



Linear boundary can fail



What to do?

Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1, X_1^2, X_1X_2, X_1X_2^2, \ldots$ Hence go from a **p-dimensional** space to a $M > p$ **dimensional space**.

- Fit a support-vector classifier in the enlarged space.

- This results in non-linear decision boundaries in the original space.

The Kernel trick

- The original SVM algorithm, proposed by Vladimir Vapnik in 1963, was a linear classifier, but there is a way (**the kernel trick**) to design non-linear classifiers.

- The trick is based on the observation that all data terms are exclusively used to compute dot products:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- This fact suggests a way to compute SVM solutions in higher dimensional spaces: to find a function $k(x_i, x_j)$, called the kernel, that corresponds to the dot product of x_i and x_j in such a space.

The Kernel trick

- In this case, the problem can be written as:

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i^T x_j)$$

and the optimization problem does not change.

Kernels and Support Vector Machines

- If we can compute inner-products between observations, we can fit a SVM classifier.
- Some special kernel functions can do this for us. E.g

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

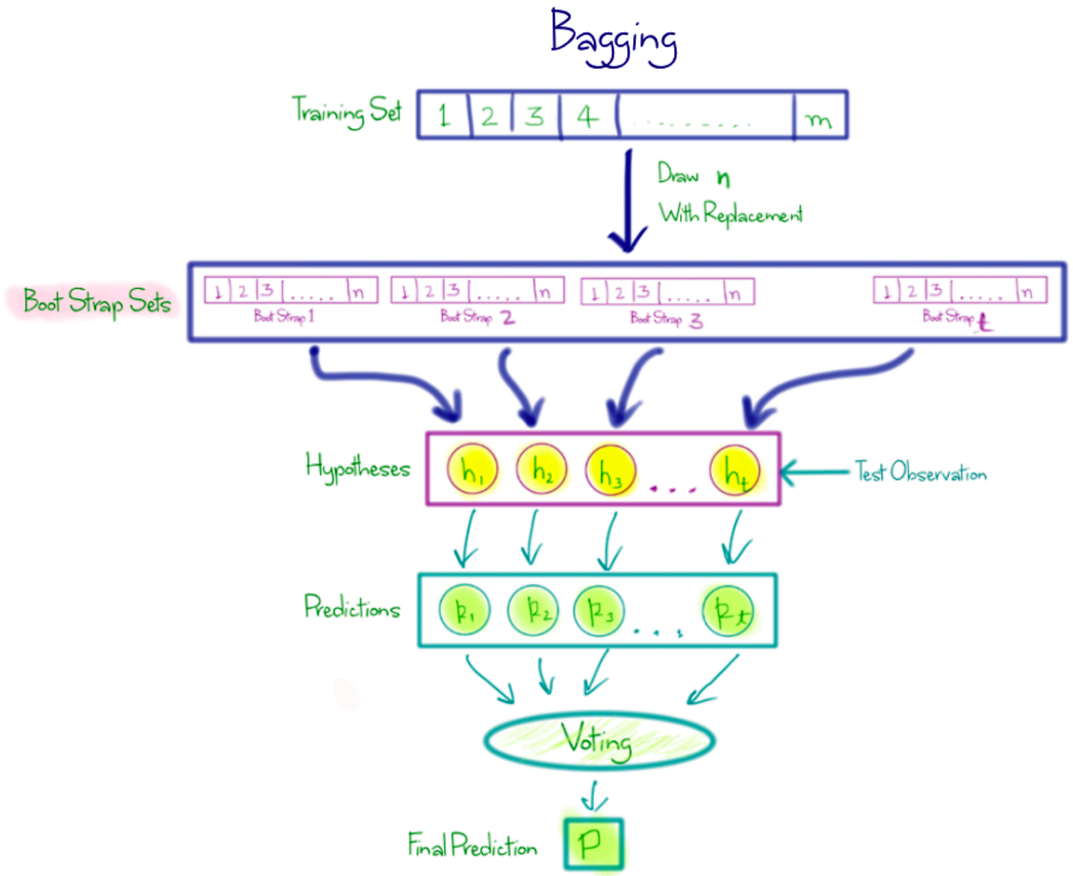
- computes the inner-products needed for d dimensional polynomials $\binom{p+d}{d}$ basis functions!

ML MODELS

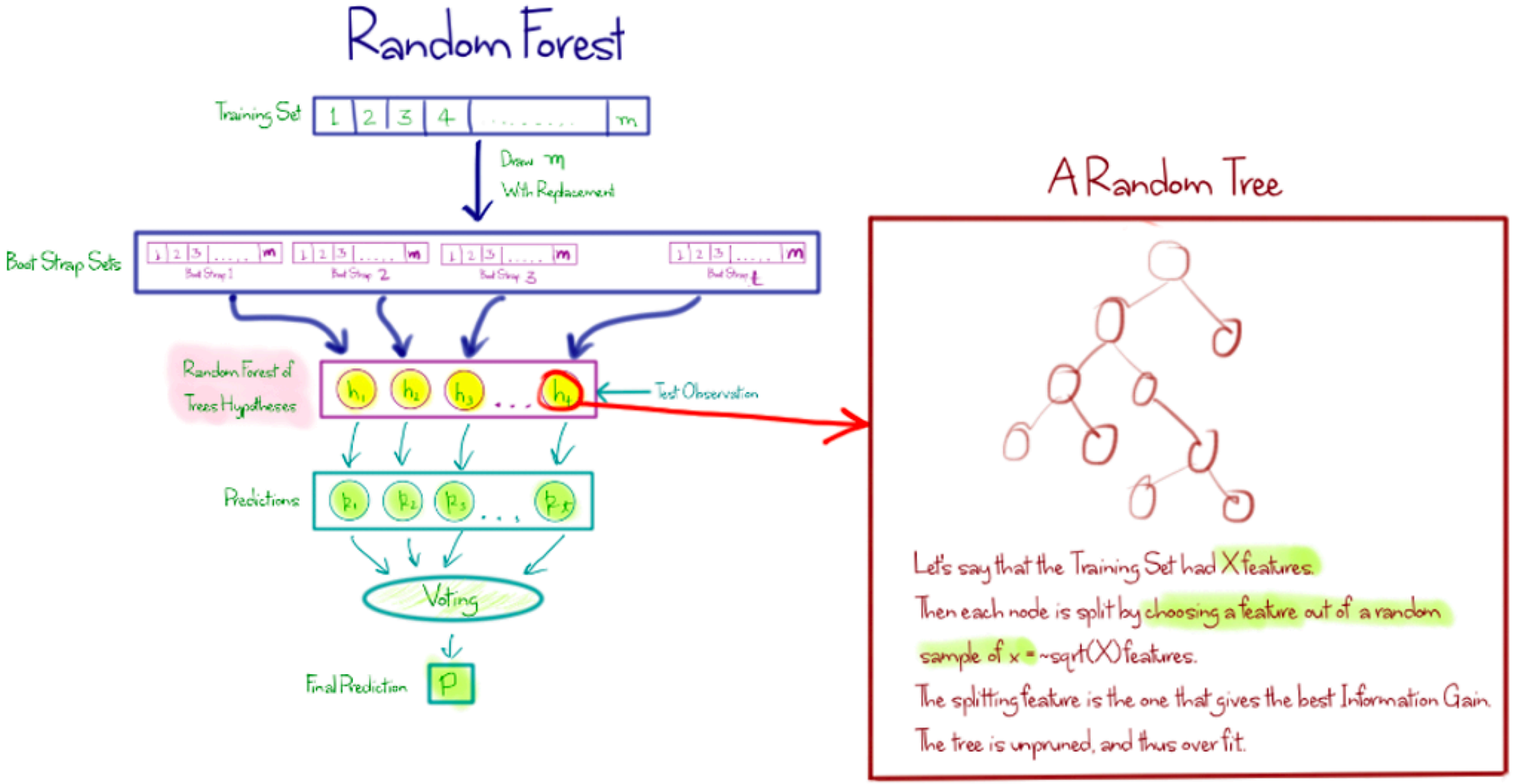
Ensembles

Nivell del cos u
Nivell del cos dos
Nivell del cos tres
Nivell del cos quatre
Nivell del cos cinc

Bagging


Bagging
Training Set
1 2 3 4 ..... m
Draw n
With Replacement
Boot Strap Sets
Boot Strap 1
Boot Strap 2
Boot Strap 3
Boot Strap t
Hypotheses
Test Observation
Predictions
Voting
Final Prediction
P

Random Forest



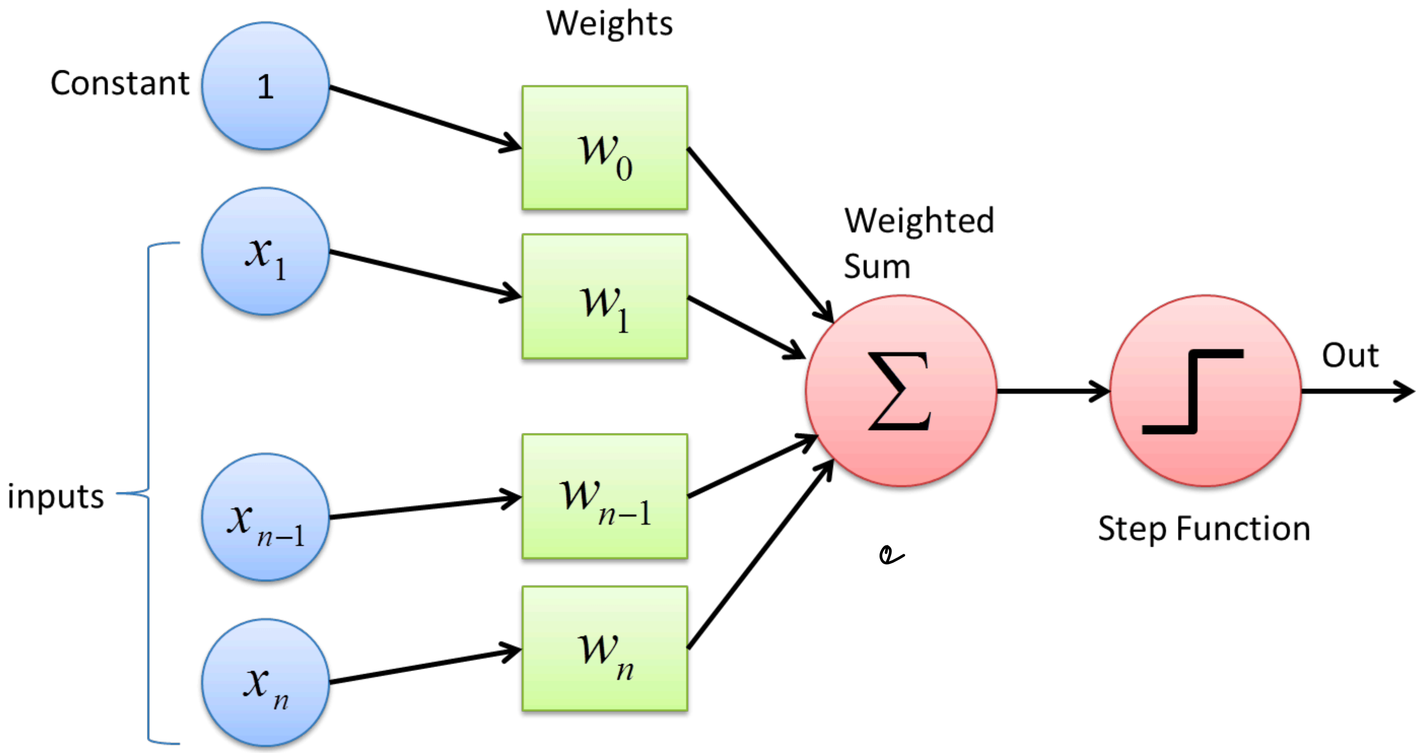
ML MODELS

Neural Networks

Nivell del cos u
Nivell del cos dos
Nivell del cos tres
Nivell del cos quatre
Nivell del cos cinc

What is a Neural Network

- It is a system **biologically inspired** that tries to emulate **human brain**.
- **Why** is it a good idea to try to **emulate** the **brain** when solving a recognition task?

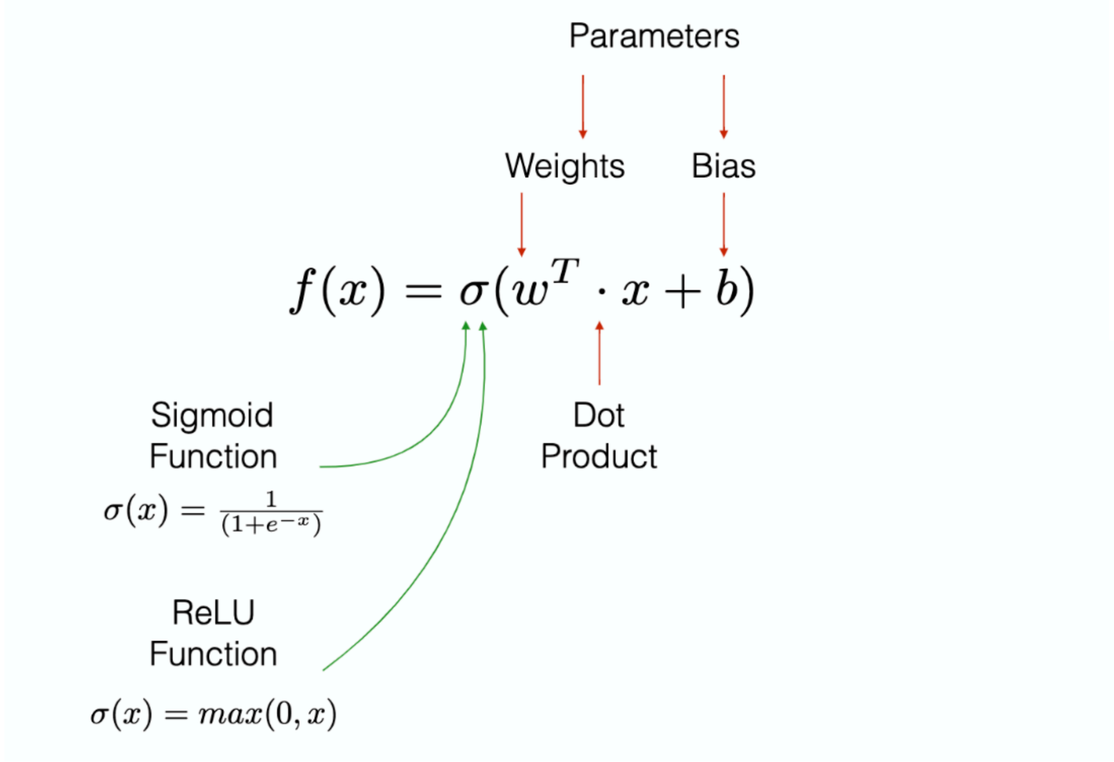


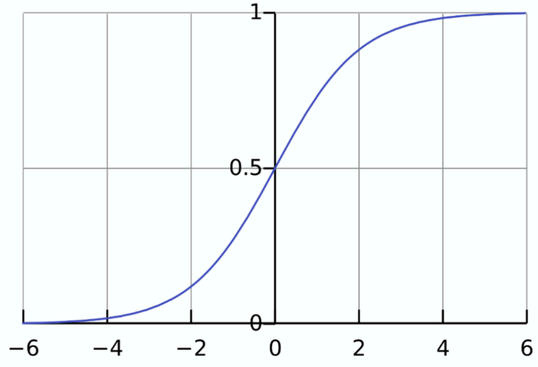
Perceptron

Perhaps the first AI model

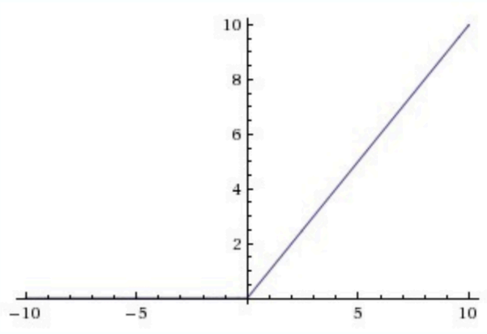
Neural Networks

1 Layer Neural Net model

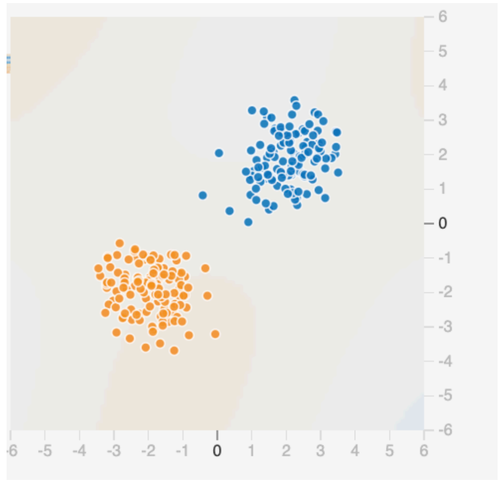
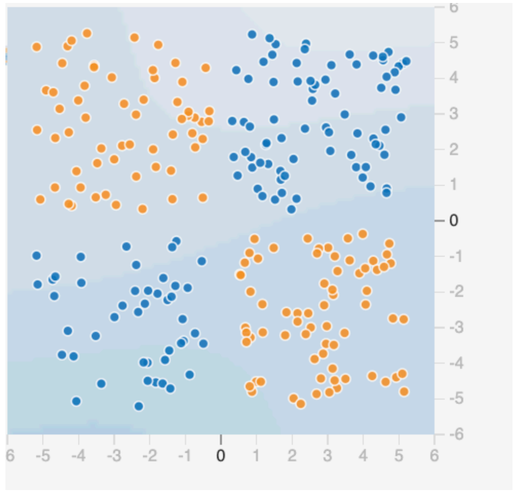
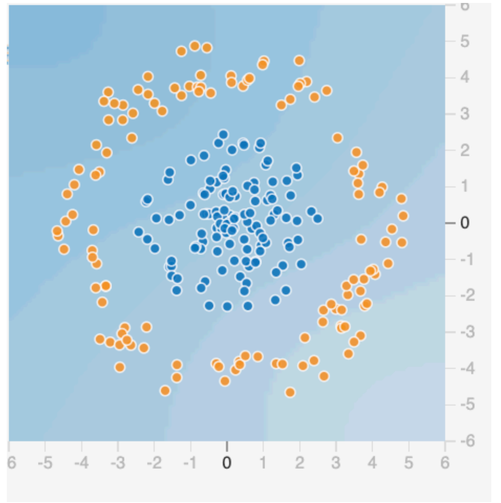




$\sigma(x) = \frac{1}{(1+e^{-x})}$

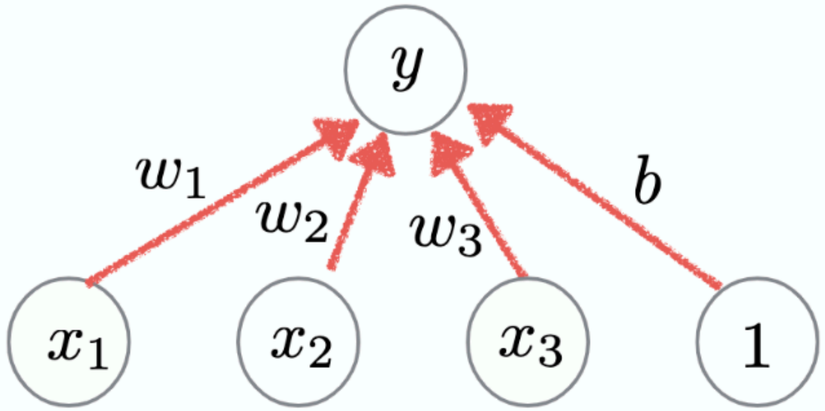


$\sigma(x) = max(0, x)$

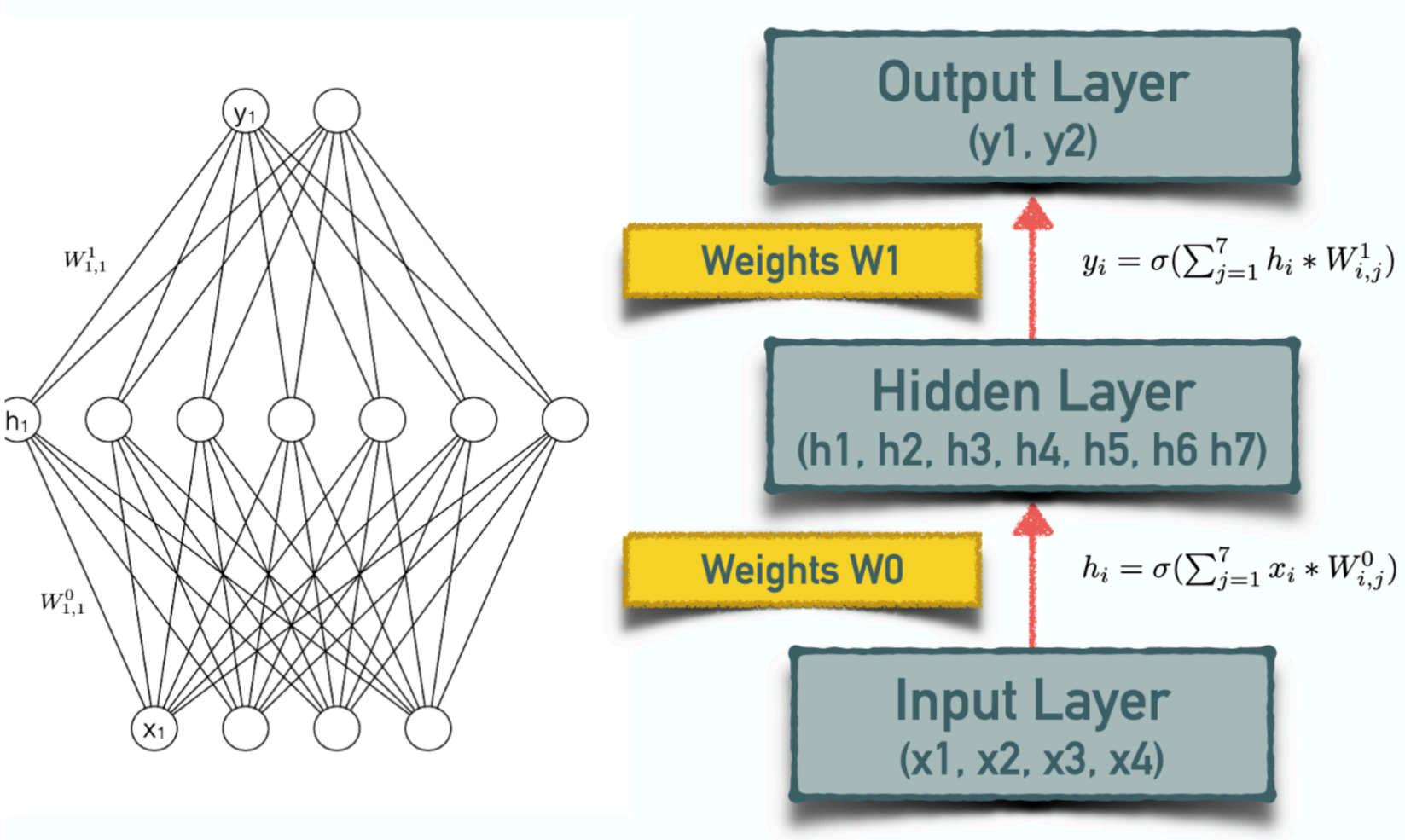




1 Layer Neural Net model

$$f(x) = \sigma(w^T \cdot x + b)$$



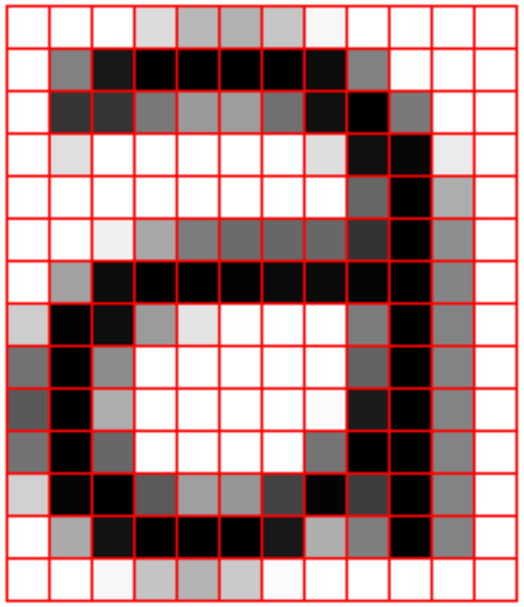
Graphical Representation



2 Layers Neural Net model

Neural Networks for Images?

An image is a matrix of size **mxnxc** pixels



1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0

Toy Dataset: Fashion MNIST

Toy Dataset: Fashion MNIST

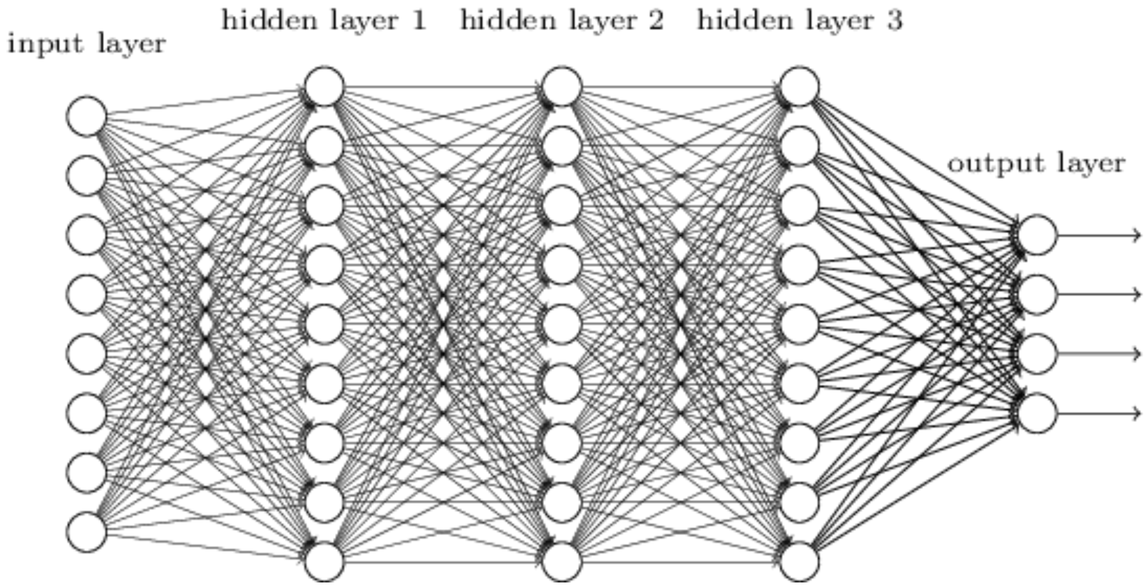
```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
```

this is equivalent to this

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(300, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(10, activation="softmax")
])
```

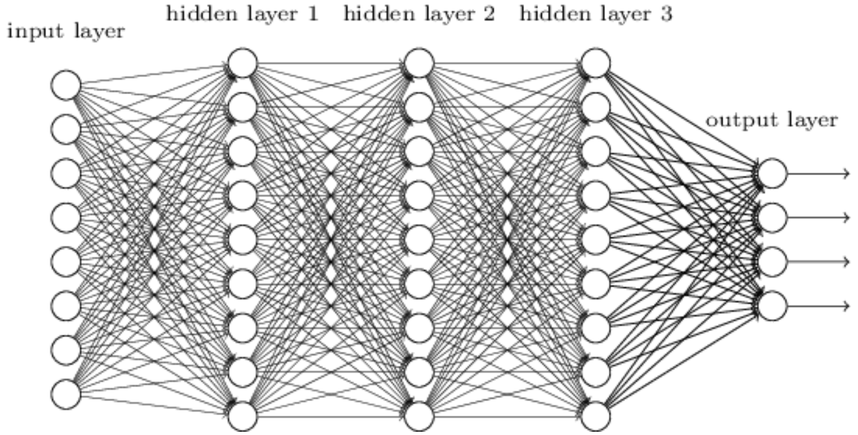
hands on!

Neural Networks for Images?

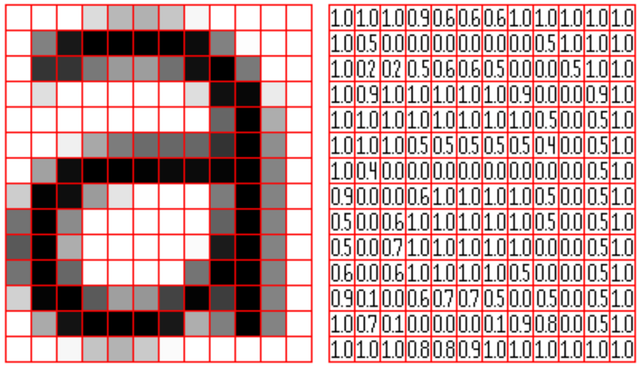


How many parameter does this MLP has?

Neural Networks for Images?



What happens if your data is an image like this one?



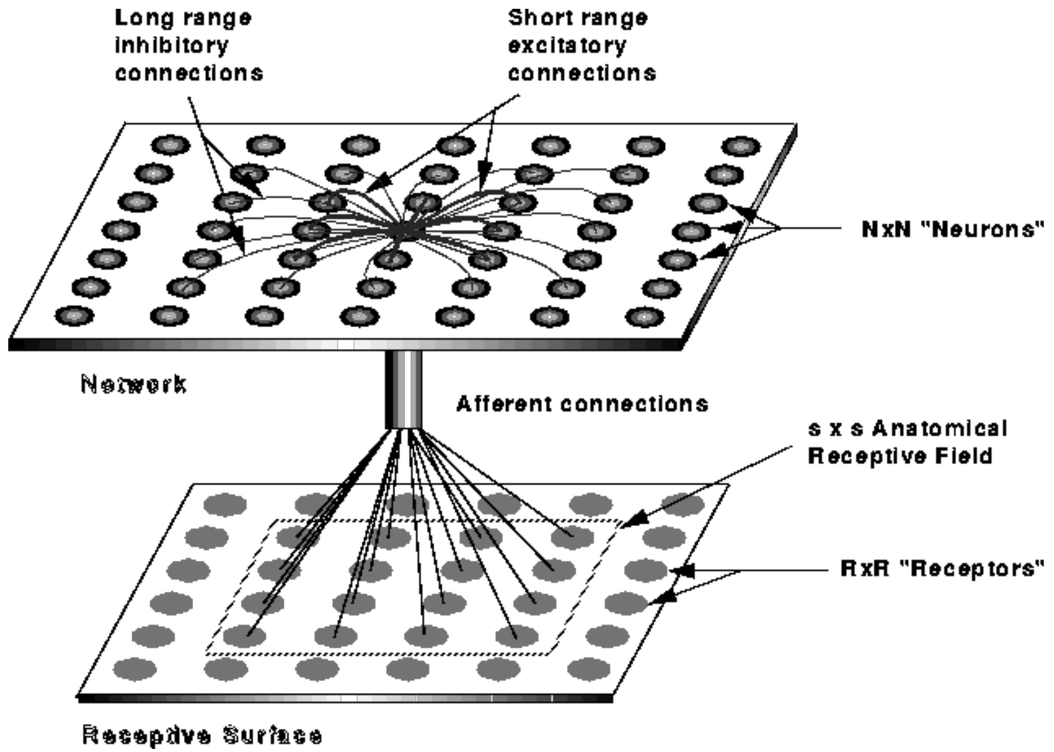
1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0			
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0		
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.0	0.5	1.0	1.0		
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.0	0.9	1.0		
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0			
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0			
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0			
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0			
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0			
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0			
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0			
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0			
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0			
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0			

Neural Networks for Images?

Neural Networks for Images

- Input is a standard vector of size N x M x C
  - Imagine an medium resolution color image of 256x256 pixels
  - If we think on a Multi Layer Perceptron with just one hidden layer of 256 neurons + an output layer of 1 neuron it will have more than **48 million** parameters.
    - **Does it make sense? Can we do it better?**

Local Receptive Fields



David Hunter Hubel and Torsten Nils Wiesel, 1968

But, in an image:

A dog can appear **anywhere** in the image!

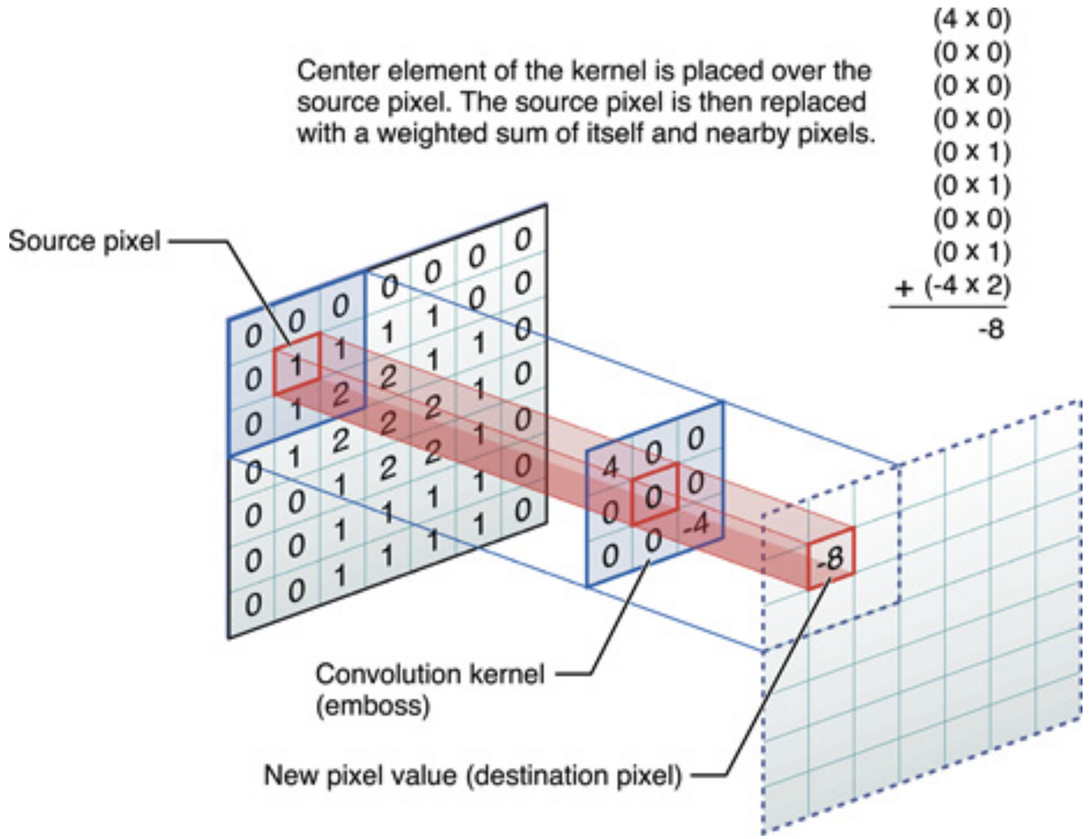
Doesn’t matter where they appear,
they look similar anywhere!

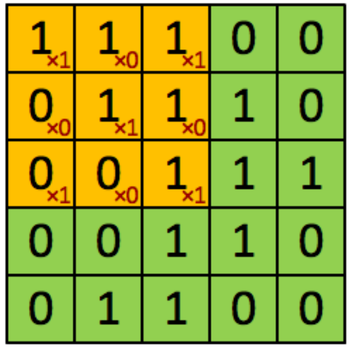
Convolutional Neural Networks (CNNs)

- Three main ideas:

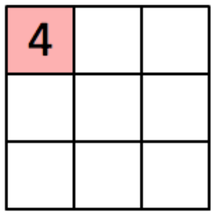
    1. **local receptive fields**,
    2. **shared weights**,
    3. **sub-sampling**

What is an image convolution?





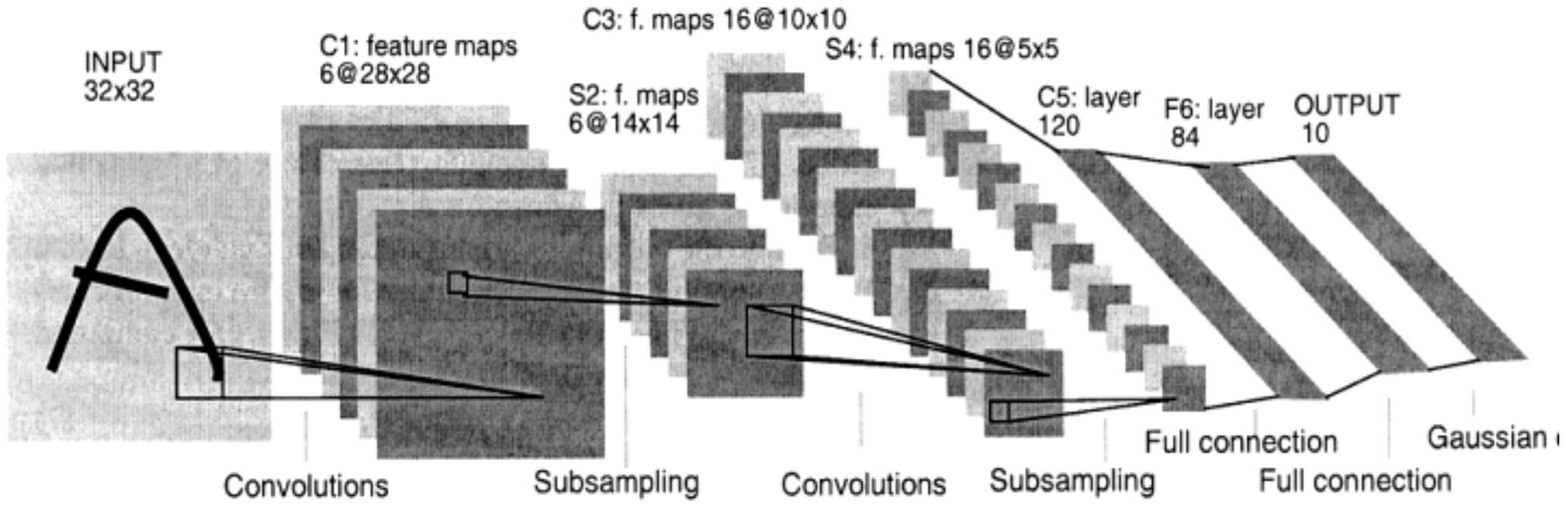
Image



Convolved Feature

What is an image convolution?

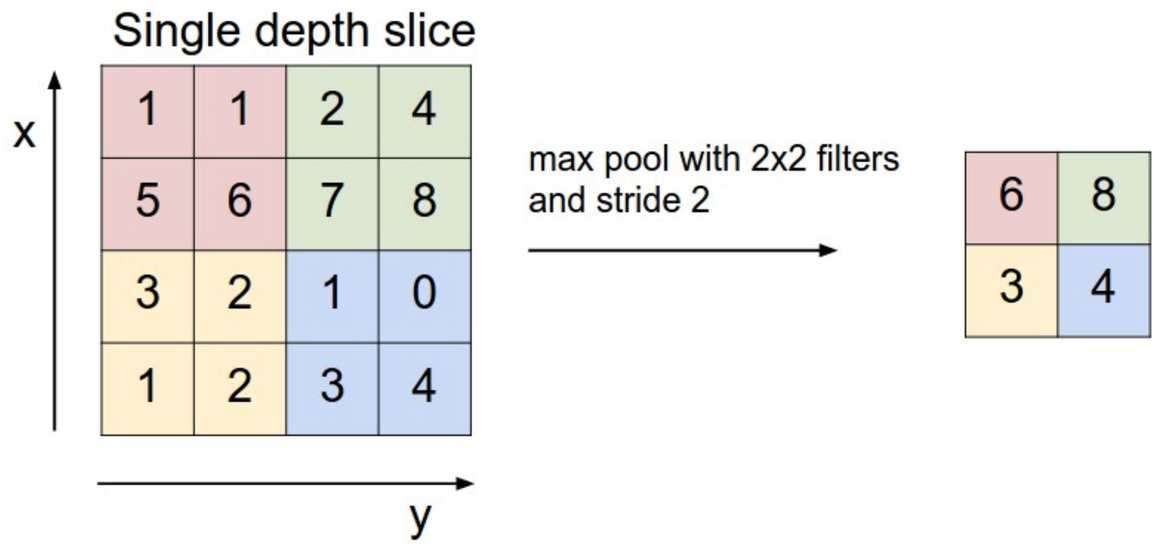
"Nothing New"



LeCun et al. 1998

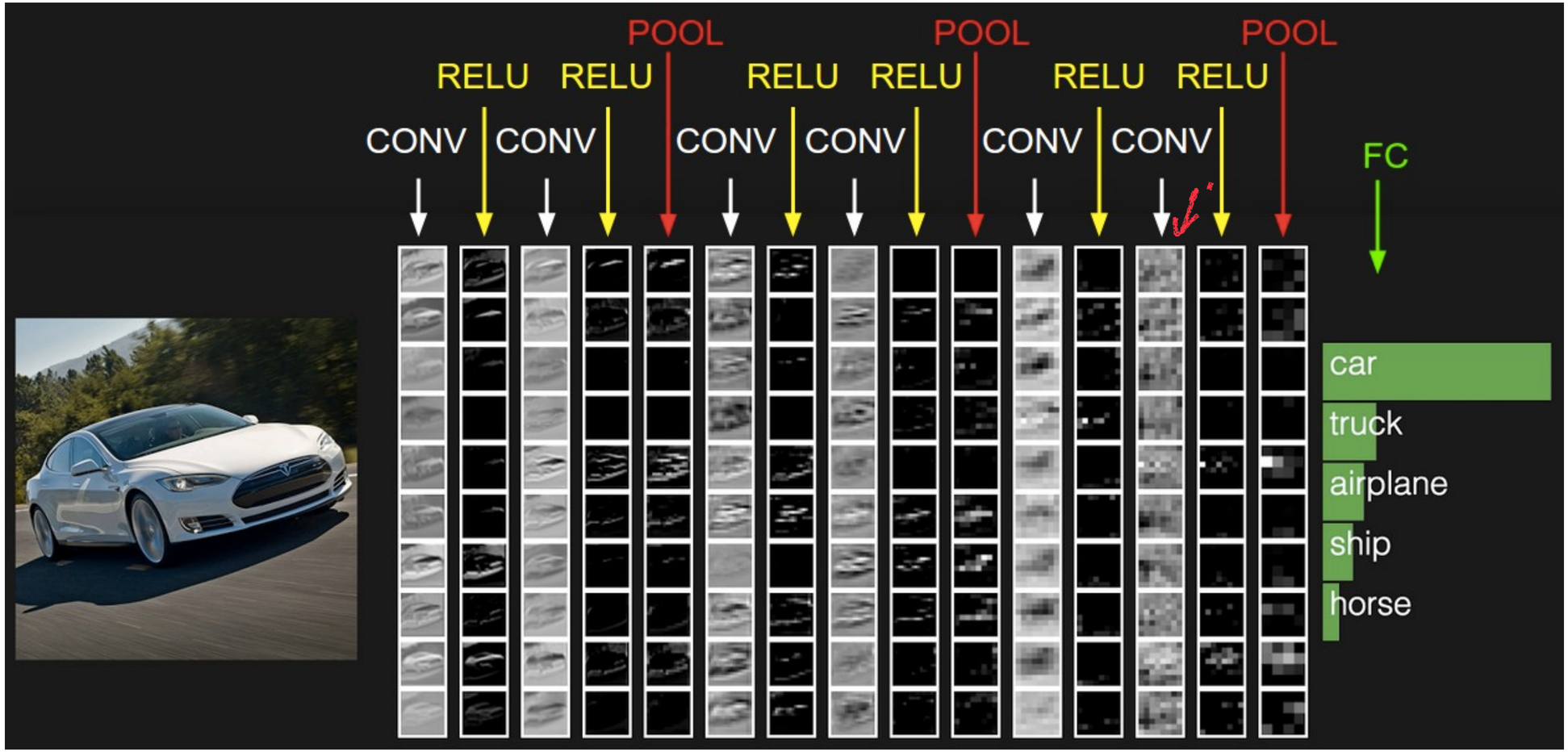
Max pooling

Pooling is a way of sub-sampling, i.e. reducing the dimension of the input (or at some hidden layer).
It is usually done after some of the convolutional layers



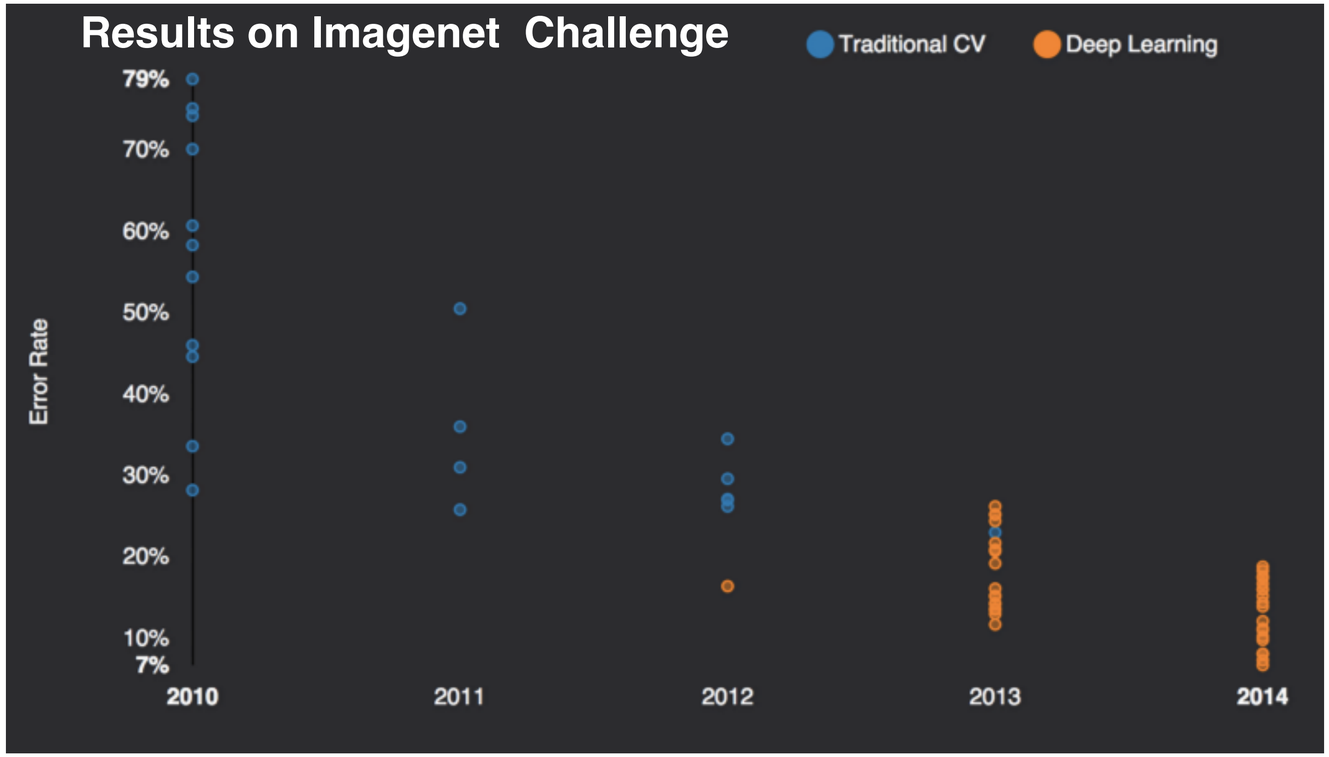
But it is also useful since it provides a form of translation **invariance**

Finally..


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Convolutional Neural Networks (CNNs)

In computer Vision the breakthrough resulted in 2011 when Ciresan et.al introduced an algorithm to train these networks by using graphical cards (GPUs)



AlexNet

Similar framework to LeCun'98 but:

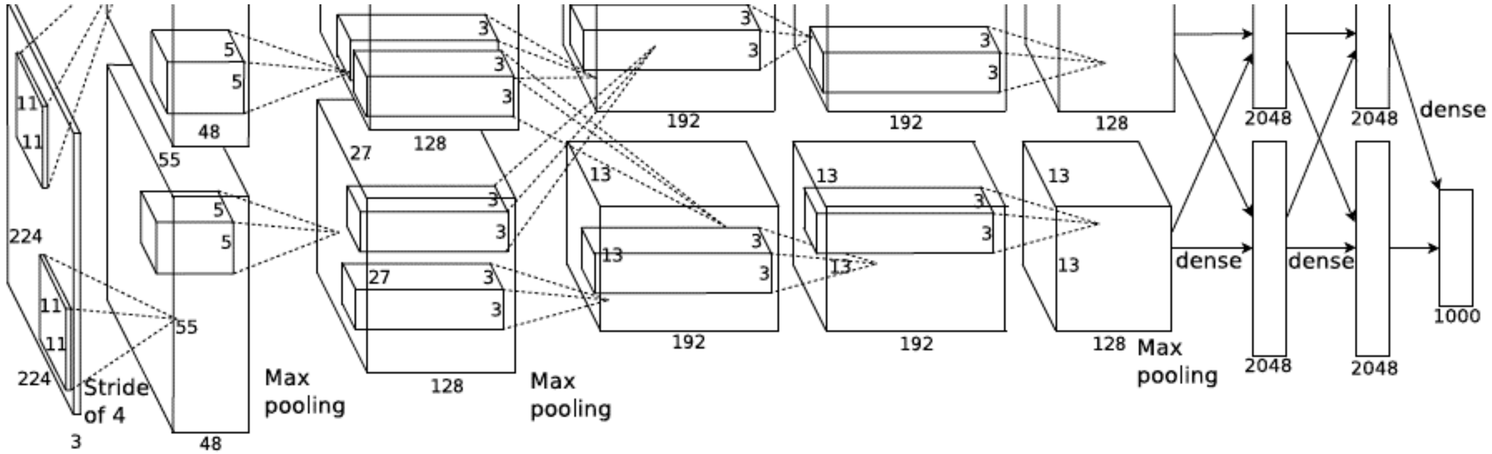
Bigger model:

7 hidden layers, 650.000 units, 60 million parameter

More Data:

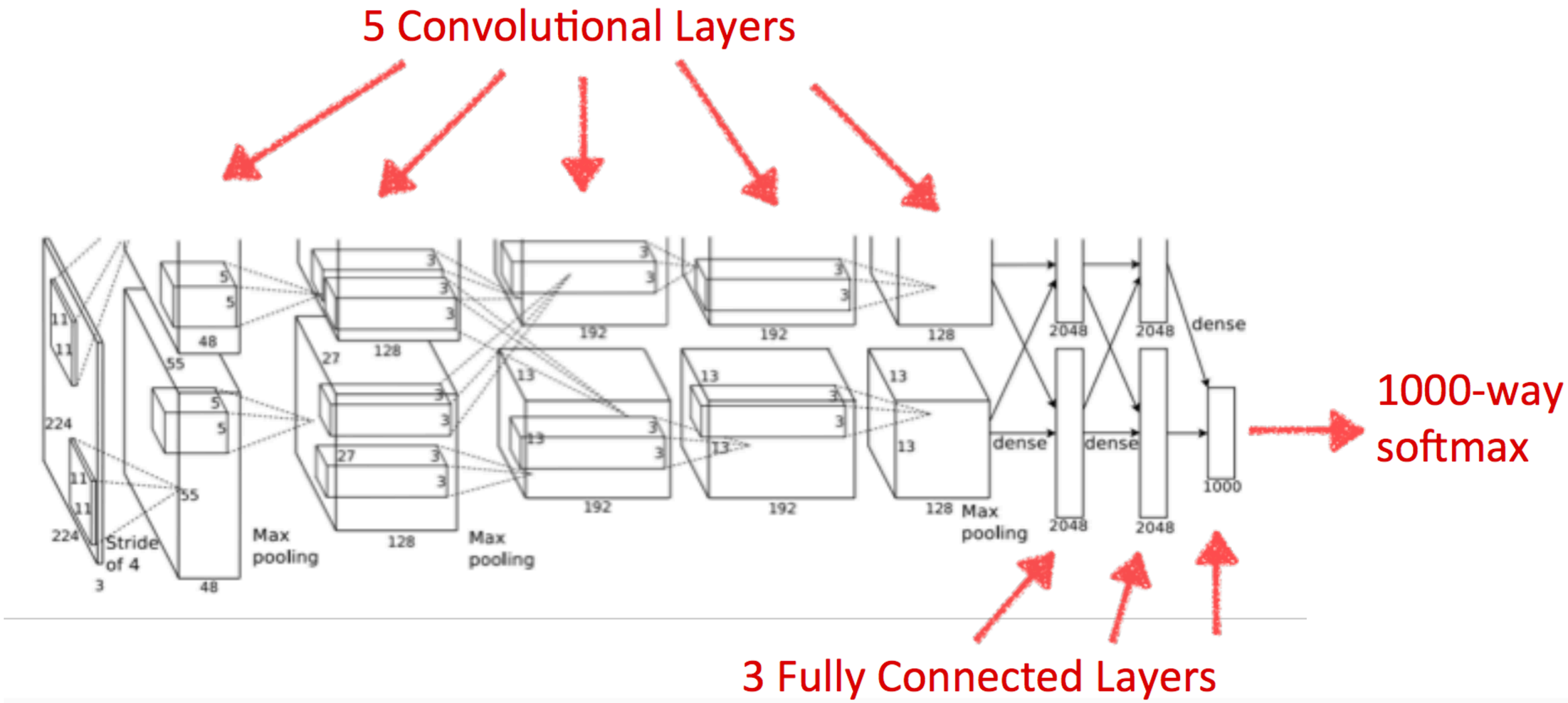
10^6 vs 10^3 images

GPU implementation (50x speedup over CPU)

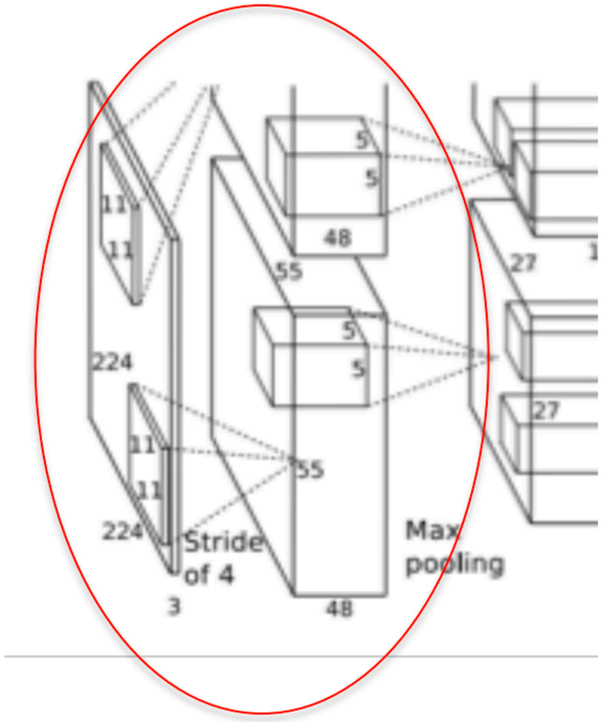


ImageNet Classification with Deep Convolutional Neural Networks. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

AlexNet

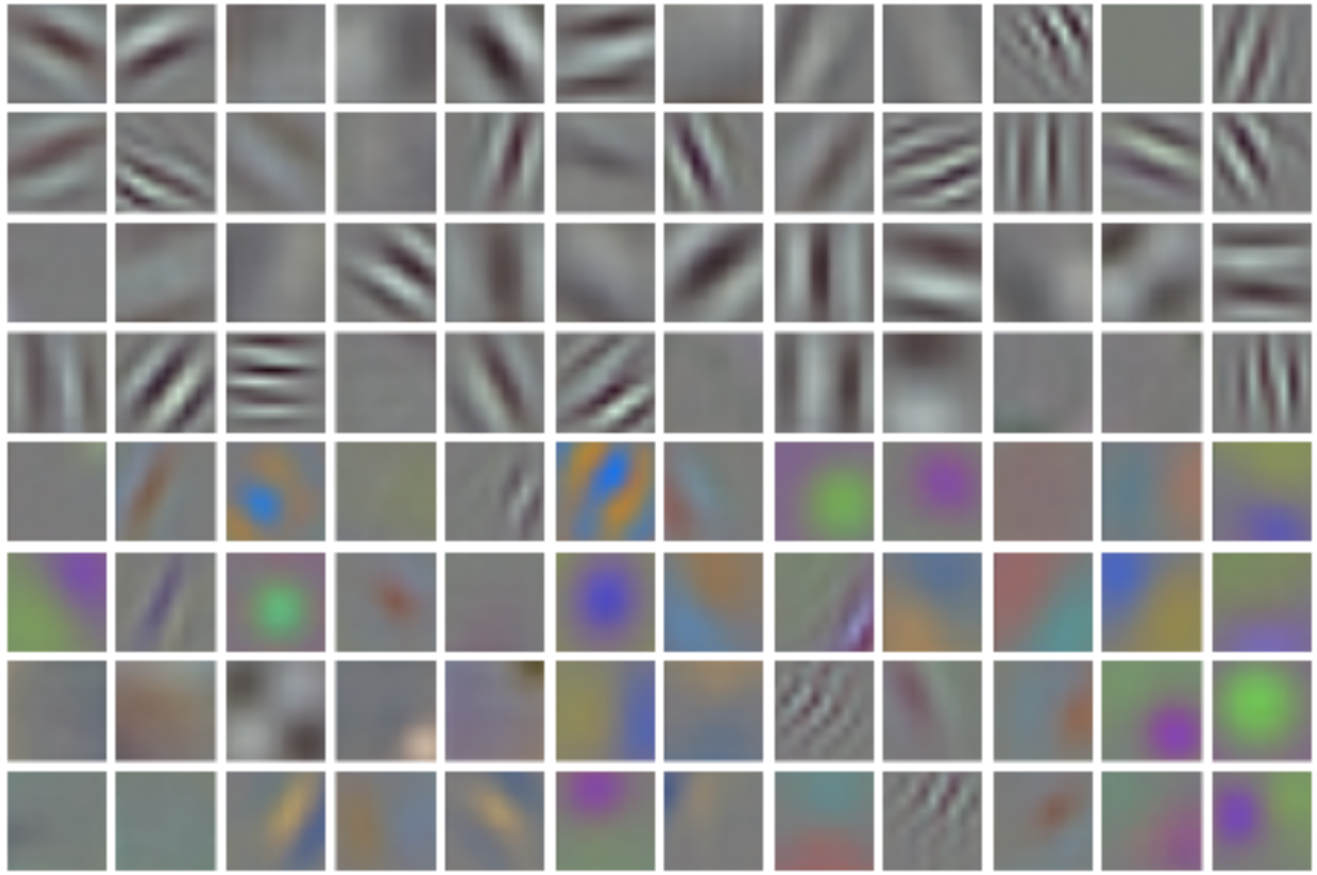

5 Convolutional Layers
1000-way softmax
3 Fully Connected Layers
Stride of 4
Max pooling
Max pooling
Max pooling
dense
dense
dense
224
224
3
55
55
48
48
27
27
128
128
128
192
192
192
192
13
13
13
13
13
2048
2048
2048
2048
1000

AlexNet
1st Convolution Layer

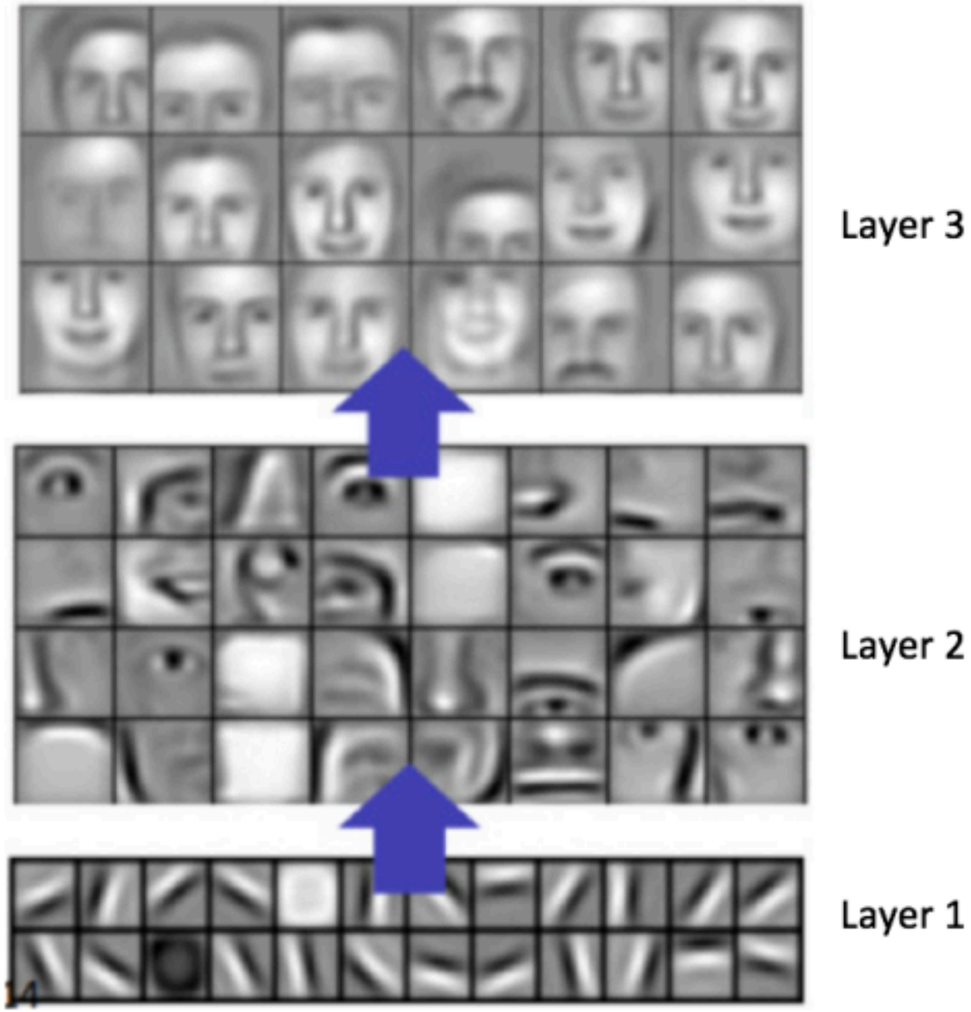


- Images: 227x227x3
- F (receptive field size): 11
- S (stride) = 4
- Conv layer output: 55x55x96

Alexnet 1st Conv Filters

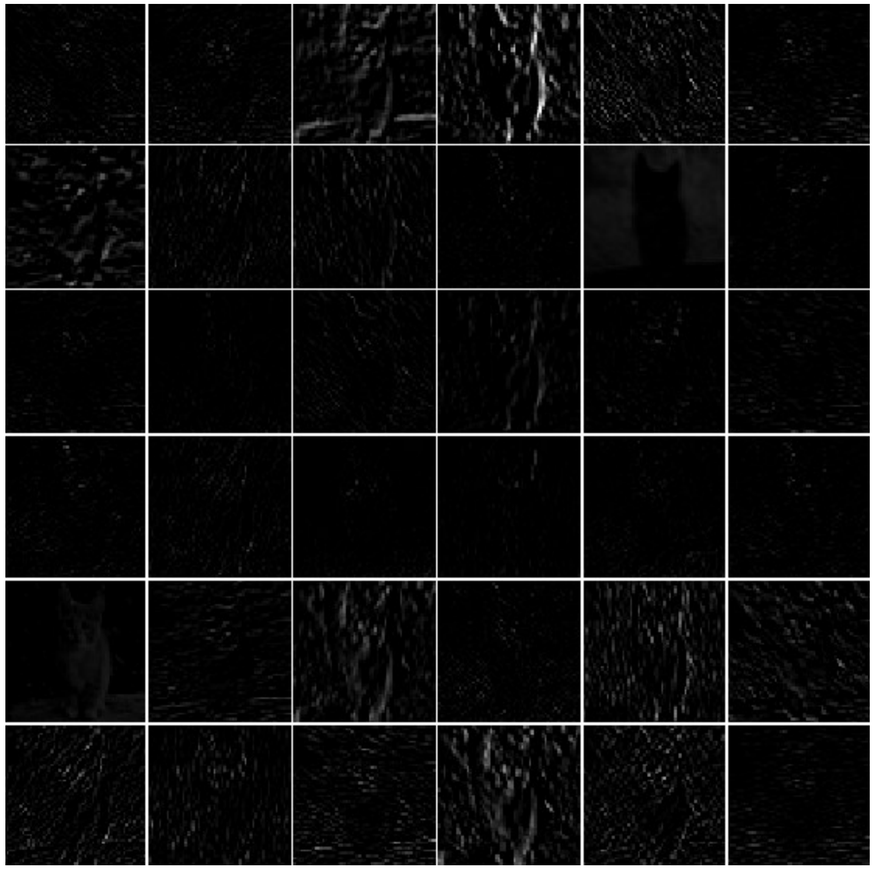


Alexnet

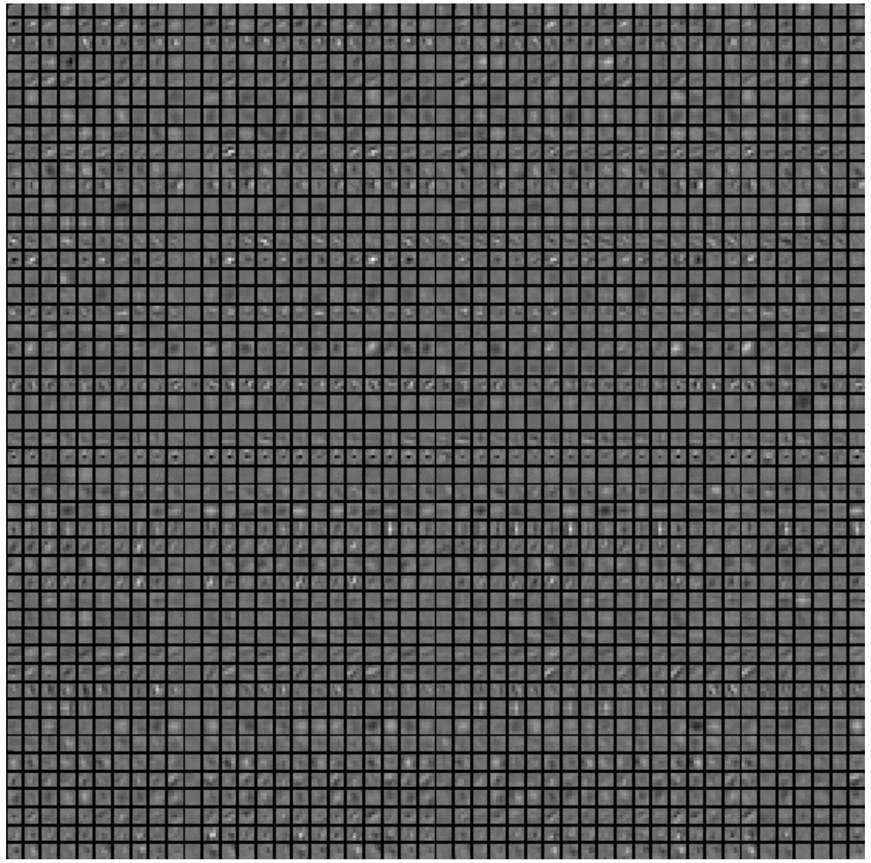


Alexnet

Feature Map Conv1



Conv2



Training a CNN

- Back-propagation + stochastic gradient descent with momentum
- Dropout
- Data Augmentation
- Batch Normalization
- Initialization
  - Transfer Learning
- The make use of **GPU's** in order to optimize the weights.

Reducing Overfitting

Data Augmentation

60 million parameters, 650,000 neurons
-> overfits a lot

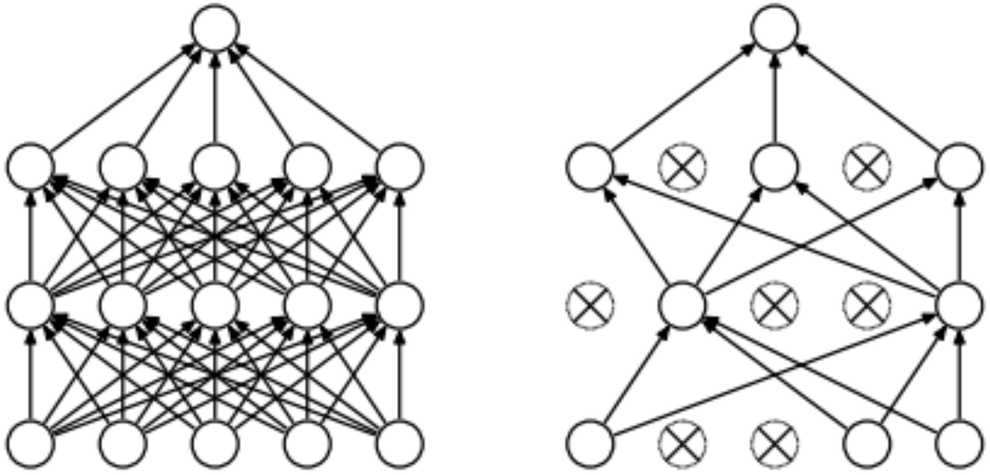
Crop 224x224 patches (and their horizontal reflections)

Data Augmentation at test
average the predictions on the 10 patches.

Reducing Overfitting

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Journal of Machine Learning Research 15 (2014) 1929-1958



(a) Standard Neural Net

(b) After applying dropout.