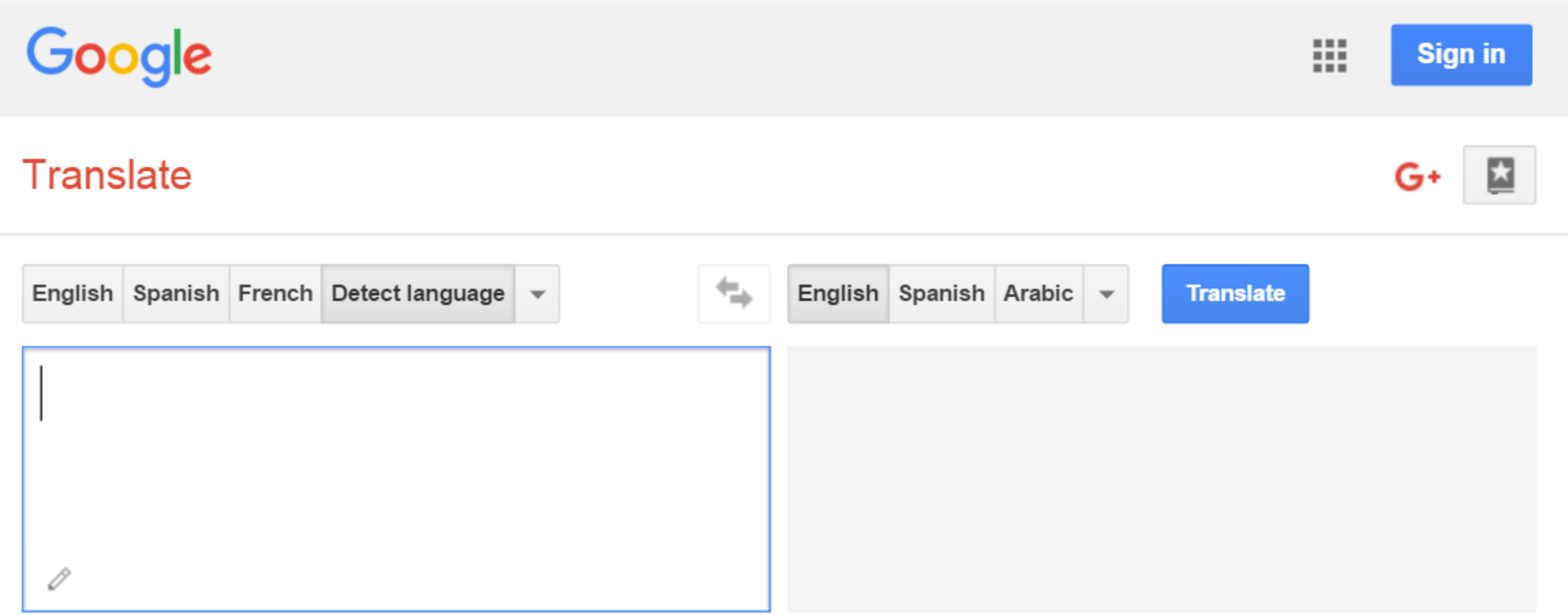
Language recognizer

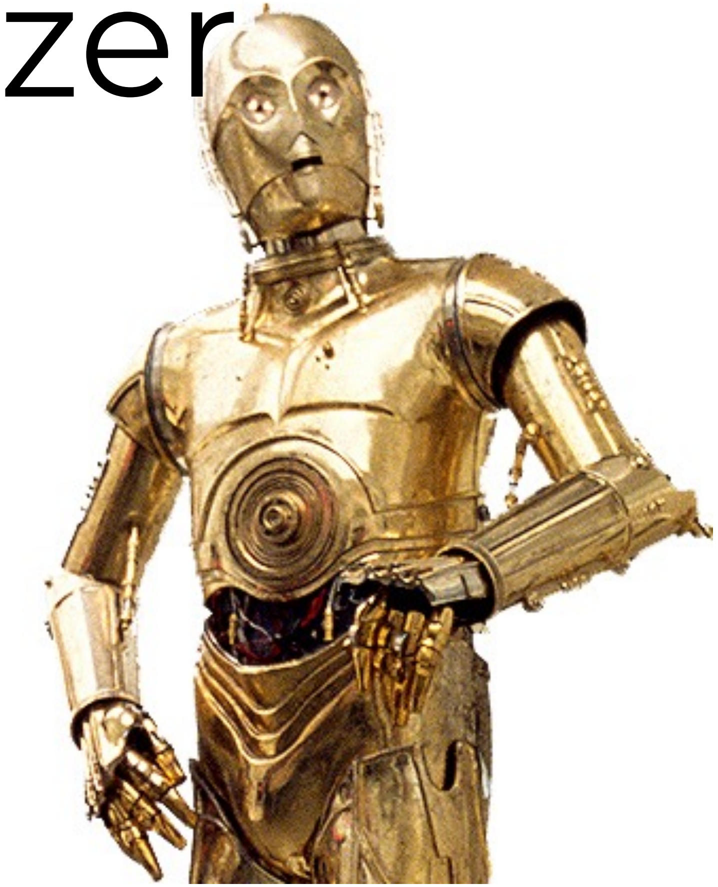

Google
Sign in
Translate
G+
English
Spanish
French
Detect language
English
Spanish
Arabic
Translate

Language recognizer

What is the language of this text?

A Csillagok haboruja egy uropera filmsorozatnak, irodalmi muveknek es szamitogepes jatekoknak a neve.

Language recognizer



What is the language of this text?

A Csillagok haboruja egy uropera filmsorozatnak, irodalmi muveknek es szamitogepes jatekoknak a neve.

This is Hungarian, of course!

Classification

Texts → LABEL

English, Norwegian,
French, …

The plan

- Get sample text from from Wikipedia pages (done)
- Calculate features frequencies of letter pairs
- Compare languages using their features
- Classify language find the most similar one

Constructing features

- Texts are not directly comparable
- Frequencies of pairs of letters

the three

_t	2
th	2
he	1

e_	2
hr	1
re	1

ee	1

Constructing features

- Frequencies depend on length of text
- Compute probabilities instead

the three

_t	0.2
th	0.2
he	0.1

e_	0.2
hr	0.1
re	0.1

ee	0.1

Distance between features

	th	e_	ee	el
English	0.3	0.2	0.2	0.1
Norwegian	0.0	0.2	0.1	0.3

Nearest neighbor classifier

- Given an unknown subject to classify
- Lookup all the known examples
- Find the closest example
- Predict the label that the closest example has

Nearest neighbour classifier

1. Transform data into features that are comparable
2. Define a distance measure
3. ???
4. PROFIT

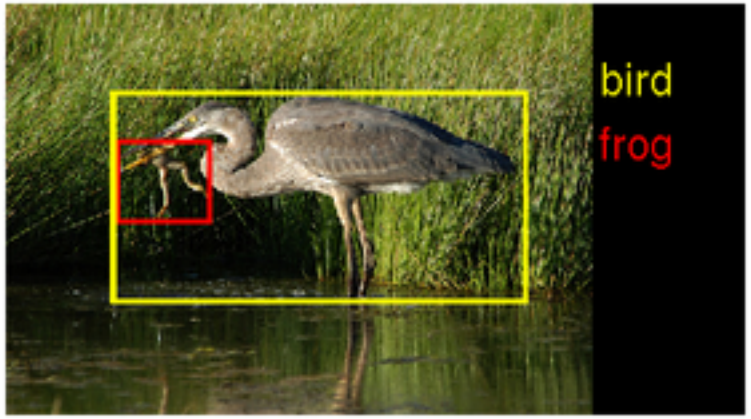
Coding

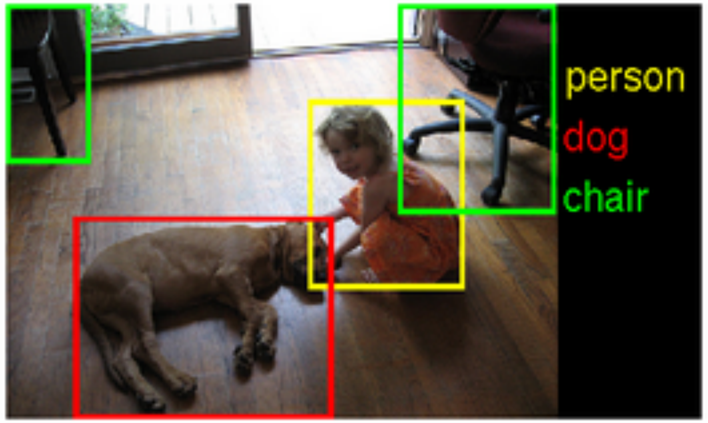
Classification with neural networks

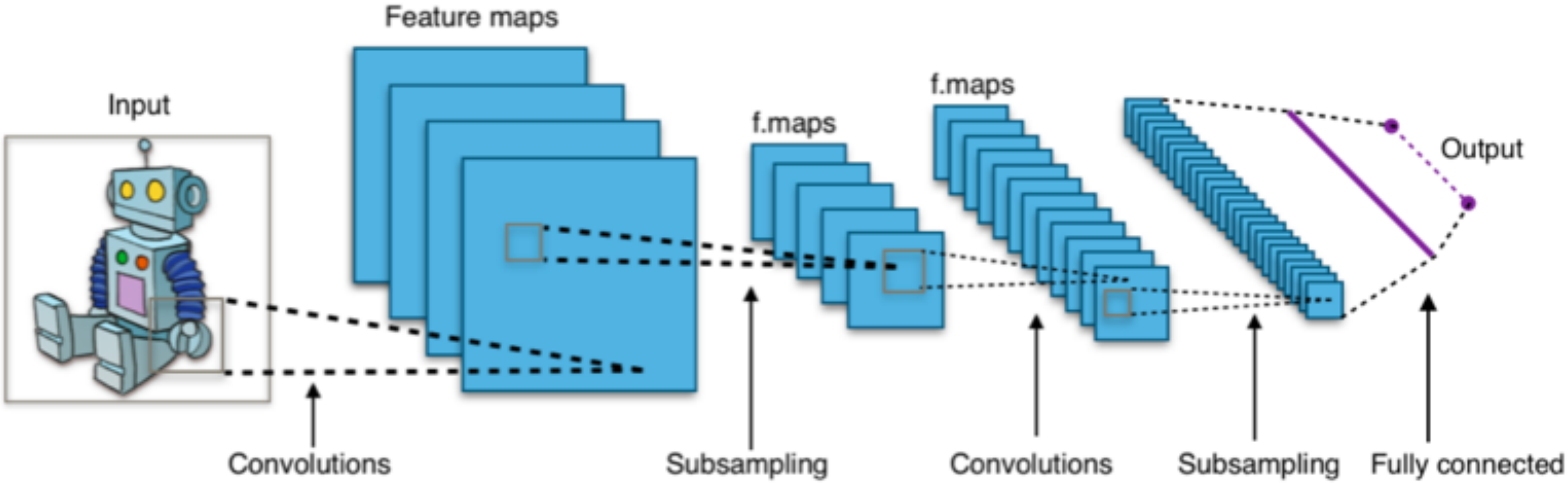
- K-nearest neighbours doesn’t actually learn anything

Perceptron

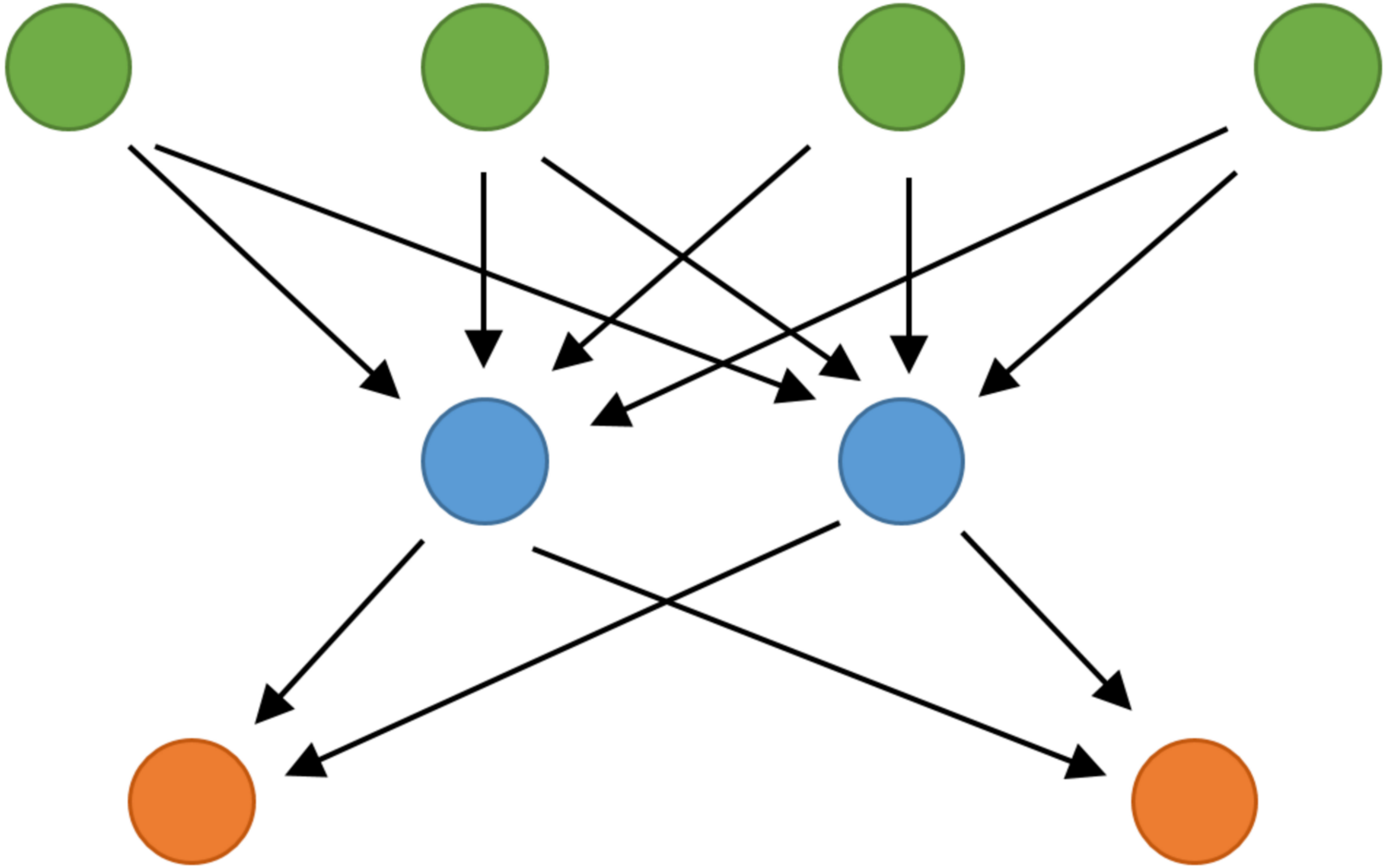
Logistic regression


bird
frog

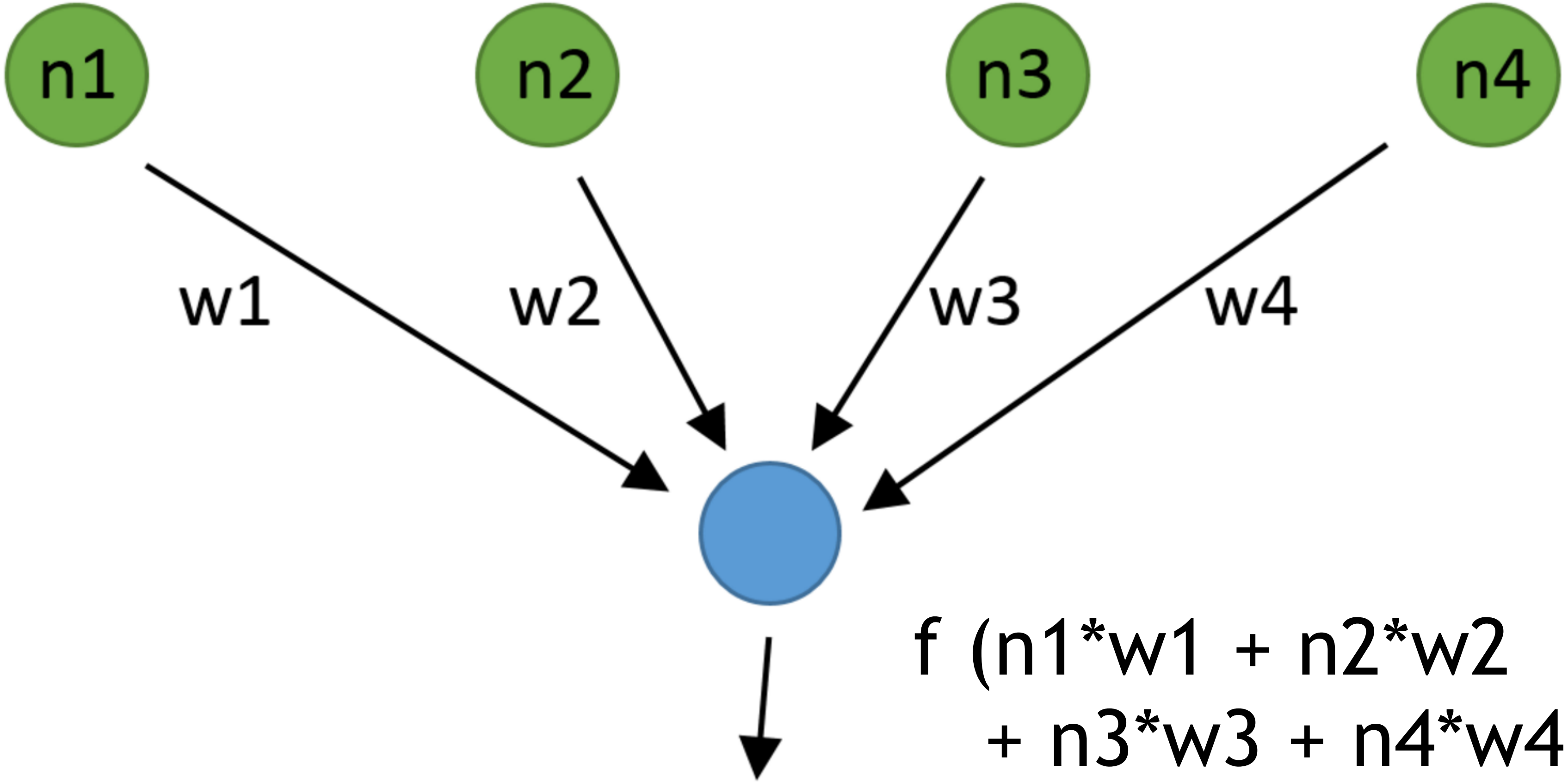

person
dog
chair


Feature maps
Input
f.maps
f.maps
Output
Convolutions
Subsampling
Convolutions
Subsampling
Fully connected

Neural networks for classification



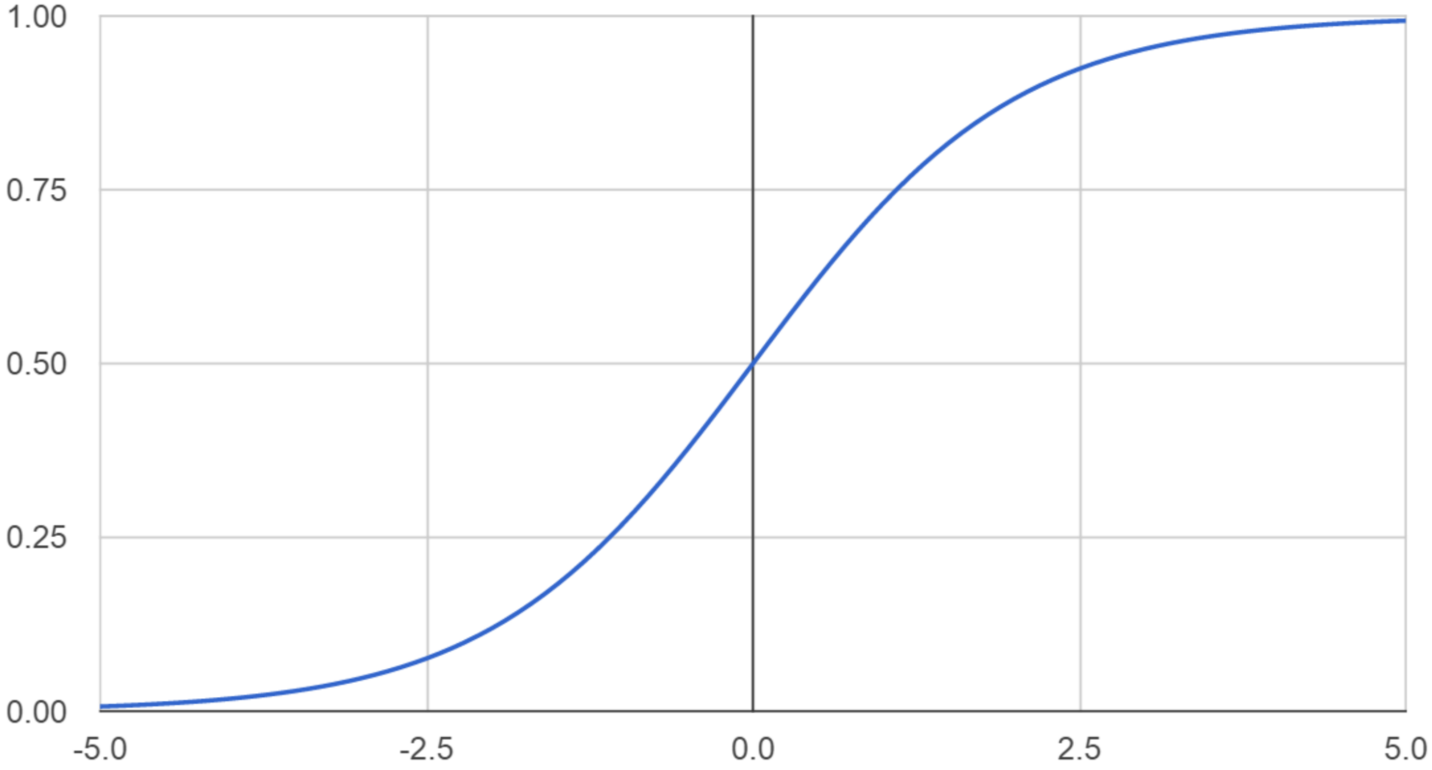
Neuron is a function



f (n1*w1 + n2*w2
+ n3*w3 + n4*w4)

Sigmoid function

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



Classification

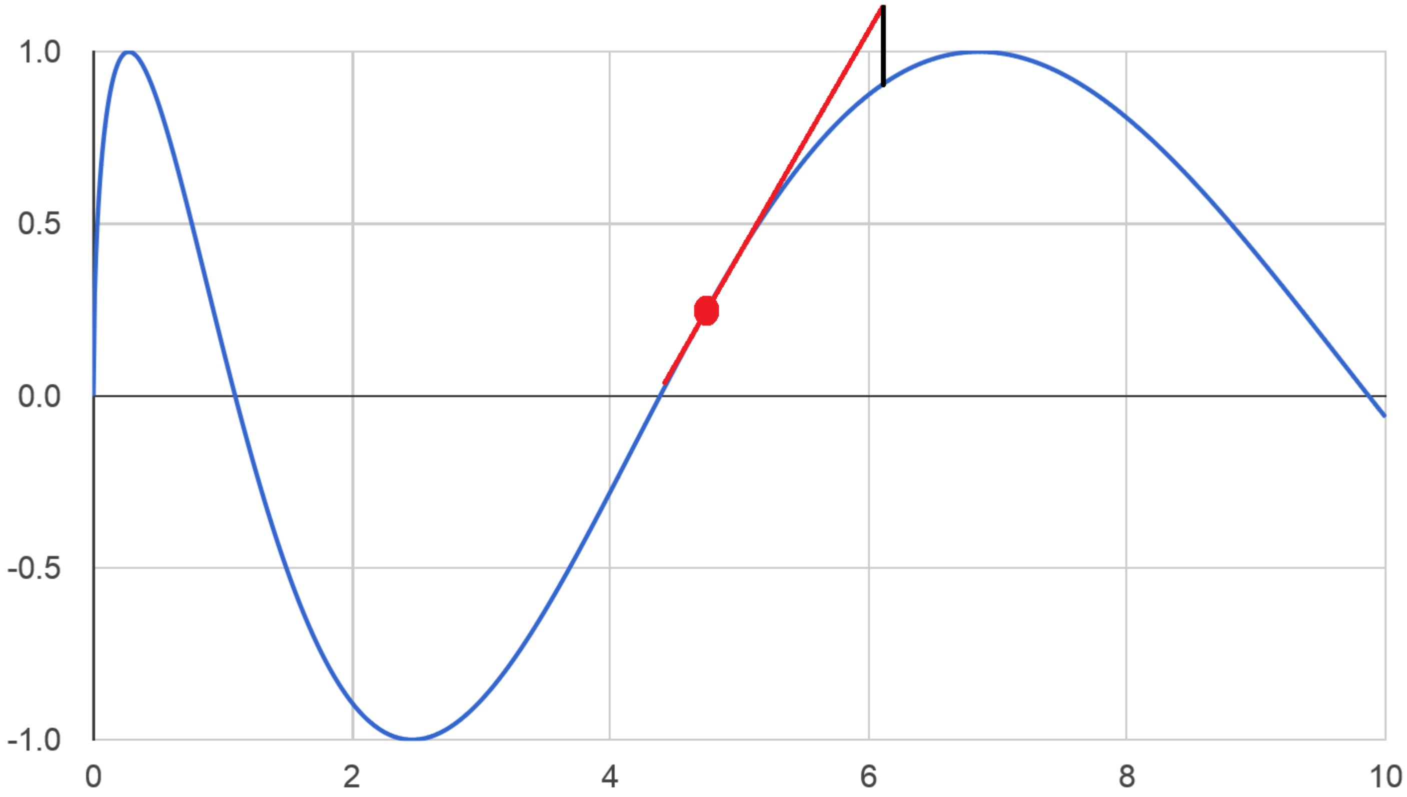
Texts → LABEL

0 for Language A
1 for Language B

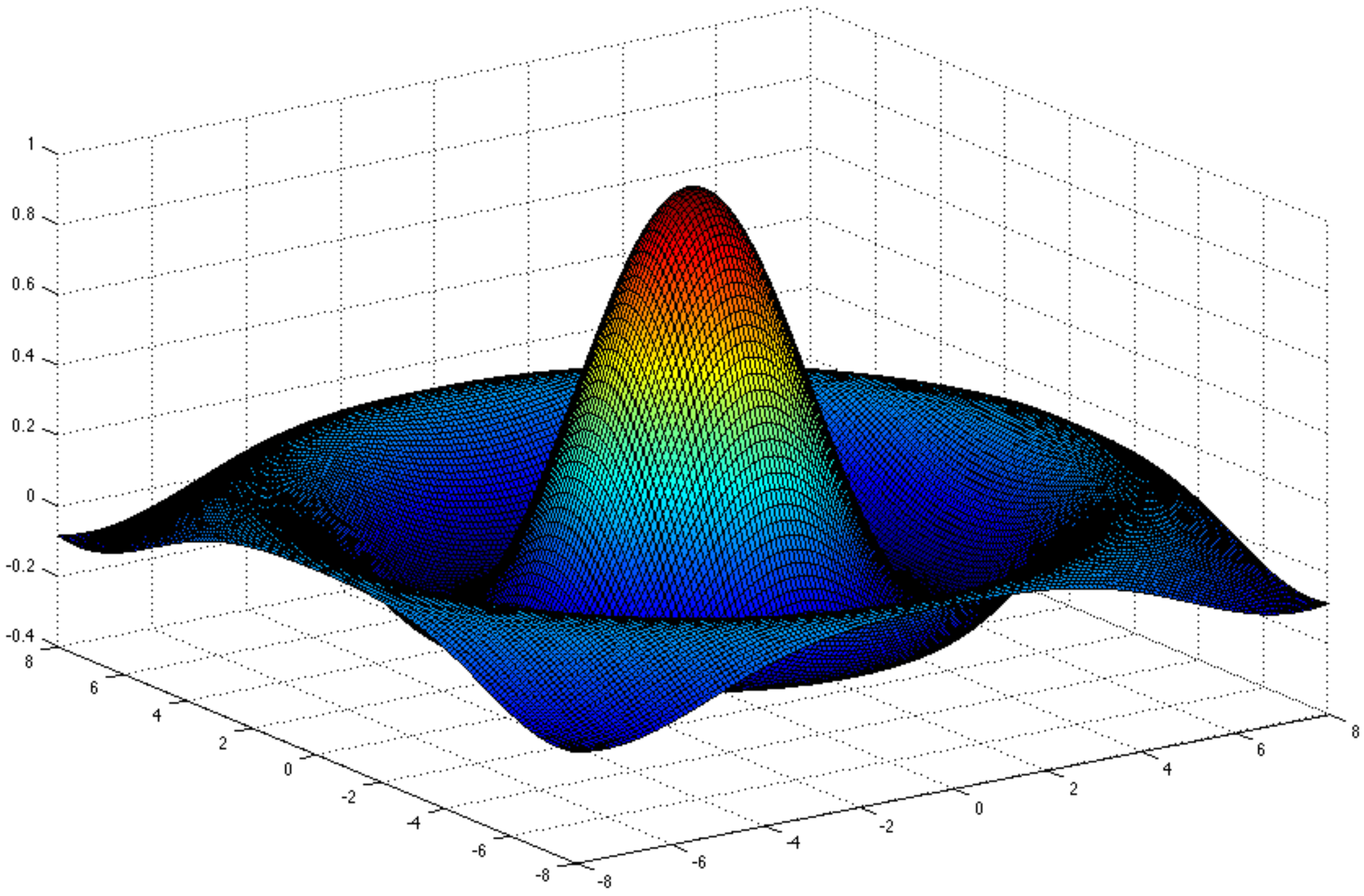
Training the perceptron

- Start with random weights
- Compute output of the perceptron
- How different is the output from the true label?
- Improve weights by gradient descent
- Iterate!

Gradient descent



Gradient descent


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
8
6
4
2
0
-2
-4
-6
-8
-8
-6
-4
-2
0
2
4
6
8