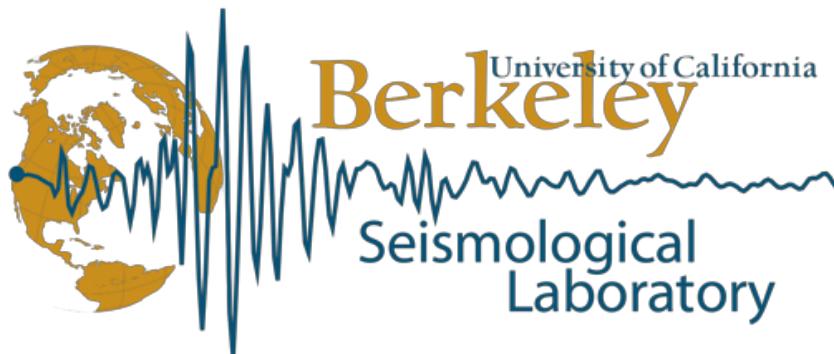




# Classification

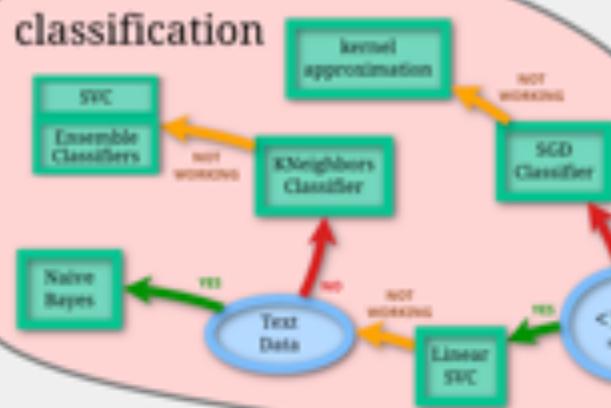
## Qingkai Kong



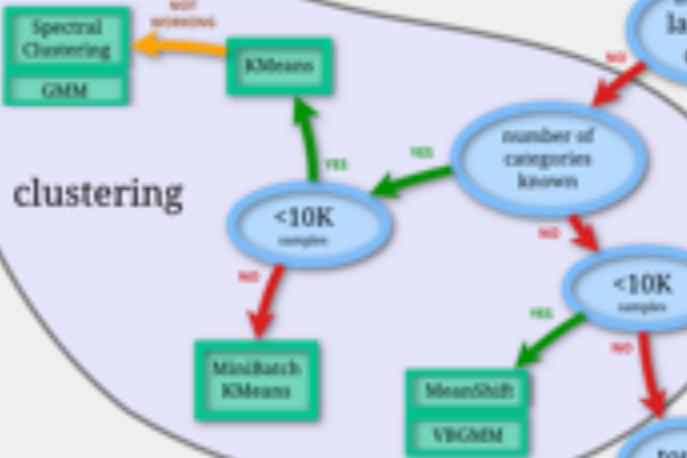
<http://seismo.berkeley.edu/qingkaikong/>

# scikit-learn algorithm cheat-sheet

## classification

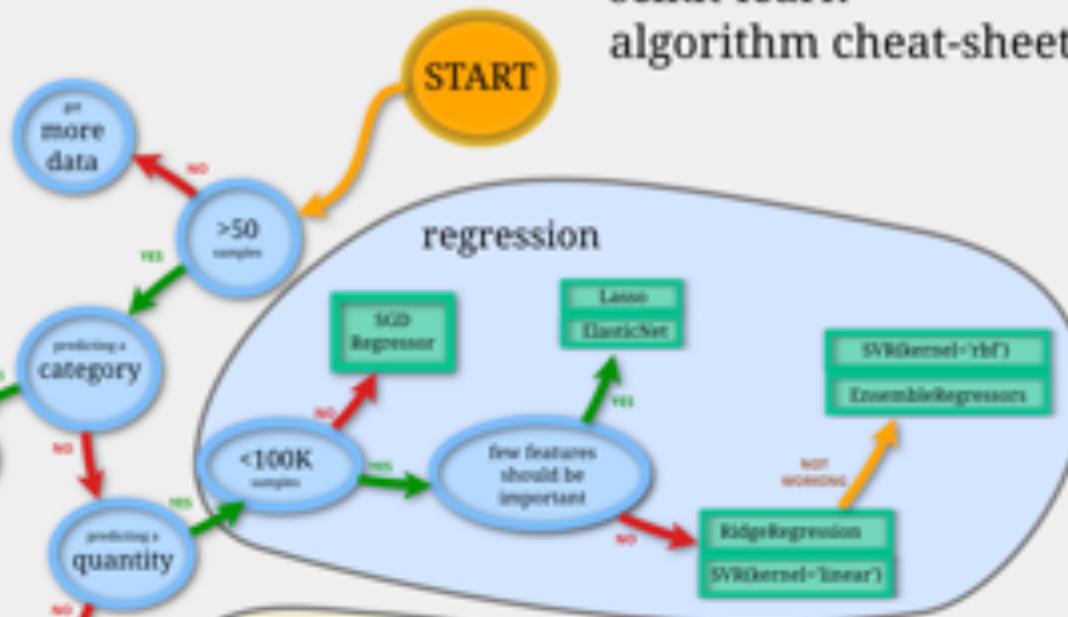


## clustering



Interactive version: [http://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/index.html](http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)

## regression



## dimensionality reduction



# Pipeline of training a machine learning model.

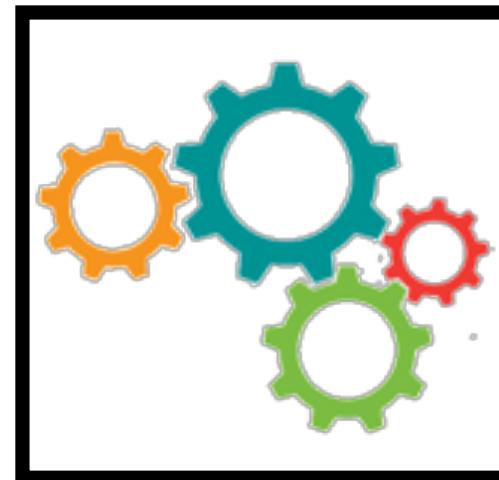
Data examples

01100  
10110  
11110

Optimization algorithm



Tunable Model



Trained Model

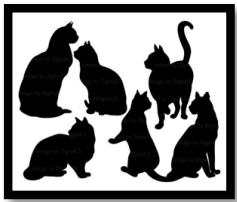


# Representation of data

## Raw data



Documents



Images



Numbers

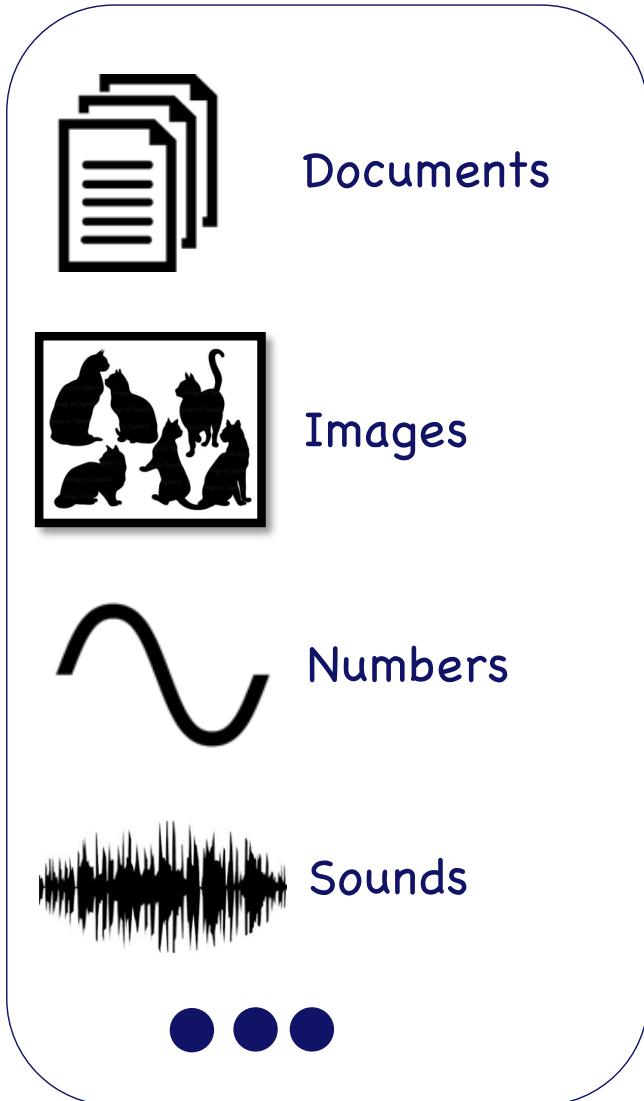


Sounds

• • •

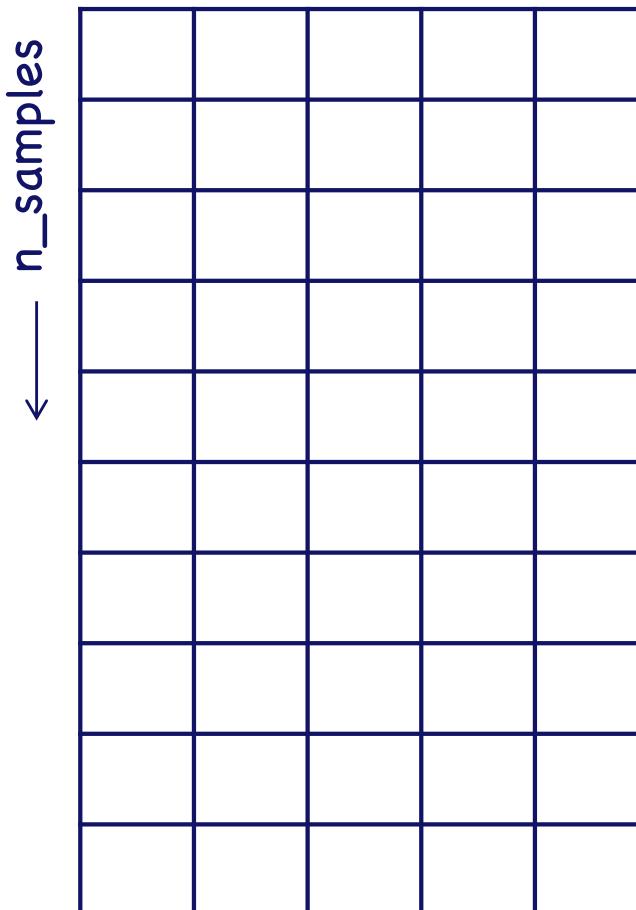
# Representation of data

# Raw data

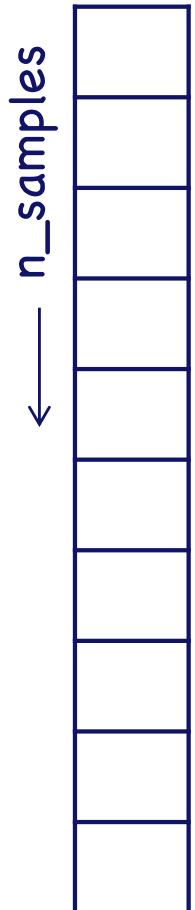


# Feature matrix ( $X$ )

`n_features` →



# Target (y)



# Example



# Example



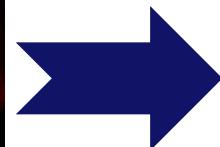
## Feature matrix ( $X$ )

n\_features →

# Target (y)

$n_{samples}$

# Example



n\_samples

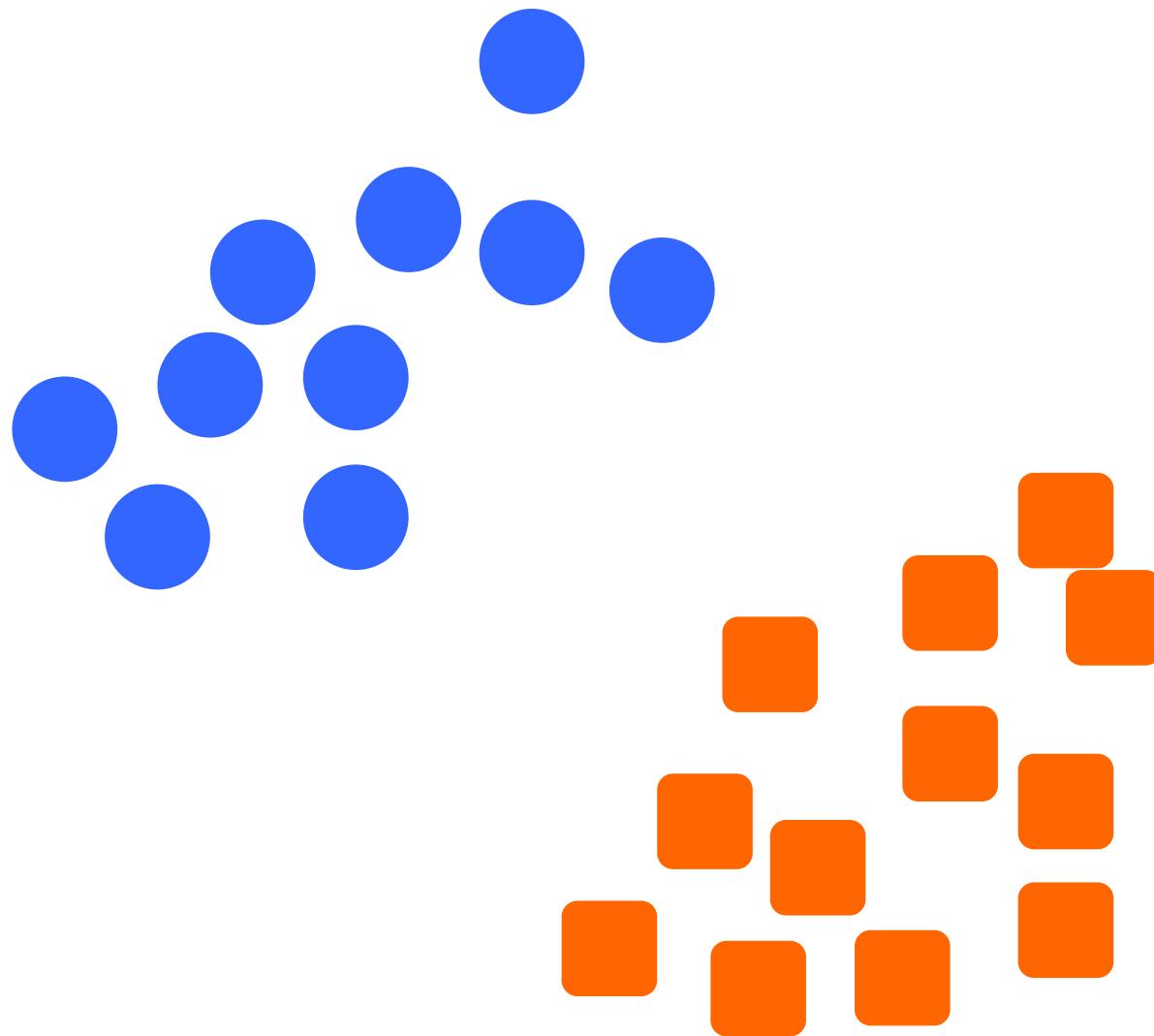
# Feature matrix ( $X$ )

`n_features` →

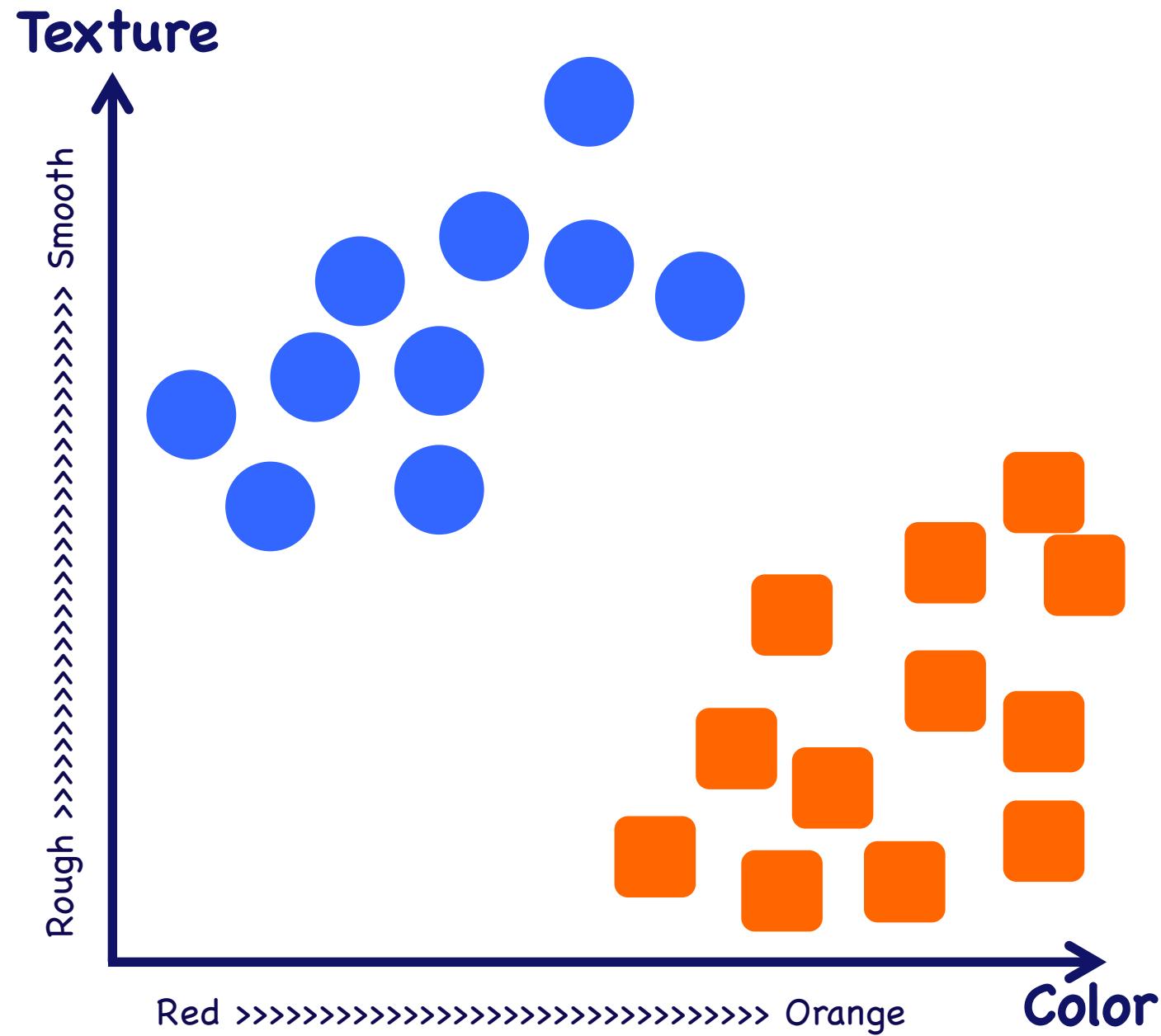
# Target (y)

0
1
1
0
1
1
1
0
0
1

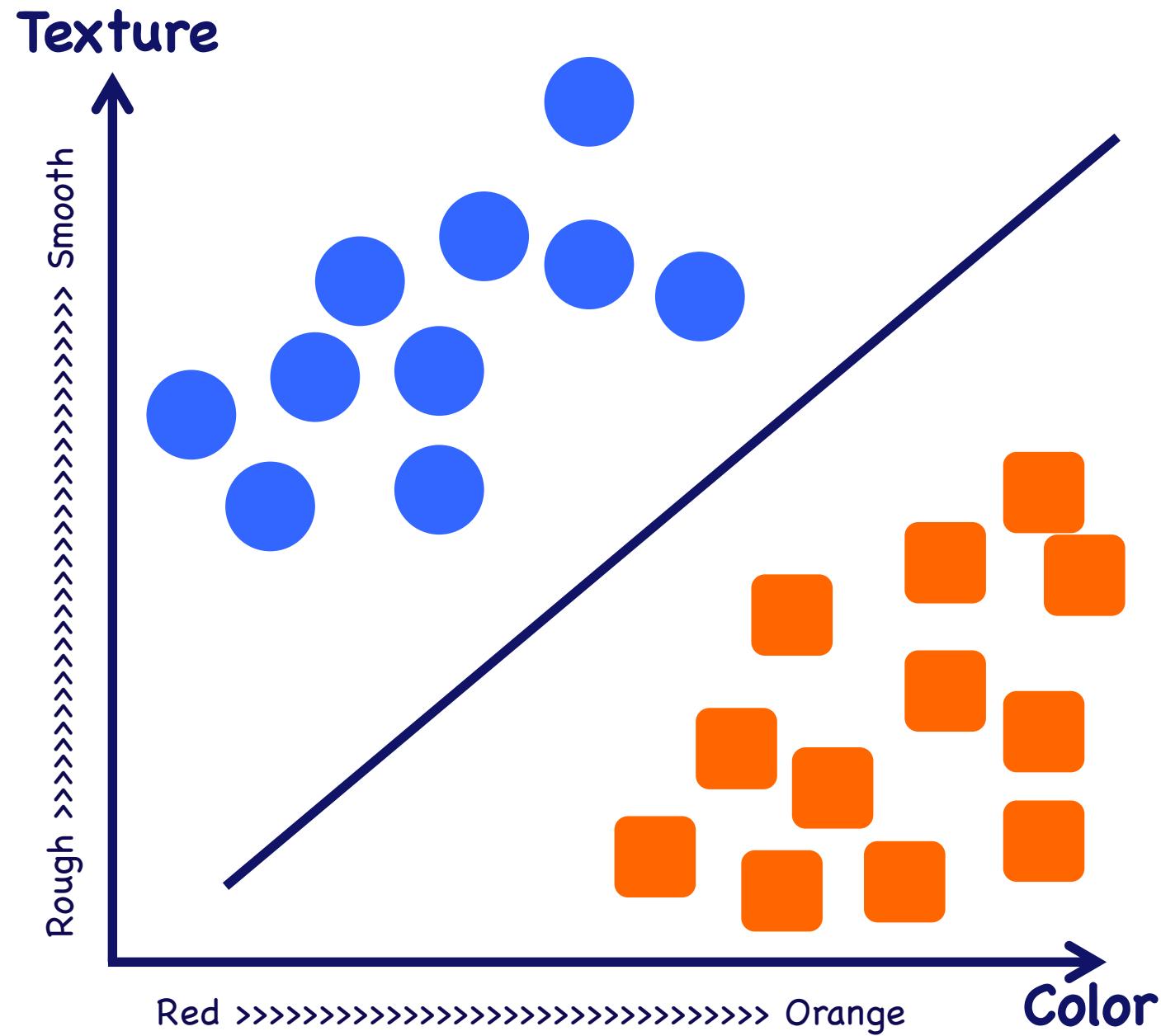
# Classification



# Classification



# Classification

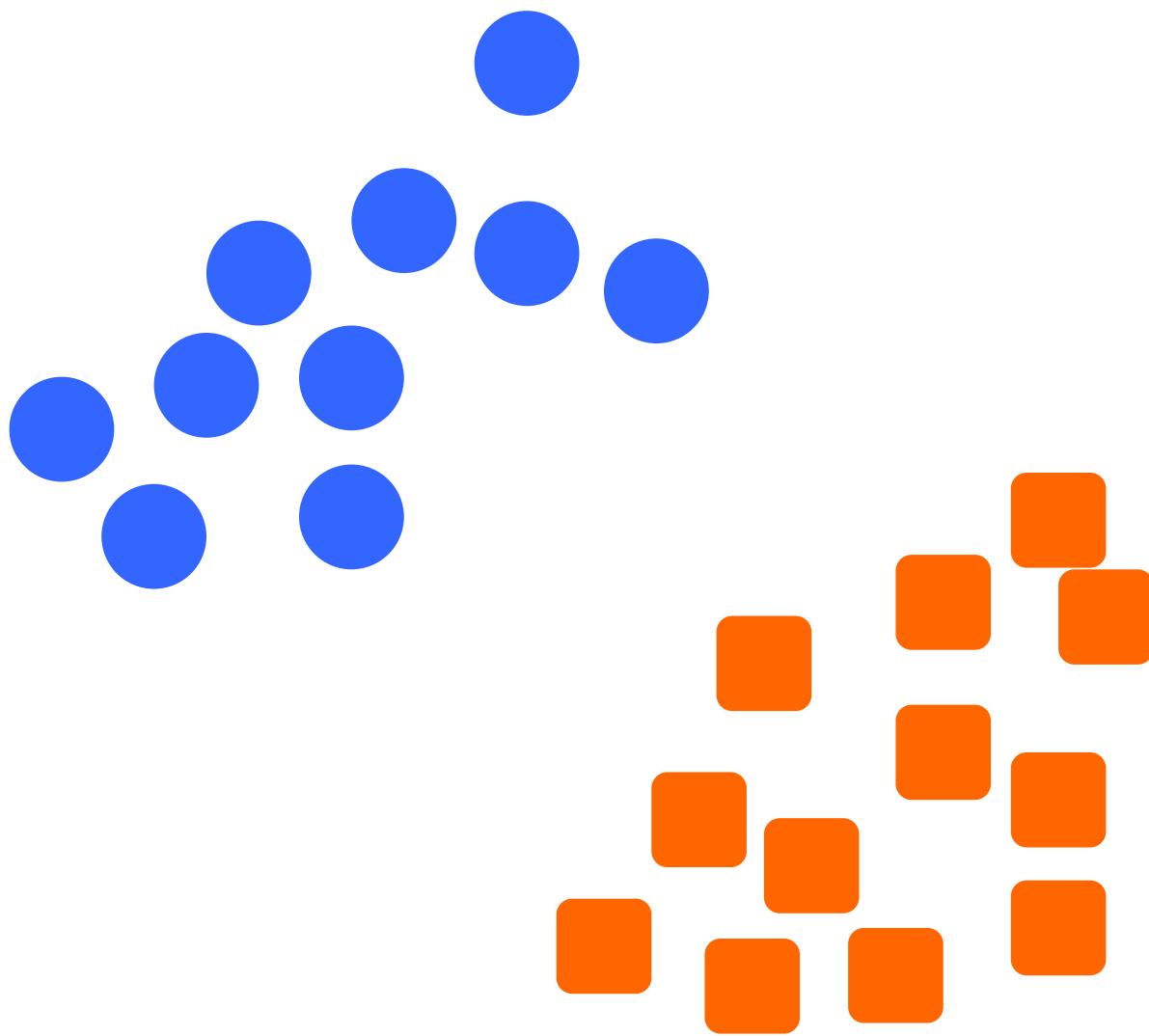


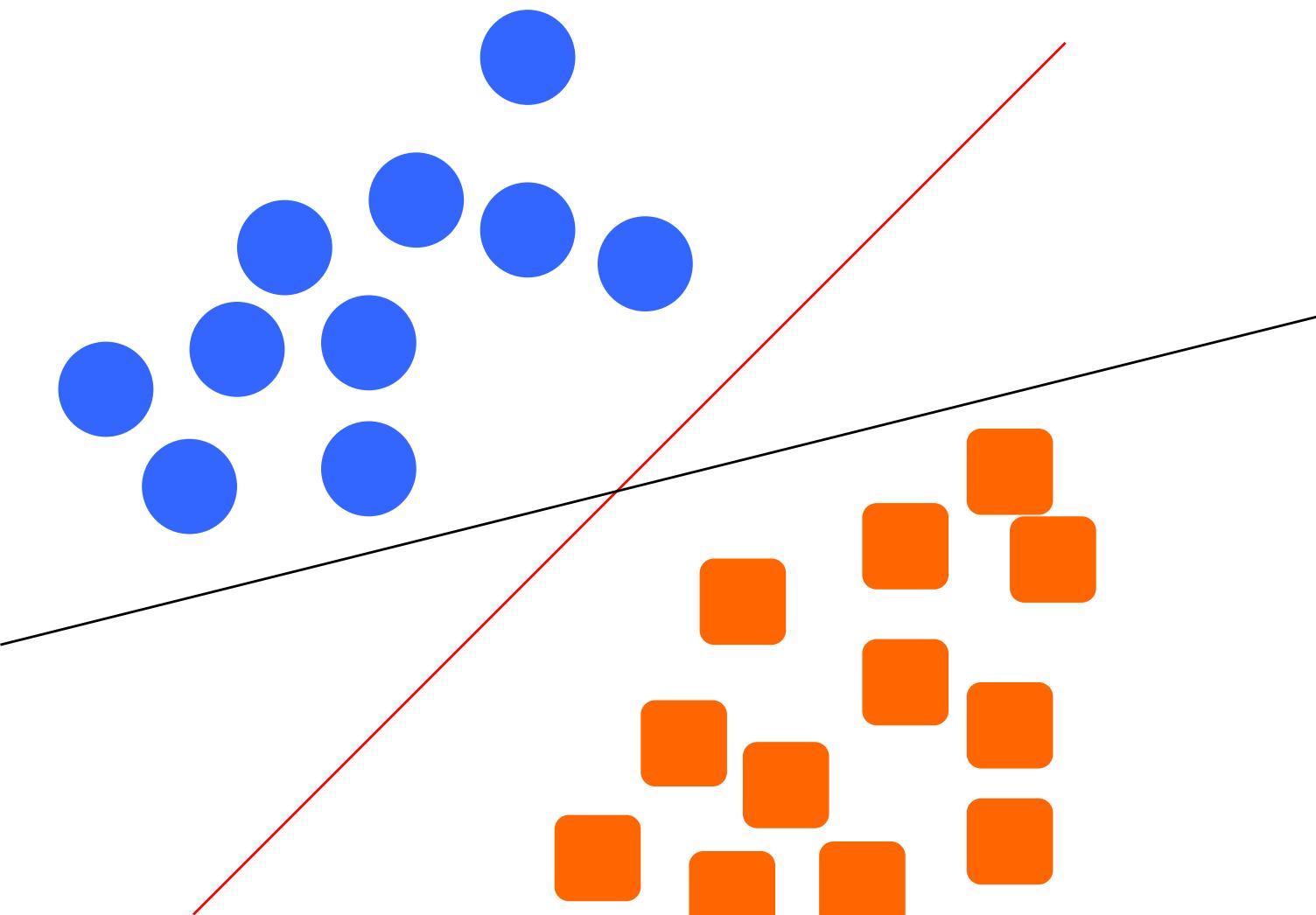
# Classification

Support Vector Machine  
Artificial Neural Network

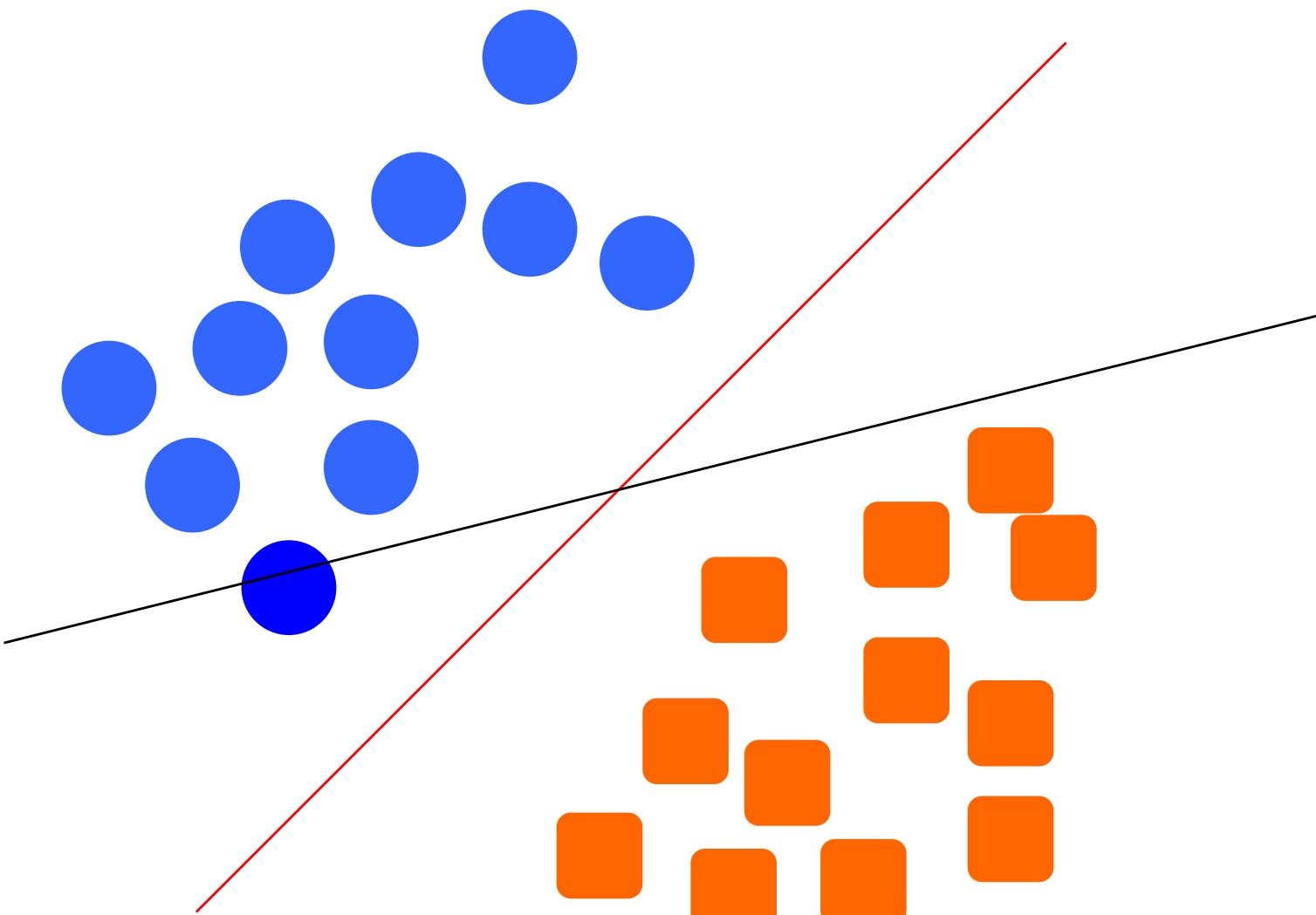


Supervised  
learning

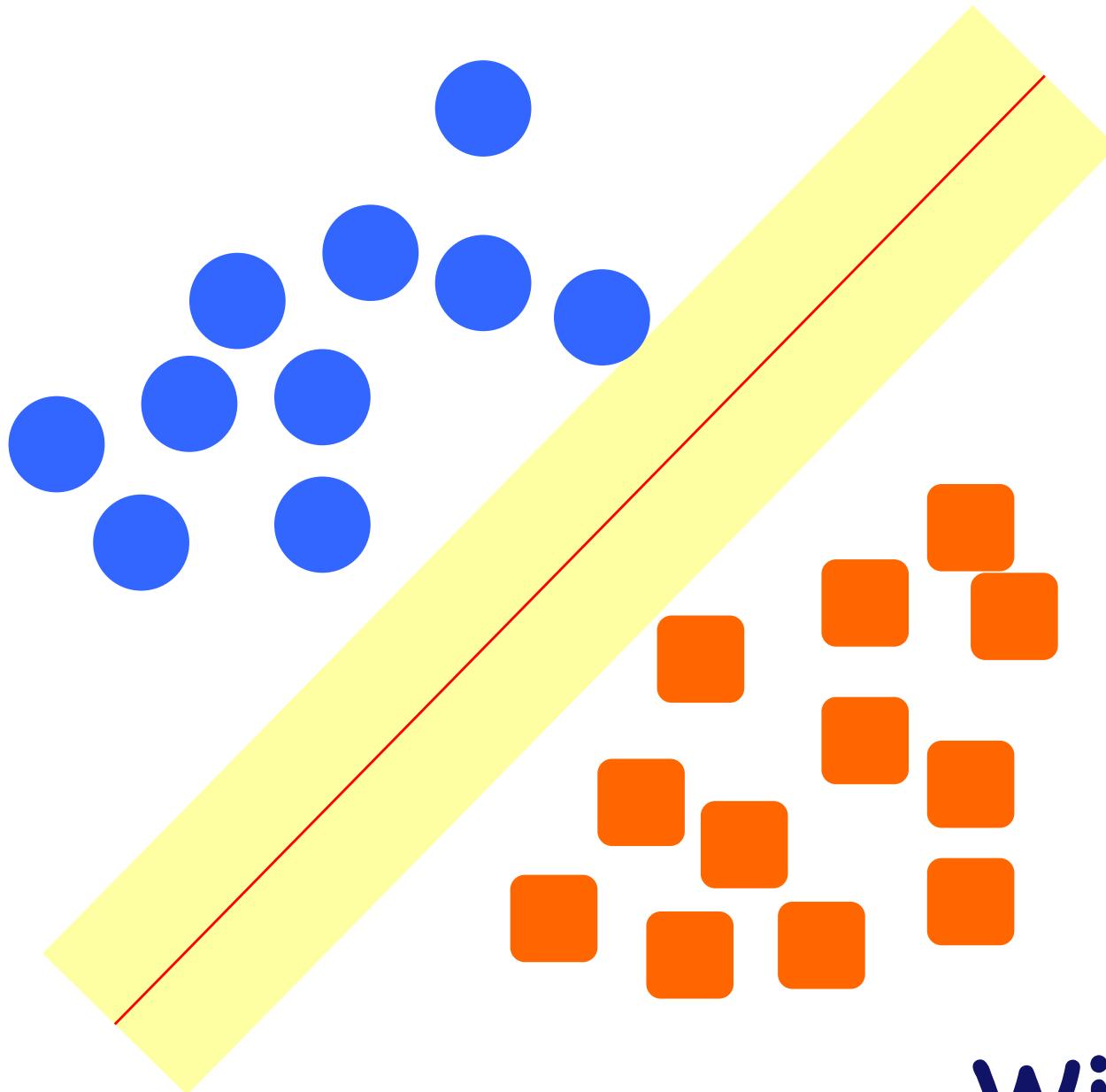




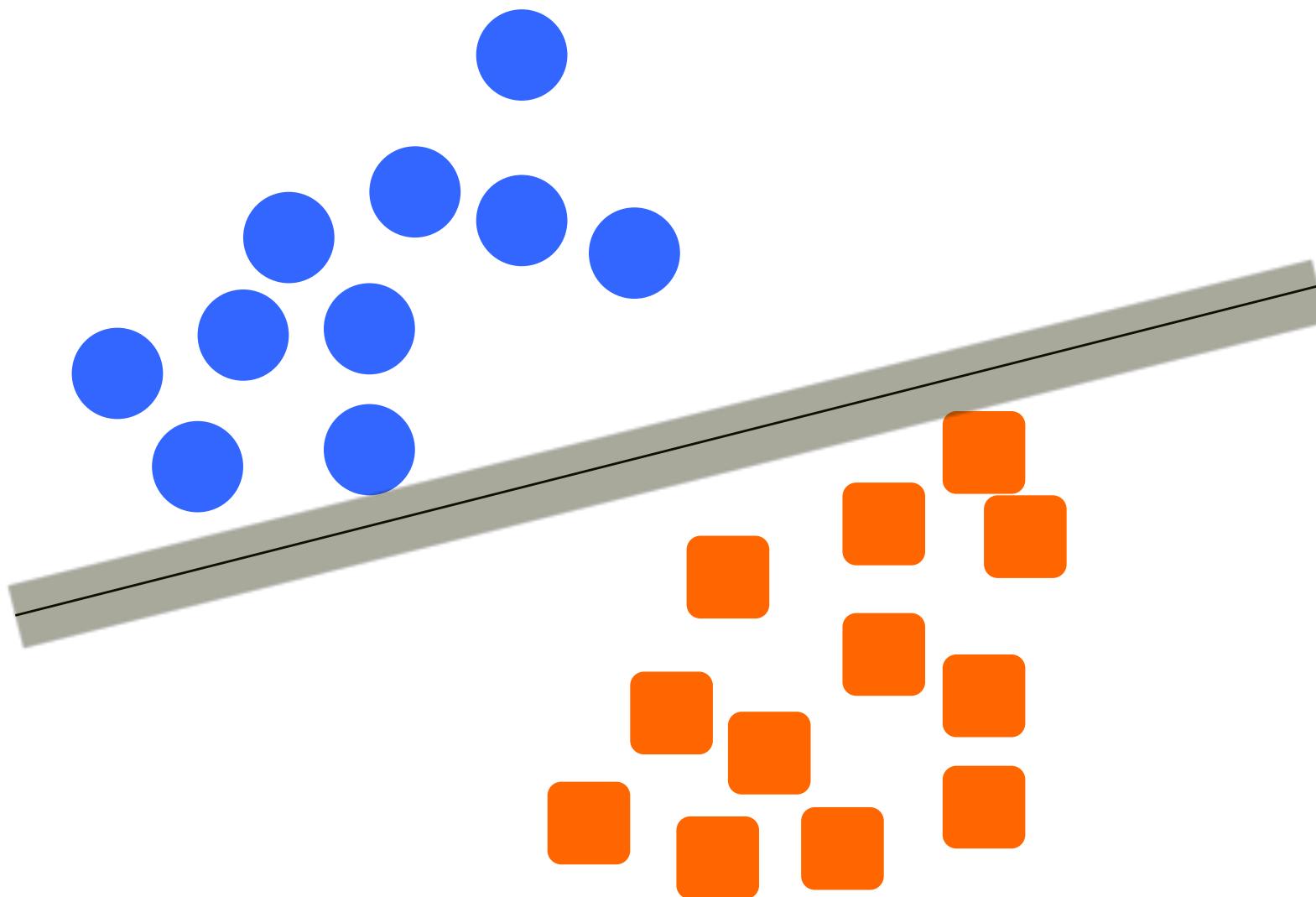
Which one is  
better?



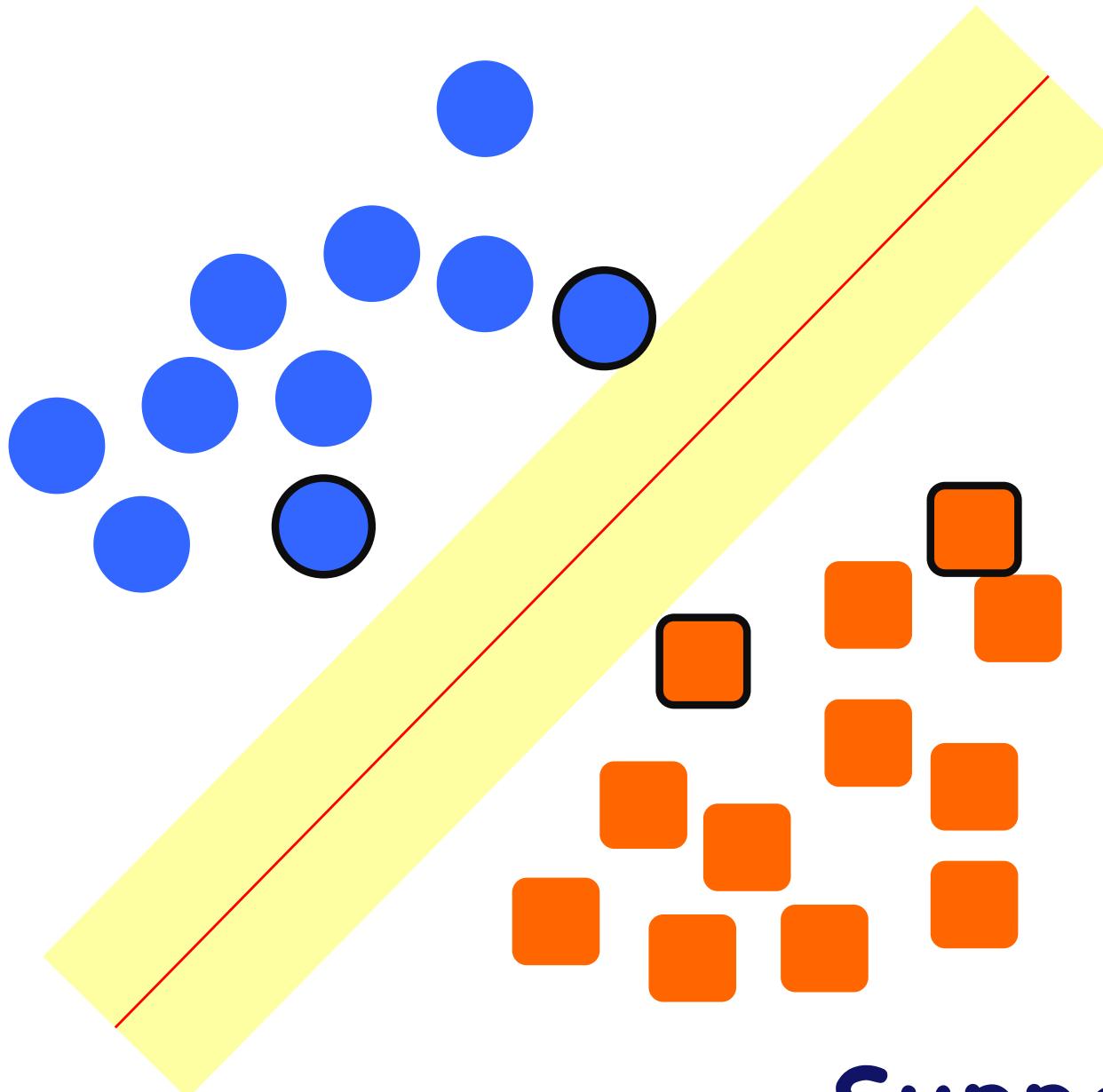
A new data point



Wide margin



Narrow margin



# Support Vectors

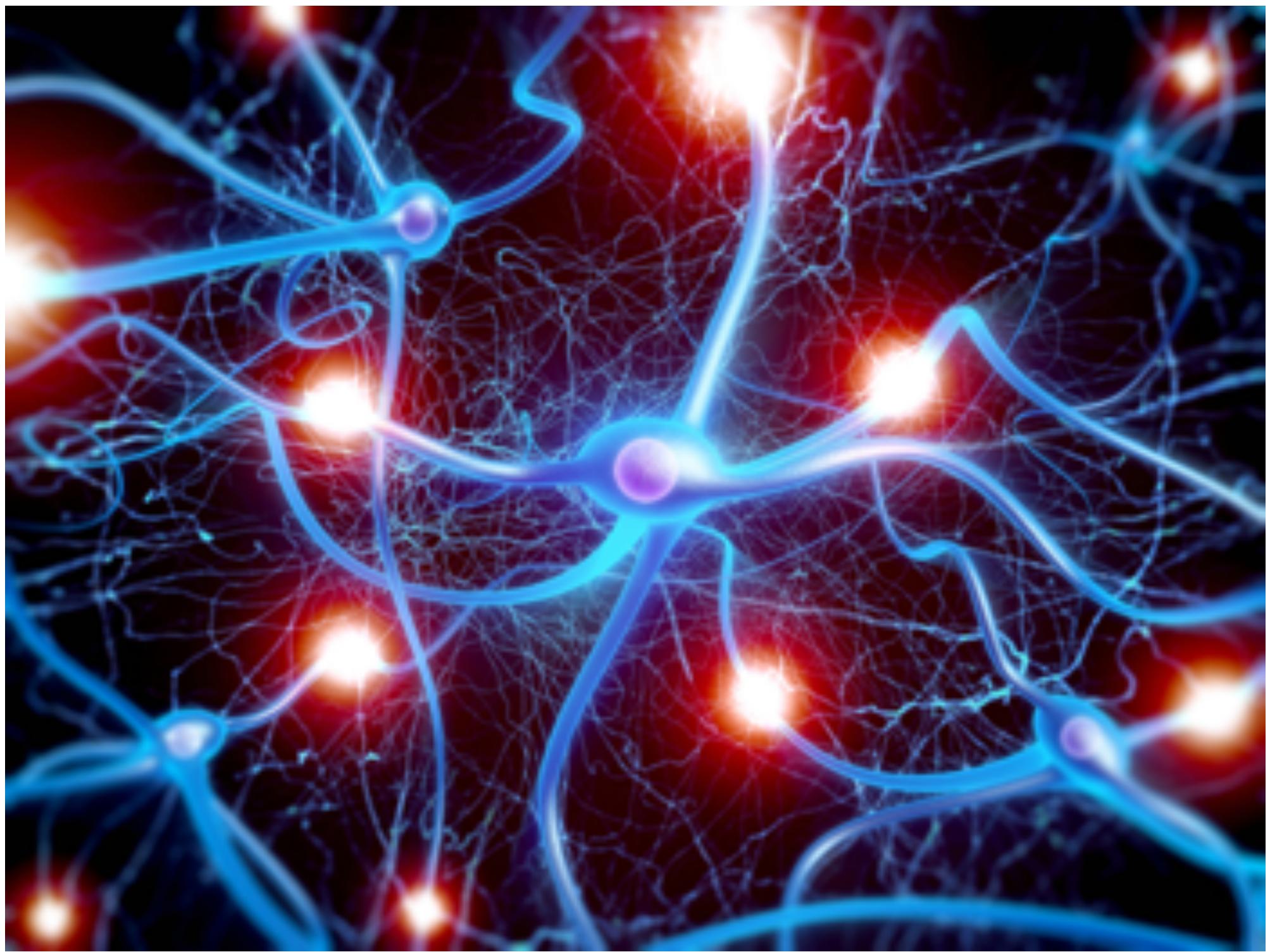
**It is maximize the margin!**

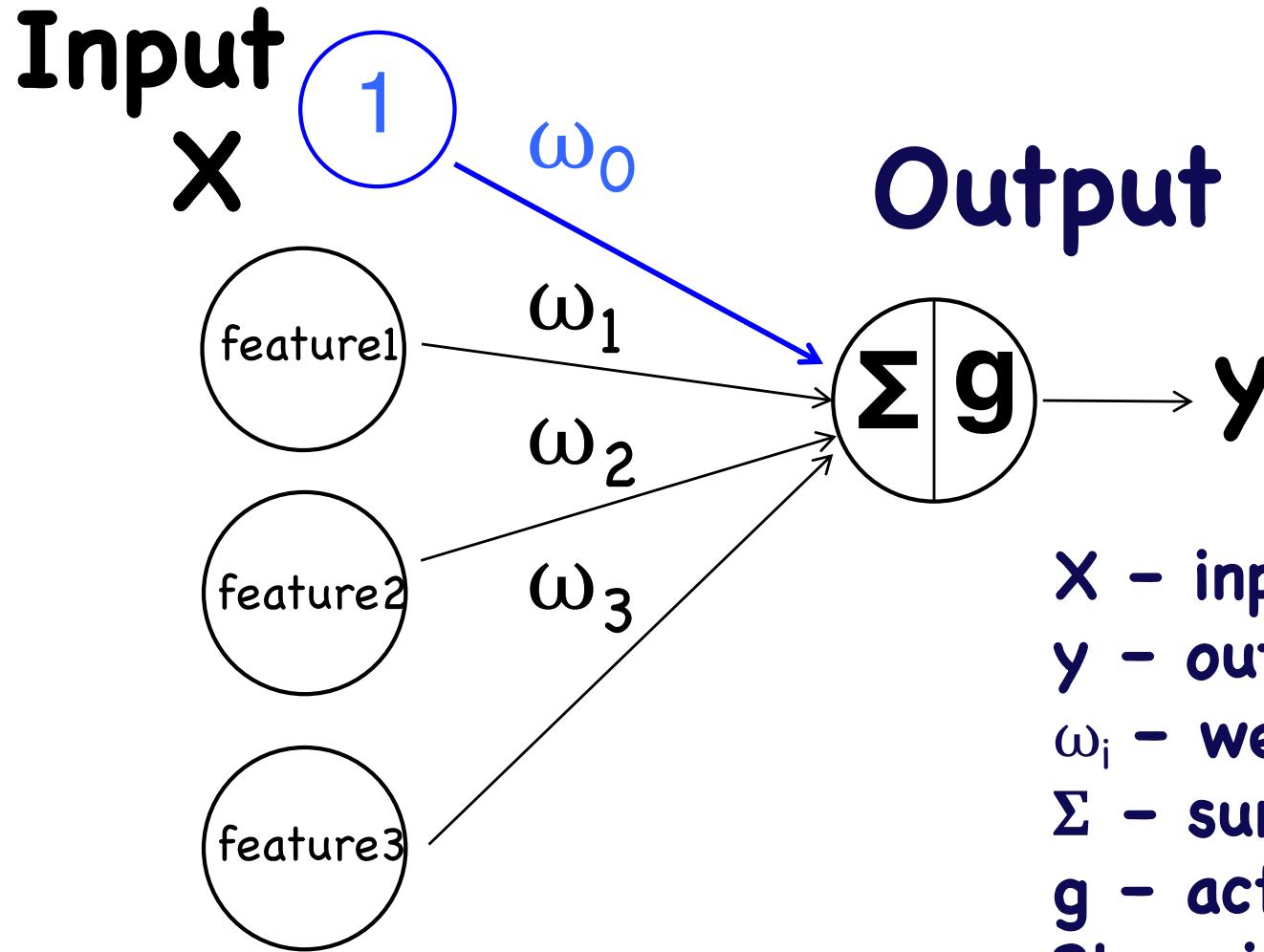
**Support Vectors**

More details: <https://www.youtube.com/watch?v=eHsErIPJWUU>

A black and white photograph of a puzzle piece. The puzzle piece is light gray with a dark gray outline. It contains the letters "ANN" in a bold, white, sans-serif font. The puzzle piece is set against a background of other light gray puzzle pieces, all with dark gray outlines. The overall image has a slightly grainy texture.

ANN



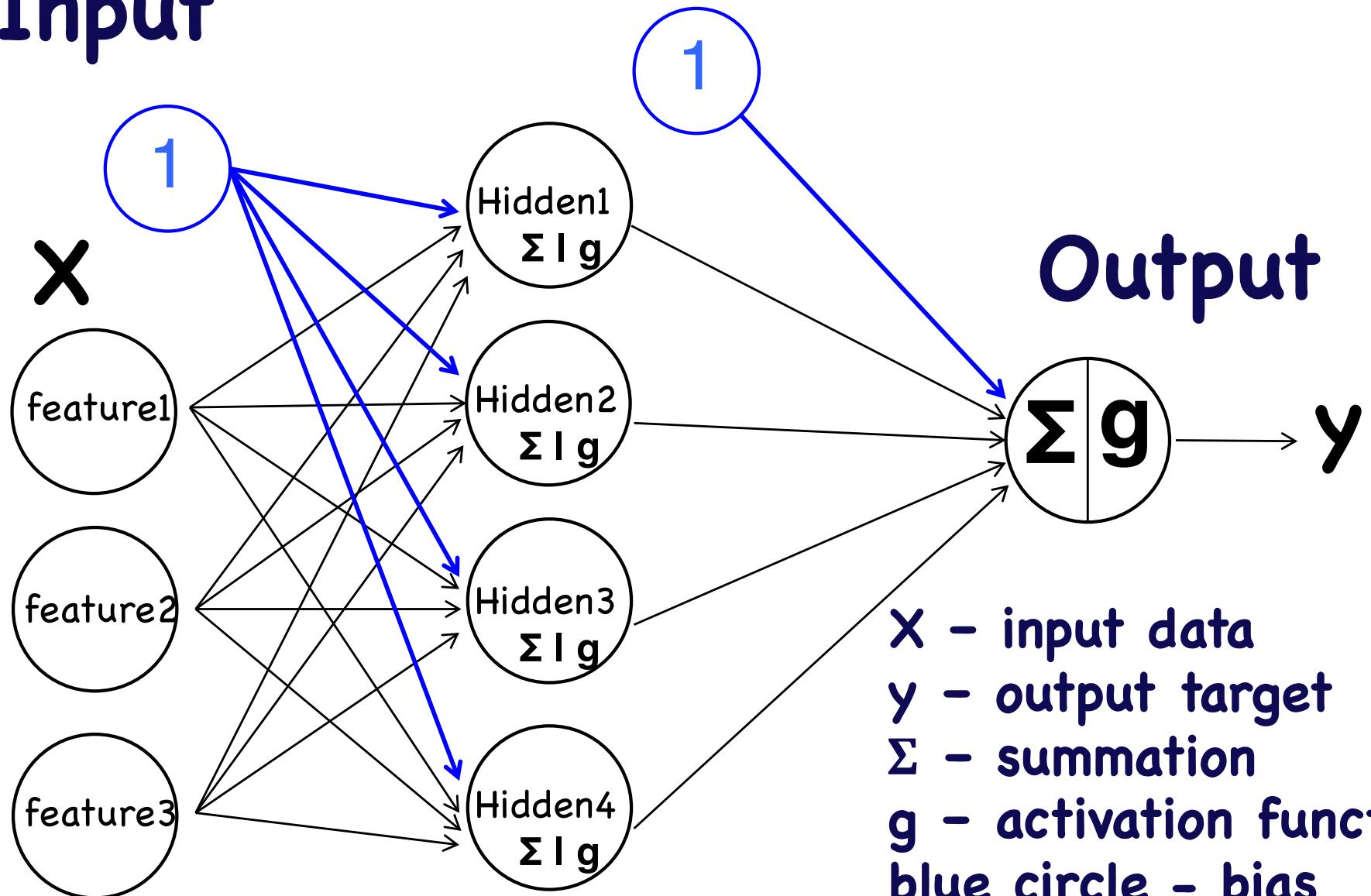


$x$  - input data  
 $y$  - output target  
 $\omega_i$  - weights  
 $\Sigma$  - summation  
 $g$  - activation function  
 Blue circle - bias

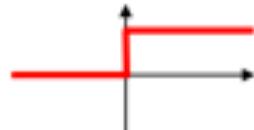
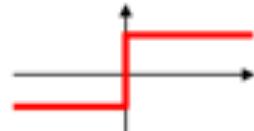
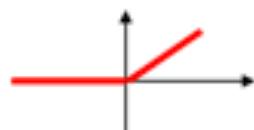
$$Z = \Sigma = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \dots + \omega_n x_n$$

$$y = g(\omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 + \dots + \omega_n x_n)$$

# Input



**x** - input data  
**y** - output target  
 $\Sigma$  - summation  
g - activation function  
blue circle - bias

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	



**YOU'RE IN MY SPOT**

GRAPHICS GARAGE

# Input



.

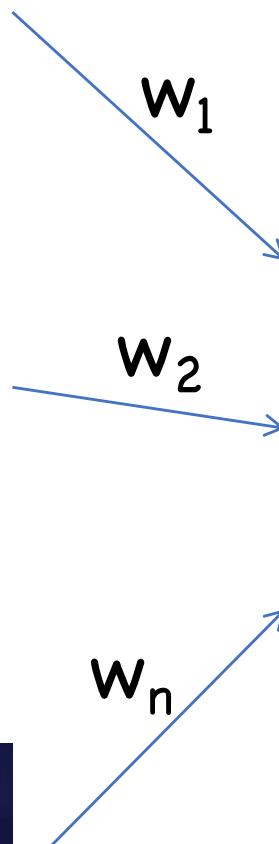
.

.

.

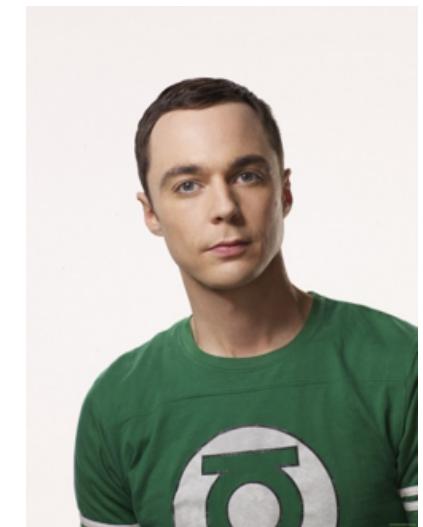


# Intuitive Artificial Neural Network



$$F(\text{eye} \times w_1 + \text{nose} \times w_2 + \dots + \text{mouth} \times w_n)$$

# Output



# Input

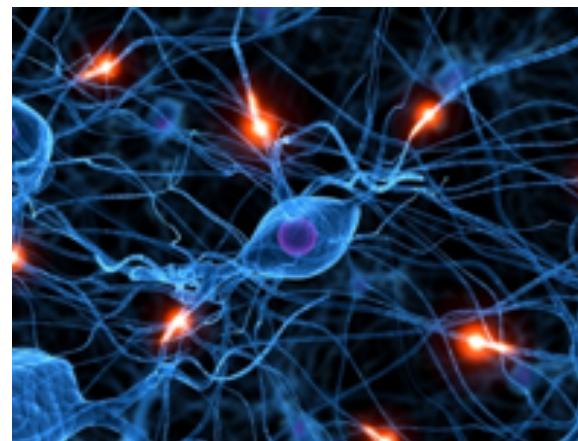


# Intuitive Artificial Neural Network

# Output



$$F(\text{eye} \times w_1 + \text{nose} \times w_2 + \dots + \text{mouth} \times w_n)$$



error  
feedback



# Input



.

.

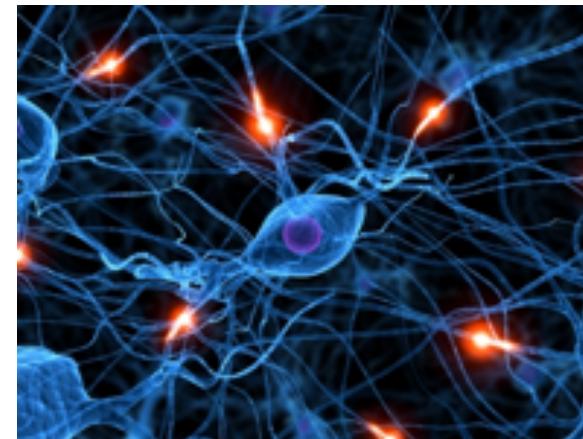
.



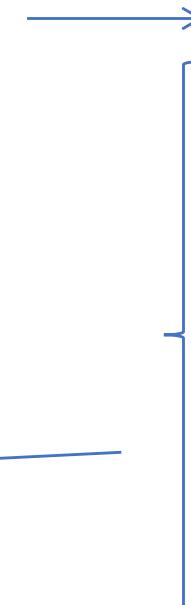
# Intuitive Artificial Neural Network

# Output

$$F(\text{eye} \times w_1 + \text{nose} \times w_2 + \dots + \text{mouth} \times w_n)$$



error  
feedback



more photos

# Input



.

.

.

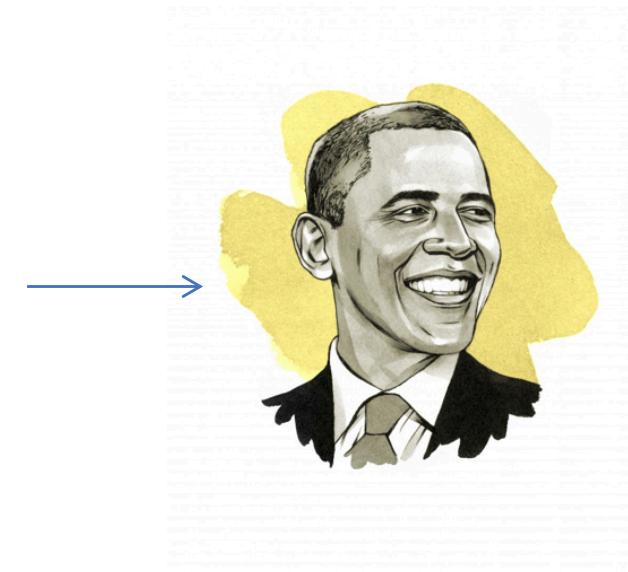
.



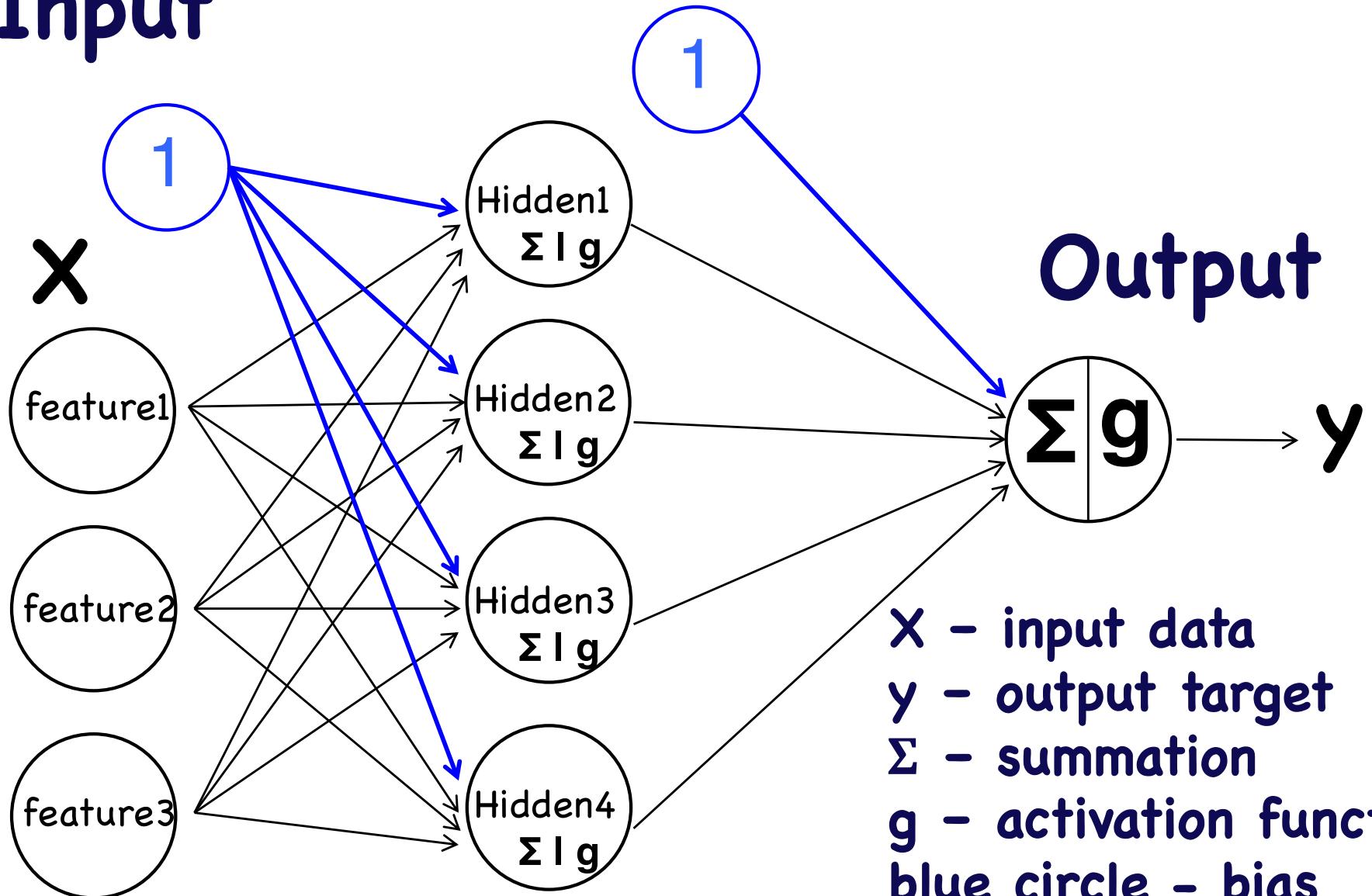
# Intuitive Artificial Neural Network

# Output

$$F(\text{eye} \times w_1 + \text{nose} \times w_2 + \dots + \text{mouth} \times w_n)$$



# Input



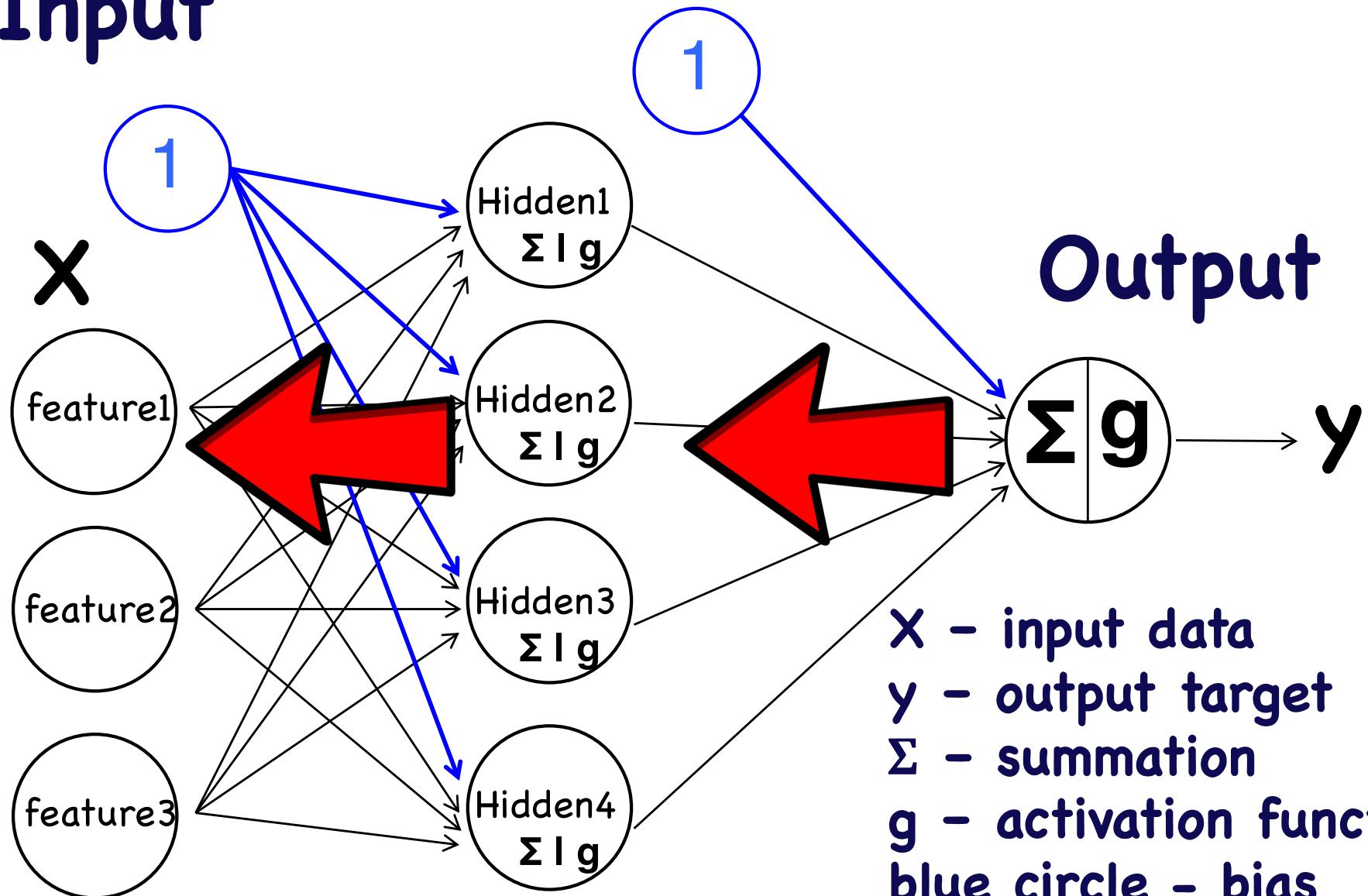
**x** - input data  
**y** - output target  
 $\Sigma$  - summation  
g - activation function  
blue circle - bias

~~Error~~



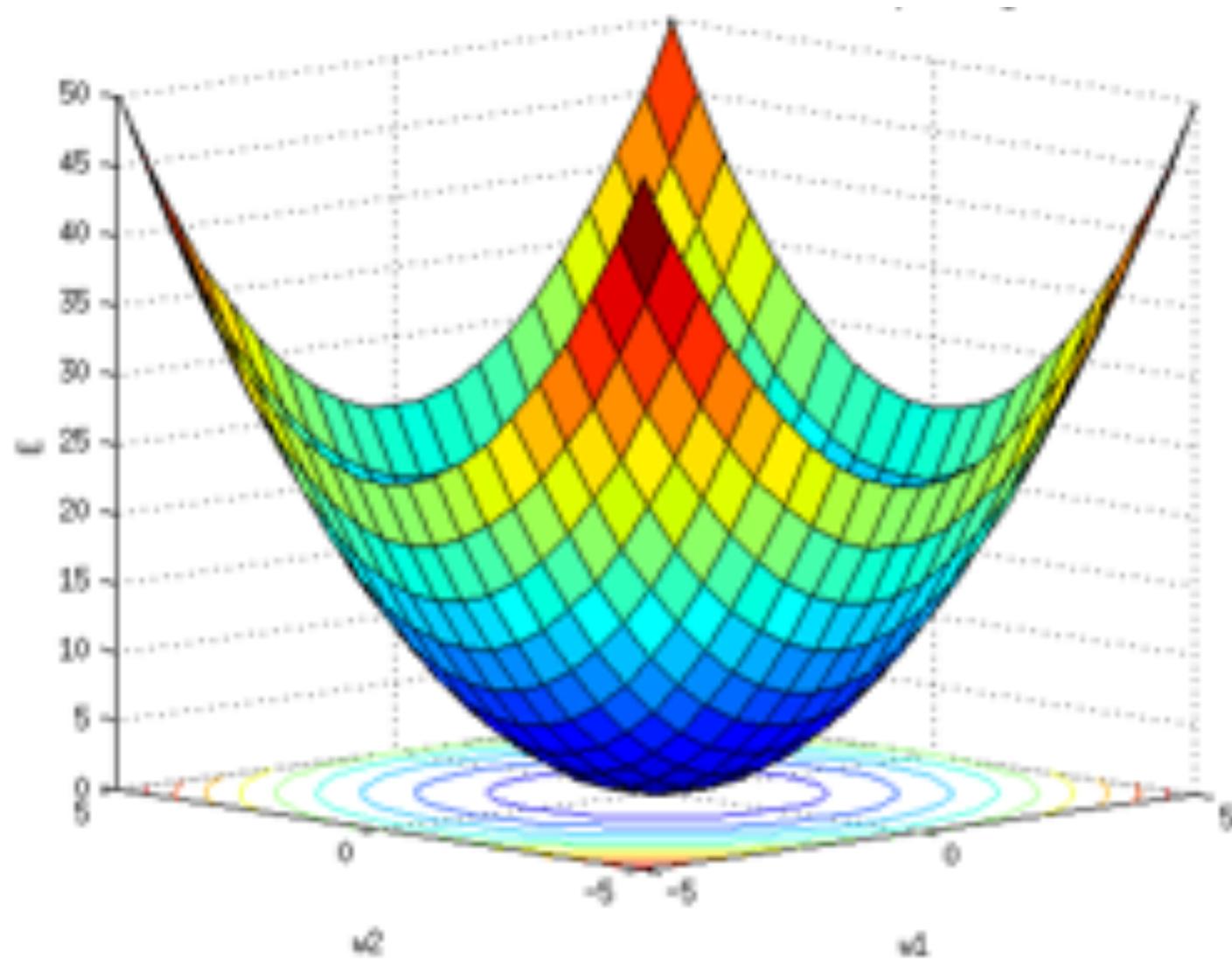
# Backpropagation

Input



More details: [https://www.youtube.com/watch?v=x\\_Eamf8MHwU](https://www.youtube.com/watch?v=x_Eamf8MHwU)

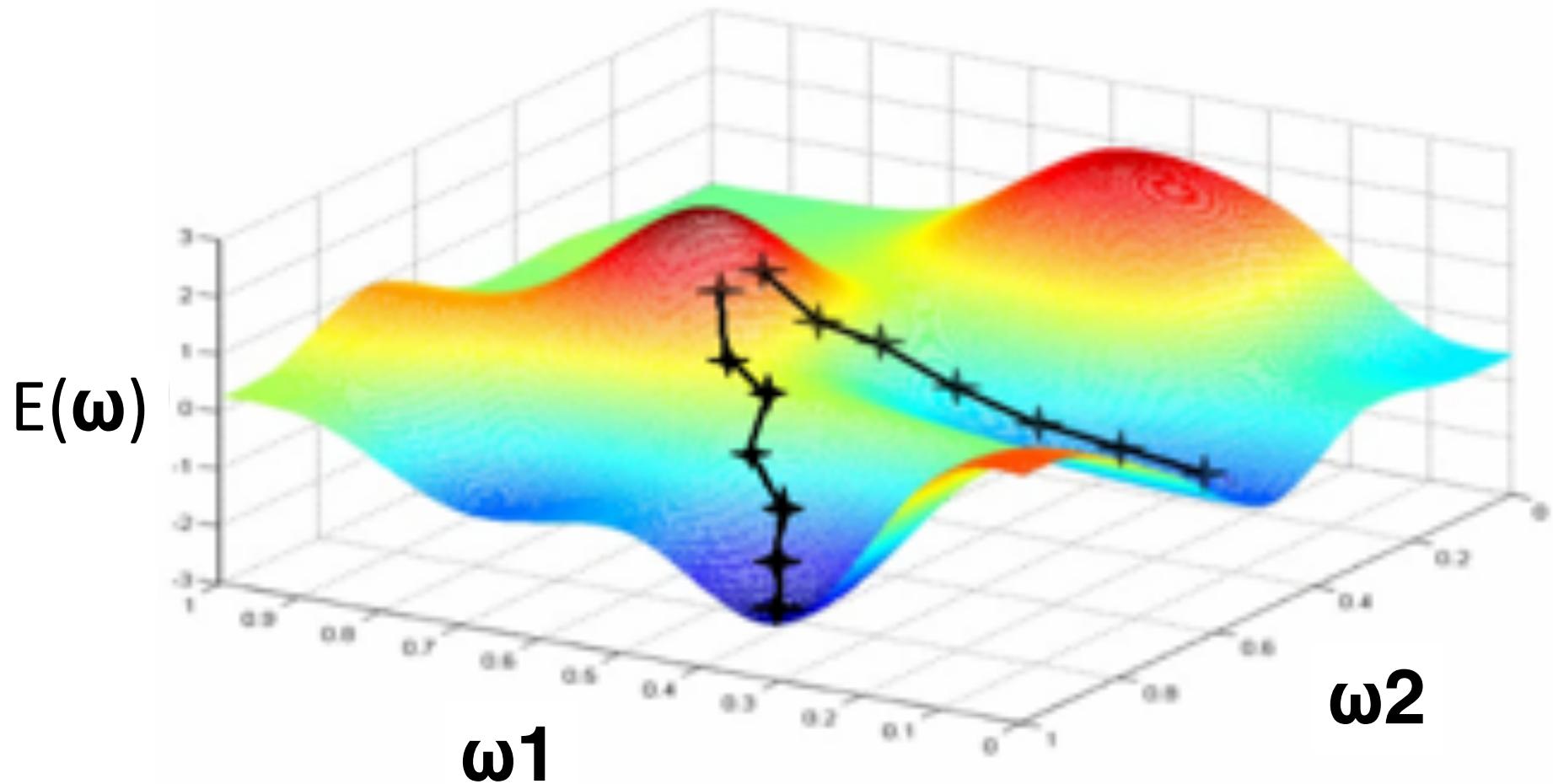
# Ideal Cost Function



# Real-world Cost Function



# Gradient descend



More details: <https://www.youtube.com/watch?v=TveJaJs5Fqk>

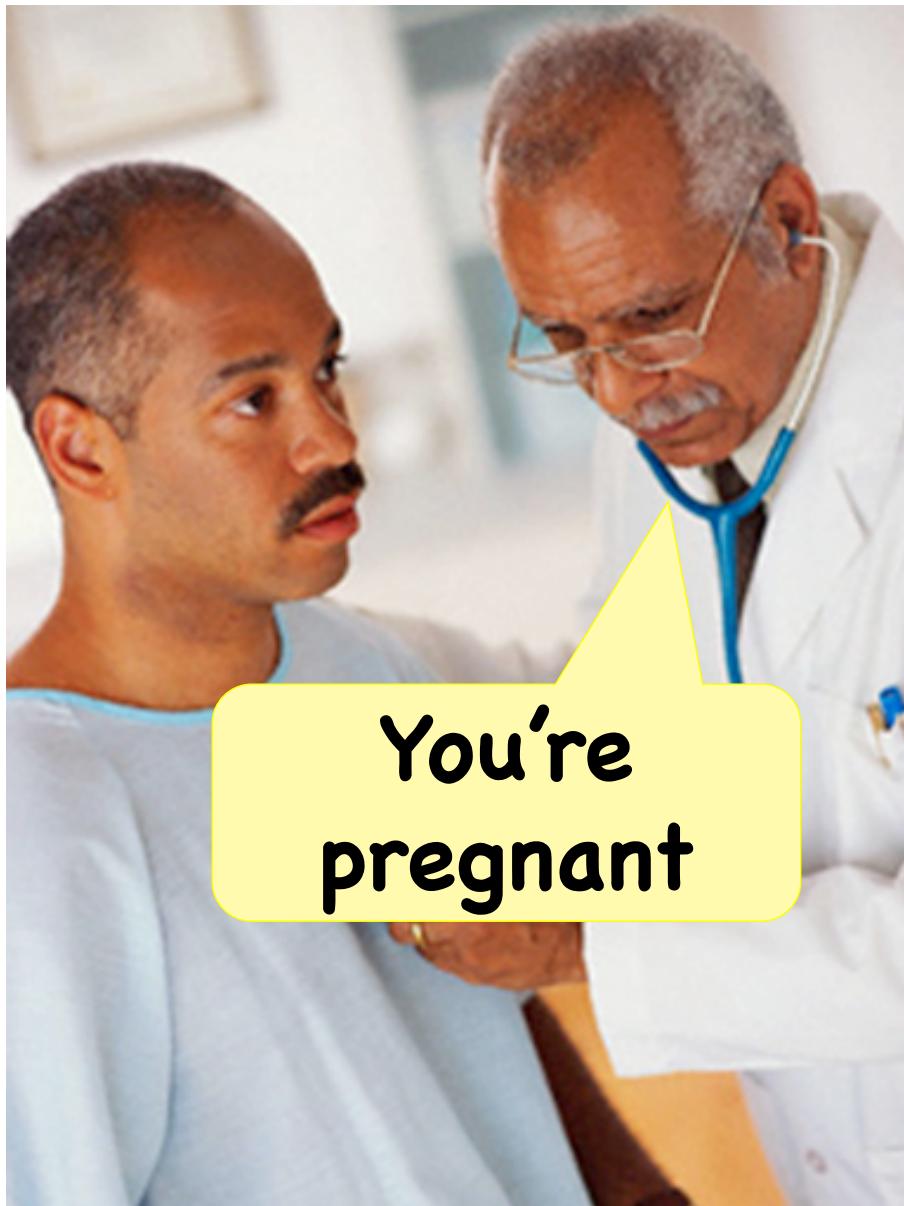
# Iterate many times



# Evaluate performance

- Confusion matrix
- Precision & Recall
- ROC curve (Receiver operating characteristic)

## False Positive



## False Negative



# Confusion matrix

## Predicted class

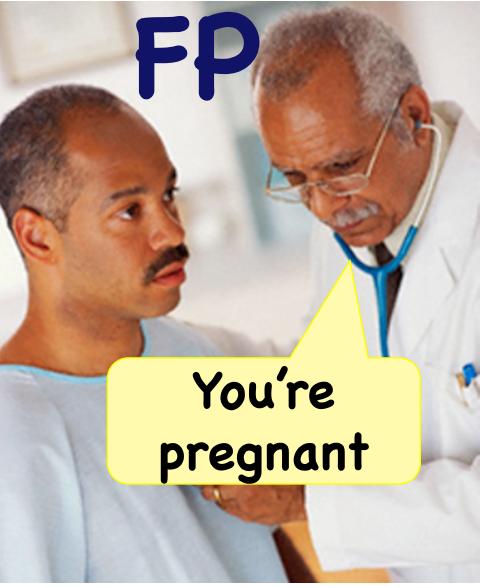
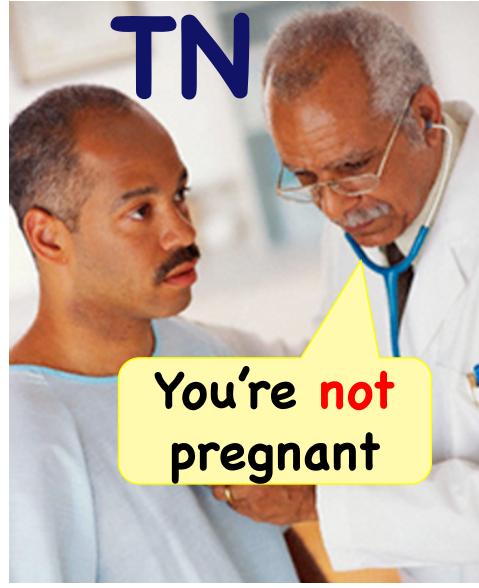
P

N

True class

P

N

		P	N
P	TP		
	FP		

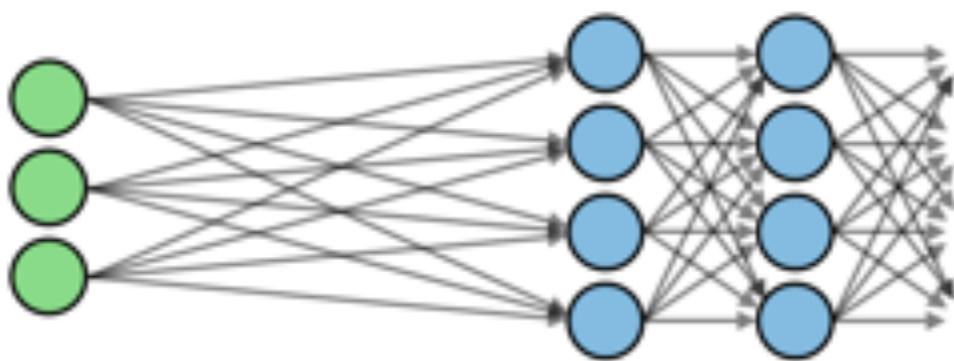
# Some useful resources

- <https://playground.tensorflow.org>
- <https://seat.massey.ac.nz/personal/s.r.marsland/MLBook.html>
- <http://neuralnetworksanddeeplearning.com/>
- [Python machine learning](#) - Sebastian Raschka
- [Deep learning with Python](#) - Francois Chollet
- [Machine Learning - An Algorithmic Perspective](#) - Stephen Marsland
- Datasets put together by Men-Andrin and Zach Ross
  - <http://scedc.caltech.edu/research-tools/deeplearning.html>

Many pictures in this talk are from internet, I thank all the authors here!

Go to the notebook

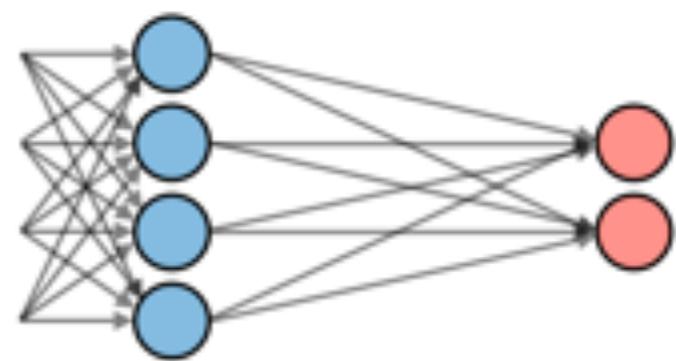
# Deep neural network



Input layer

Hidden layer 1

...



...

Hidden layer  $k$

Output layer

# Unreasonable Benefits of Deep Learning

simplicity

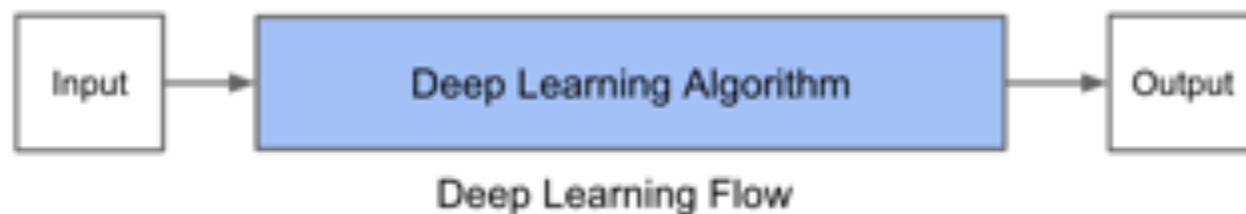
accuracy

flexibility

hacks

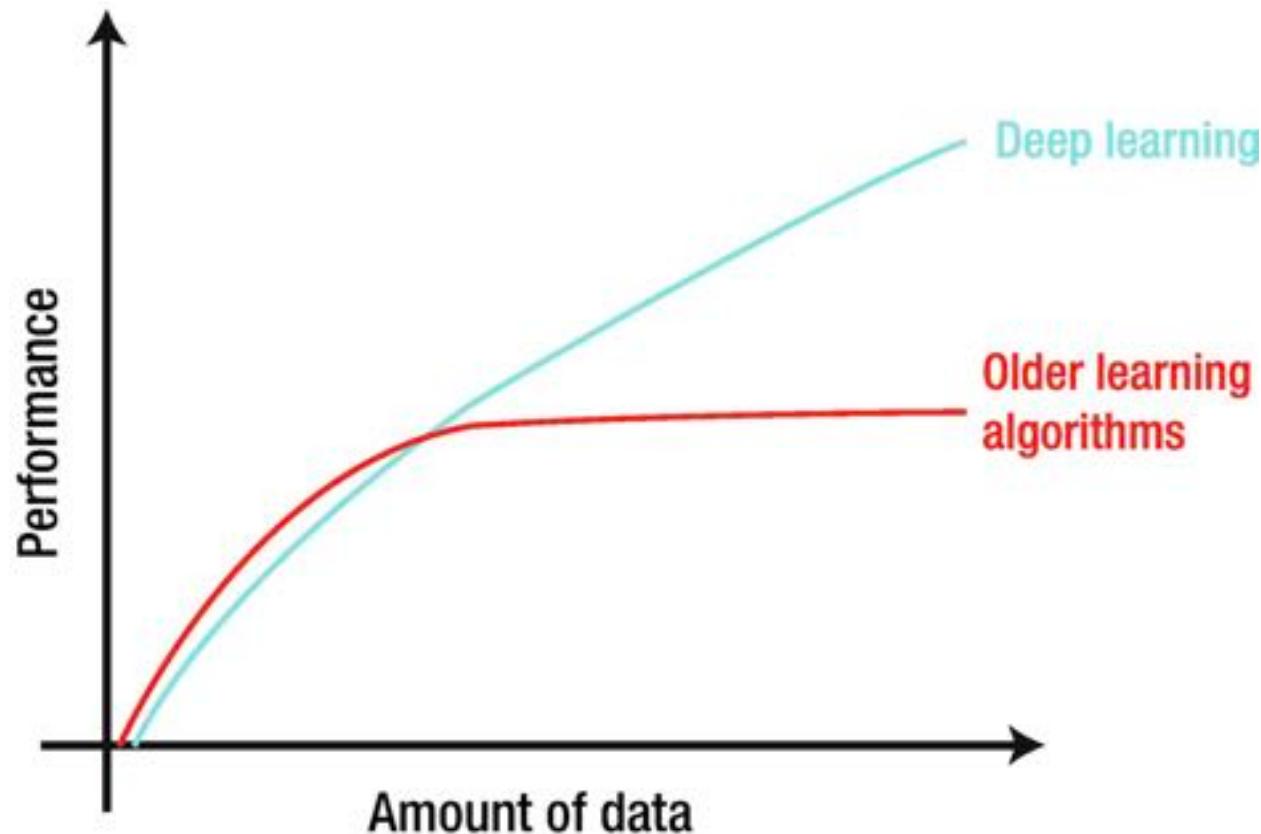
# Benefits of deep learning

## Simplicity



# Benefits of deep learning

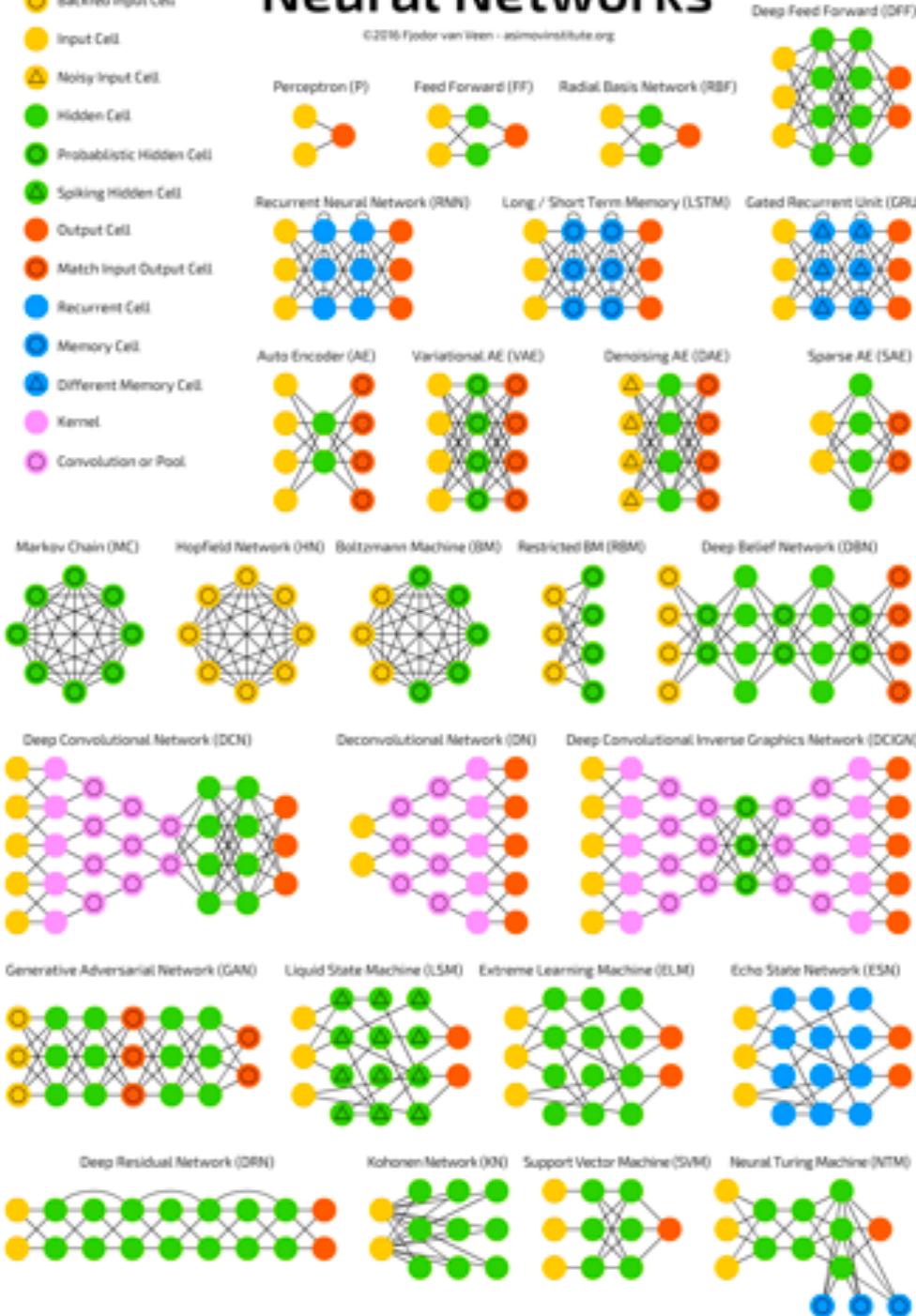
## Accuracy



- Backfield Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

## A mostly complete chart of Neural Networks

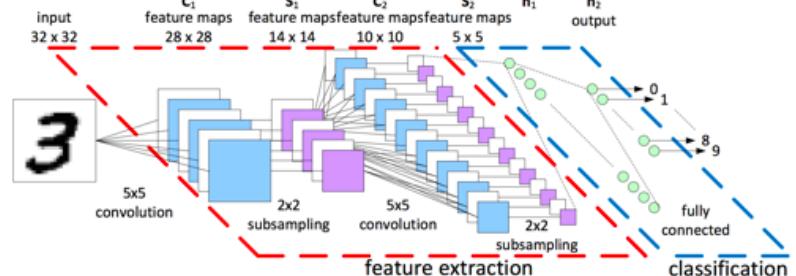
©2016 Fjodor van IJken - animovinstitute.org



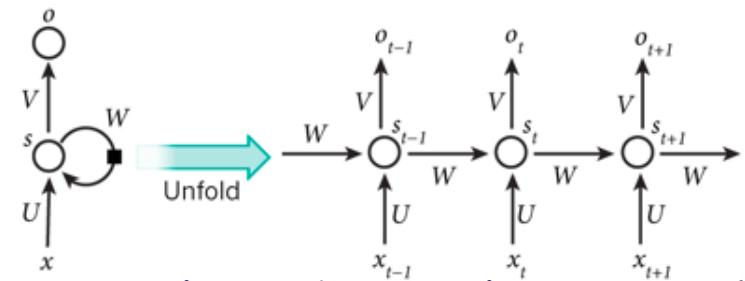
# Benefits

## Flexible

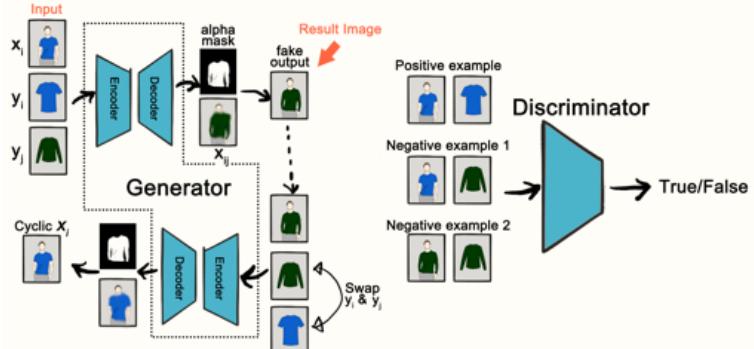
### Convolutional Neural Network



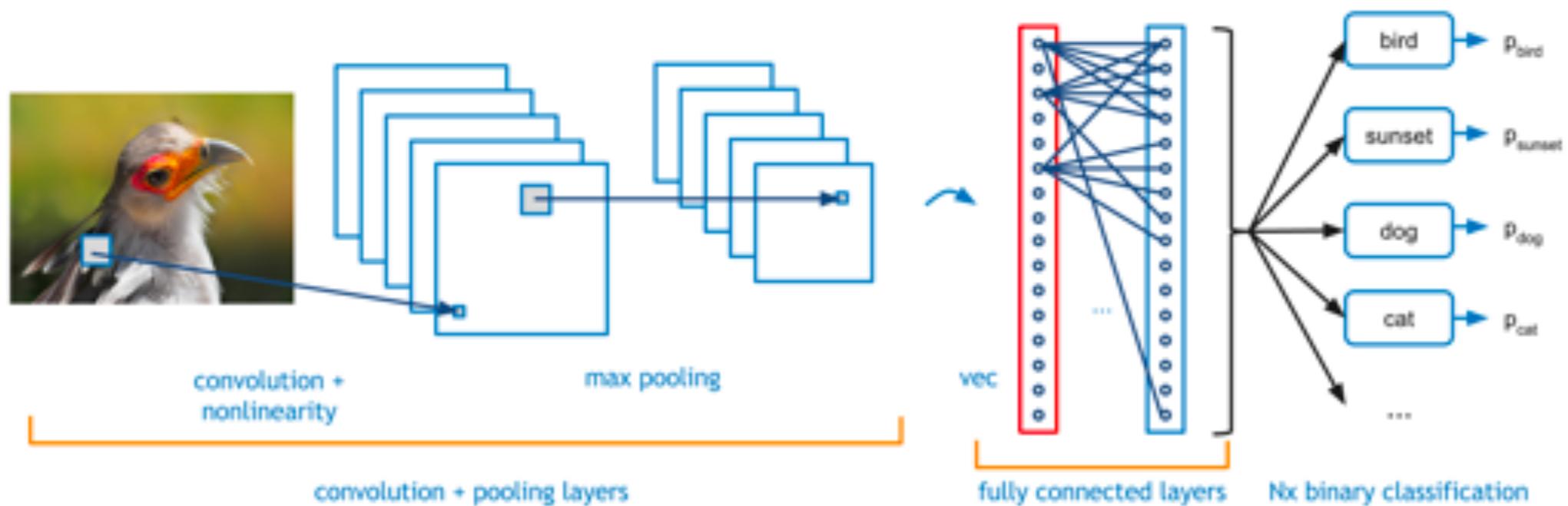
### Recurrent Neural Network



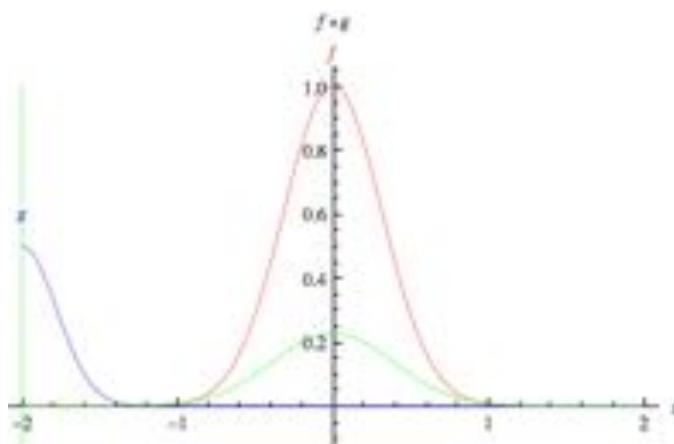
### Generative Adversarial Network



# Convolutional neural network



# Convolution operation



<http://mathworld.wolfram.com/>

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

# Max pooling

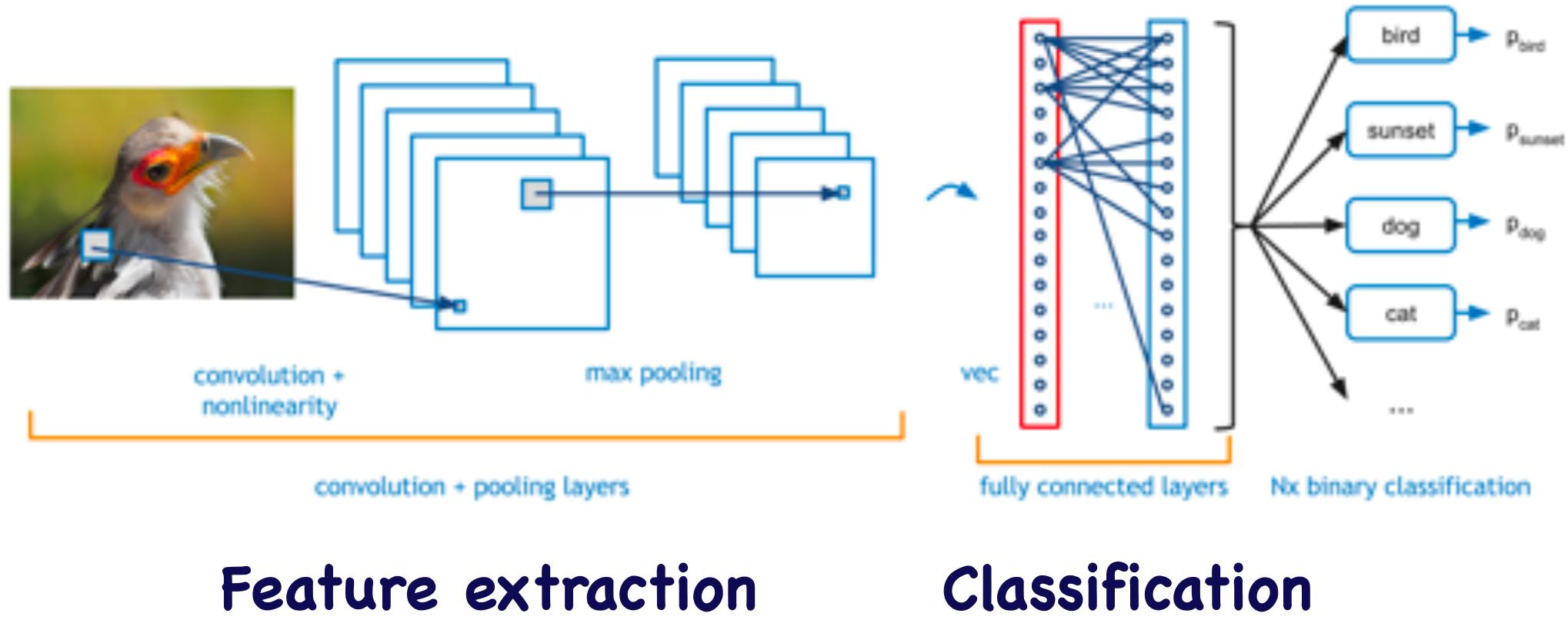
1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

max pool with 2x2  
window and stride 2

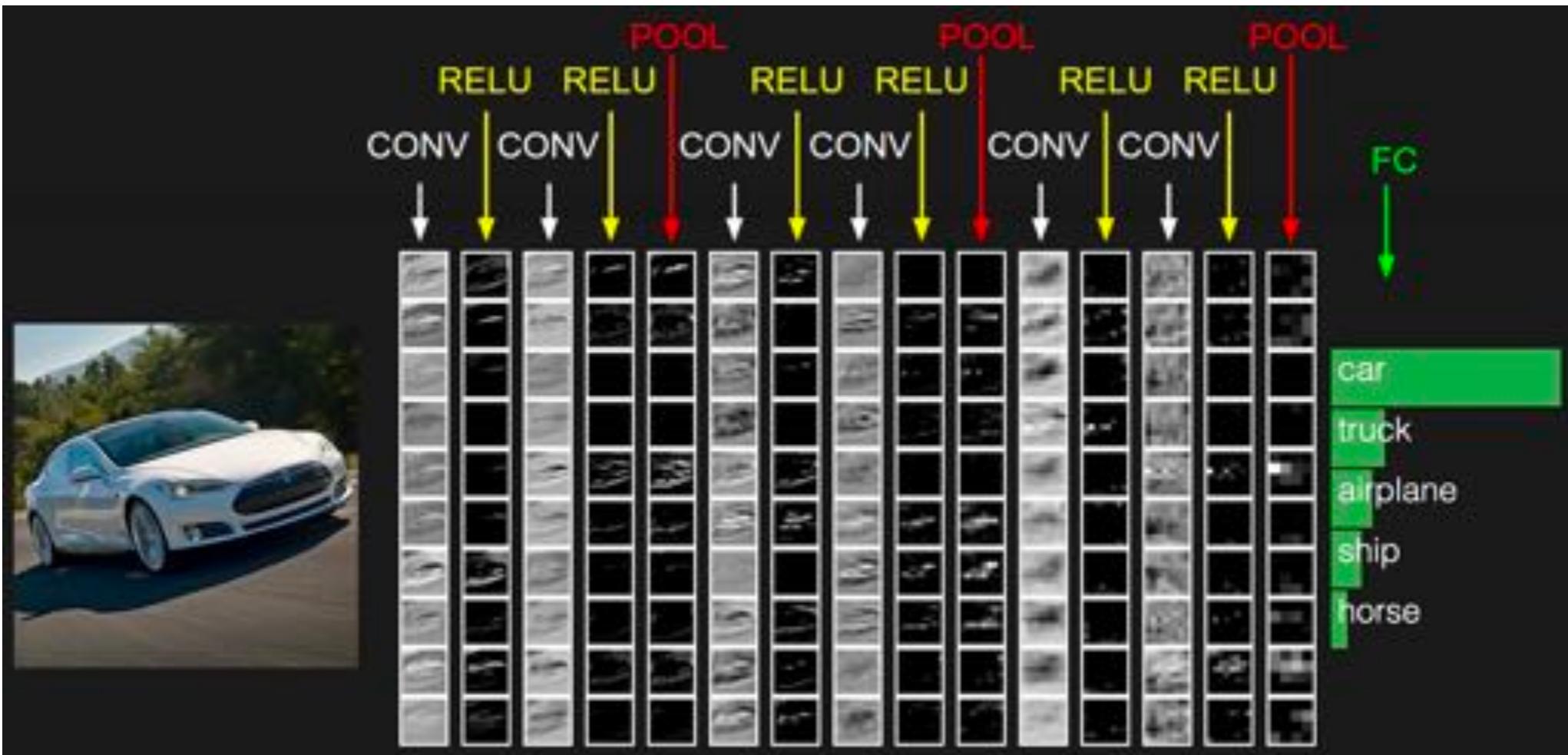
The diagram illustrates the process of max pooling. On the left, a 4x4 input matrix is shown with values: Row 1: 1, 1, 2, 4; Row 2: 5, 6, 7, 8; Row 3: 3, 2, 1, 0; Row 4: 1, 2, 3, 4. The matrix is color-coded in a checkerboard pattern. An arrow points from this input to a smaller 2x2 output matrix on the right. The output matrix has values: Top-left cell (row 1, column 1) is 6 (from the window [5, 6, 3, 2]), Top-right cell (row 1, column 2) is 8 (from the window [7, 8, 1, 0]), Bottom-left cell (row 2, column 1) is 3 (from the window [1, 2, 3, 4]), and Bottom-right cell (row 2, column 2) is 4 (from the window [3, 4]).

6	8
3	4

# Convolutional neural network



# CNN example



# Some useful resources

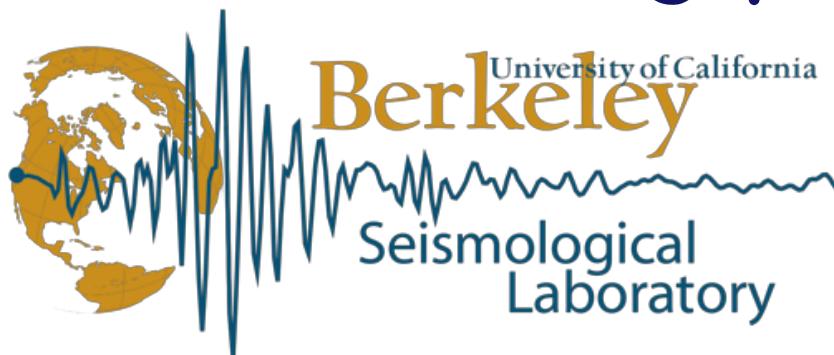
- <http://cs231n.github.io/convolutional-networks/>
- <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>
- <http://neuralnetworksanddeeplearning.com/>
- [Deep learning with Python](#) - Francois Chollet
- [Deep learning](#) - Ian Goodfellow
- Datasets put together by Men-Andrin and Zach Ross
  - <http://scedc.caltech.edu/research-tools/deeplearning.html>

Many pictures in this talk are from internet, I thank all the authors here!

Go to the notebook



Thanks  
kongqk@Berkeley.edu



<http://seismo.berkeley.edu/qingkaikong/>