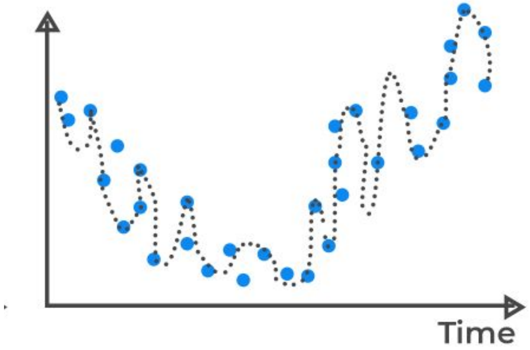
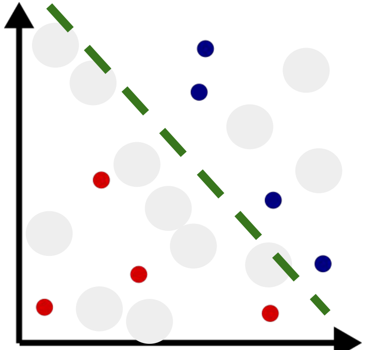
COM1011

Fundamentals of Machine Learning

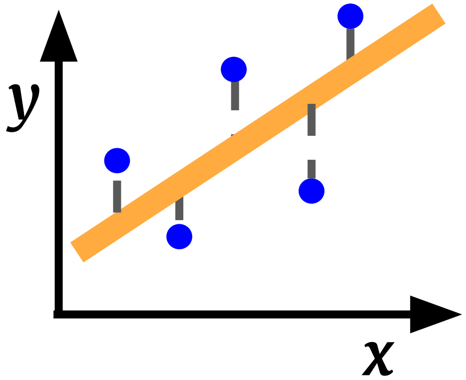
Previously on COM1011:



overfitting



train-test split



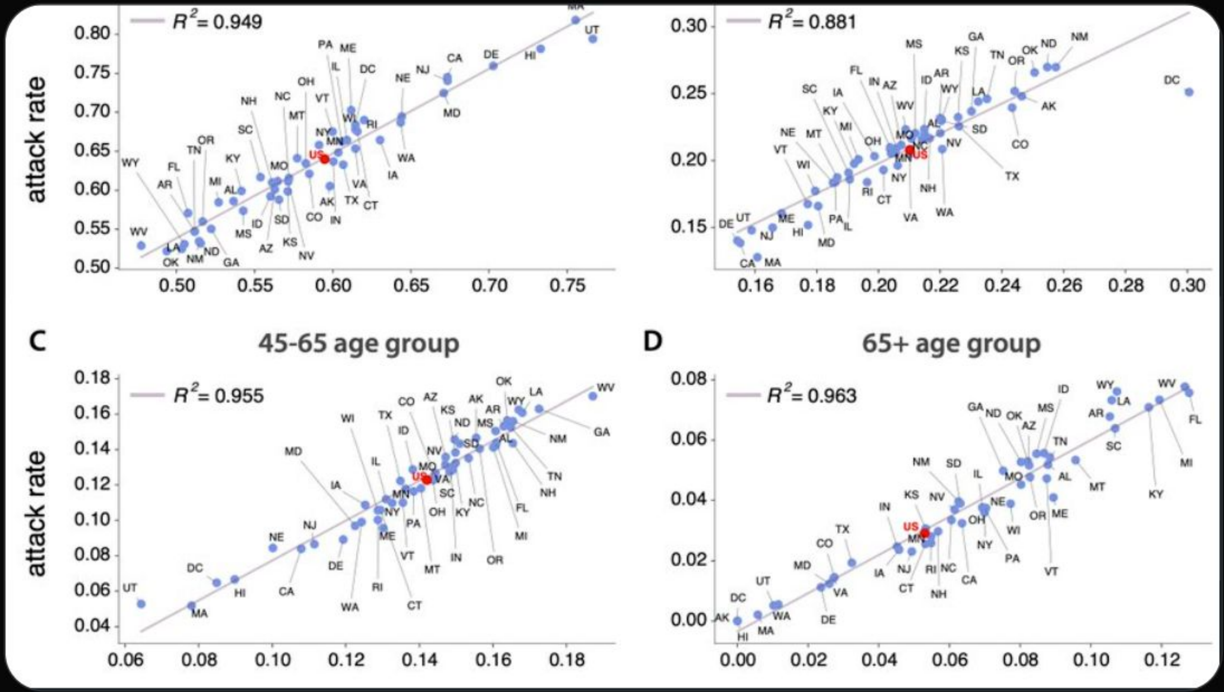
R^2

$$y = a_1x_1 + a_2x_2 + a_3x_3 + b$$

Yamir Moreno @cosnet_bifi · 12h

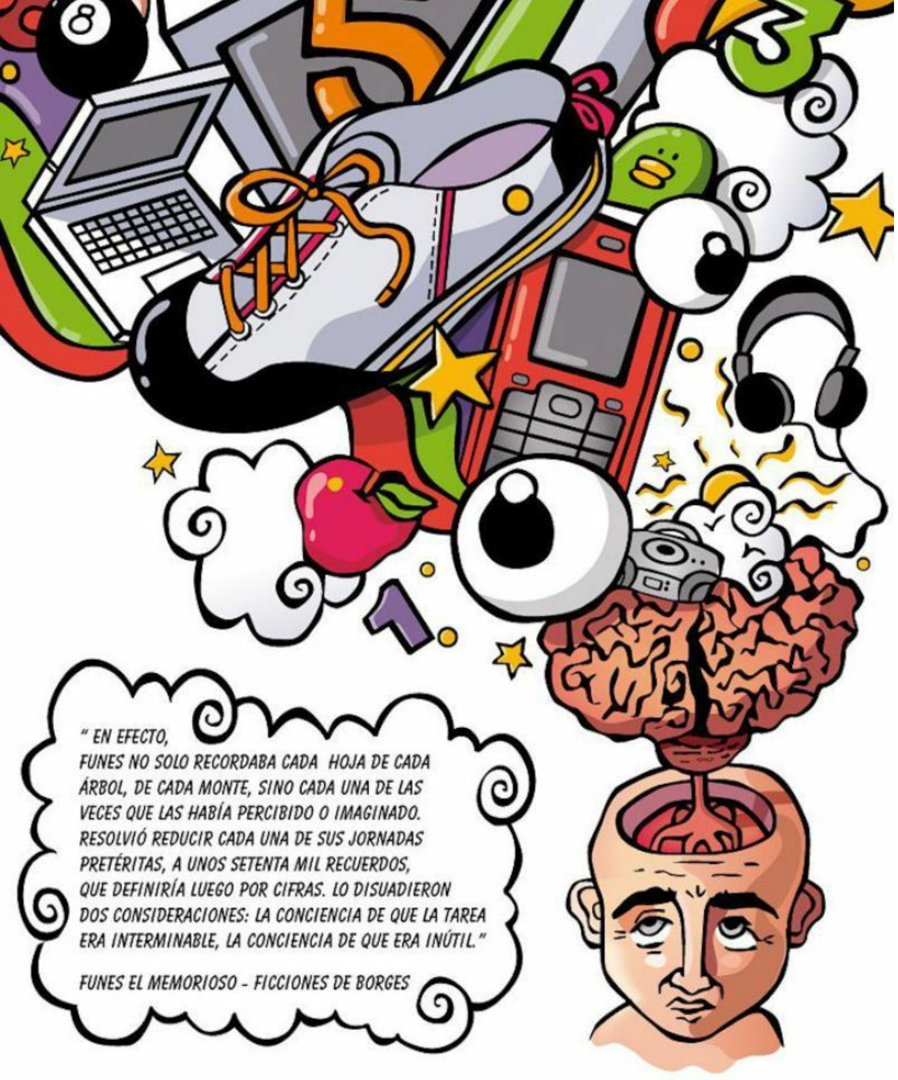
Replying to @cosnet_bifi

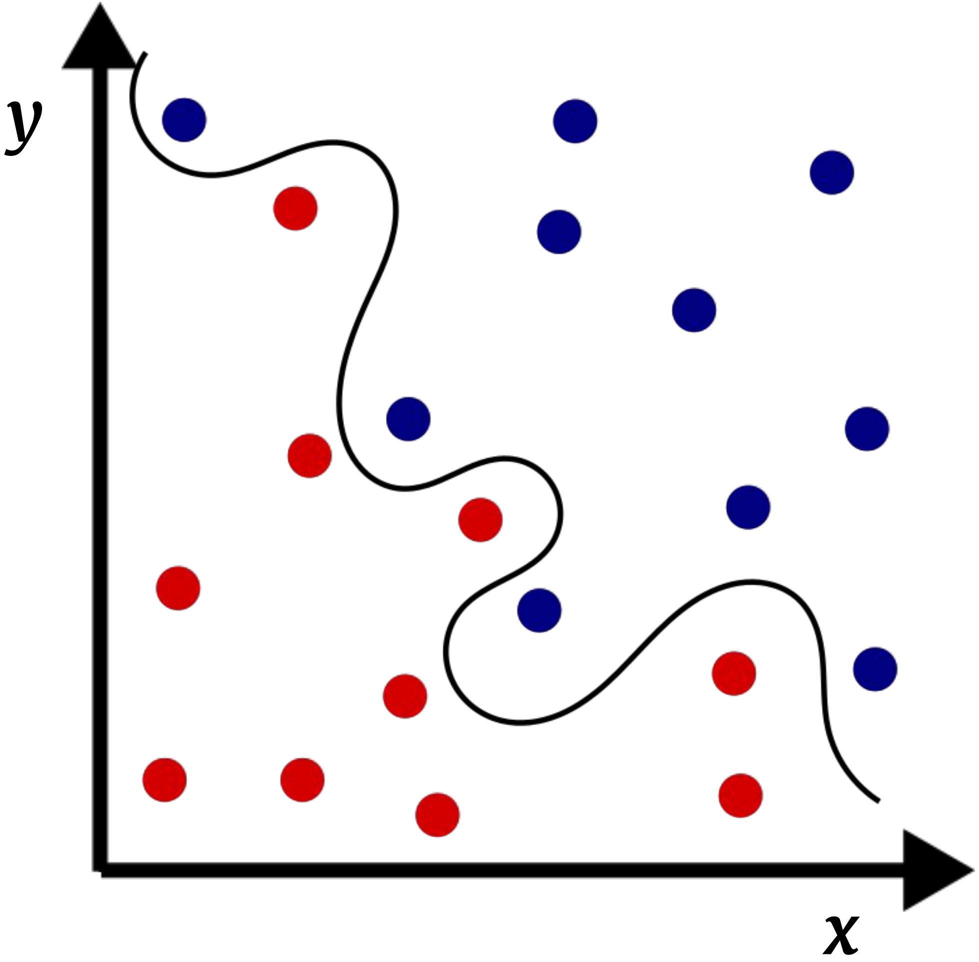
Our results clearly show that higher vaccine hesitancy ratios lead to larger outbreaks. This, however, cannot be translated directly into deaths since age plays a very important. Work with A. de Miguel & @SrAleta. Data from covidstates.org. (2/2)



Today:

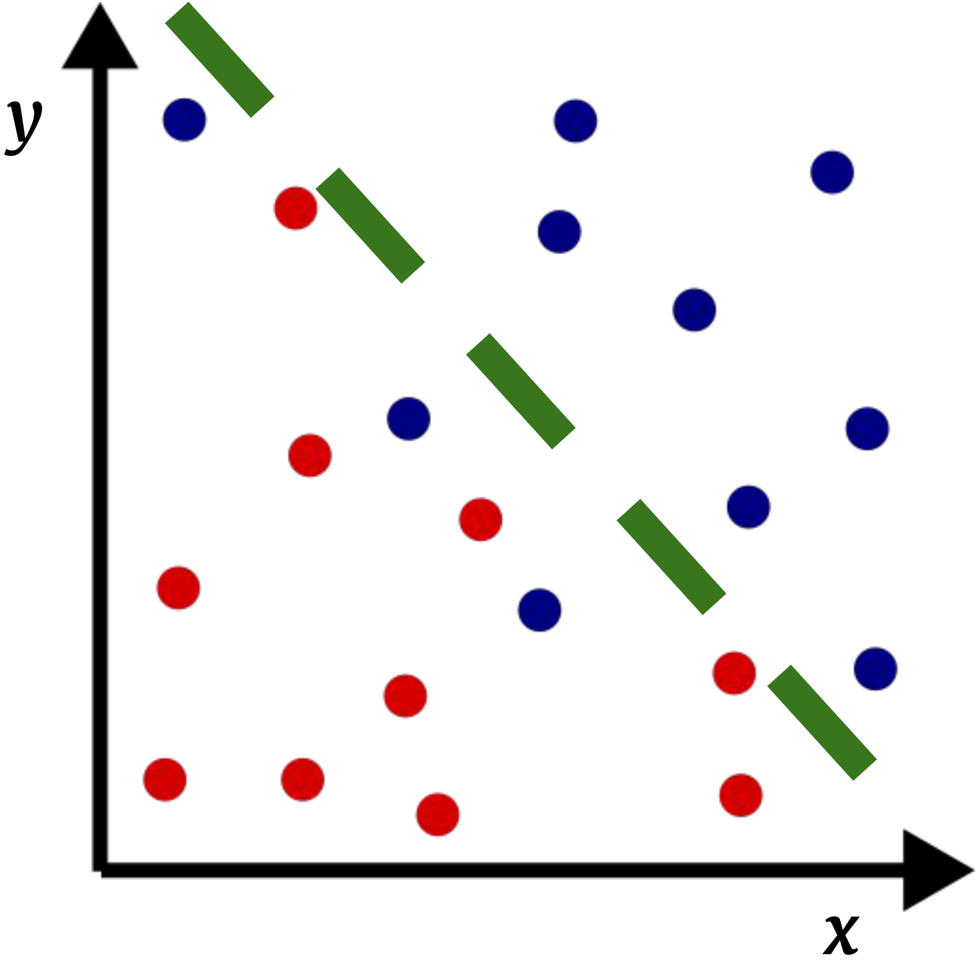
- Introduction to classification
- Logistic regression
- Linear classifiers
- The perceptron algorithm


" EN EFECTO,
FUNES NO SOLO RECORDABA CADA HOJA DE CADA ÁRBOL, DE CADA MONTE, SINO CADA UNA DE LAS VECES QUE LAS HABÍA PERCIBIDO O IMAGINADO. RESOLVIÓ REDUCIR CADA UNA DE SUS JORNADAS PRETÉRITAS, A UNOS SETENTA MIL RECUERDOS, QUE DEFINIRÍA LUEGO POR CIFRAS. LO DISUADIERON DOS CONSIDERACIONES: LA CONCIENCIA DE QUE LA TAREA ERA INTERMINABLE, LA CONCIENCIA DE QUE ERA INÚTIL."
FUNES EL MEMORIOSO - FICCIONES DE BORGES


y
x

Train-test split

- Train your algorithm on part of the data
- Test it on another part
- Splits are usually around 80% train, 20% test
- This can be used with all sorts of ML algorithms

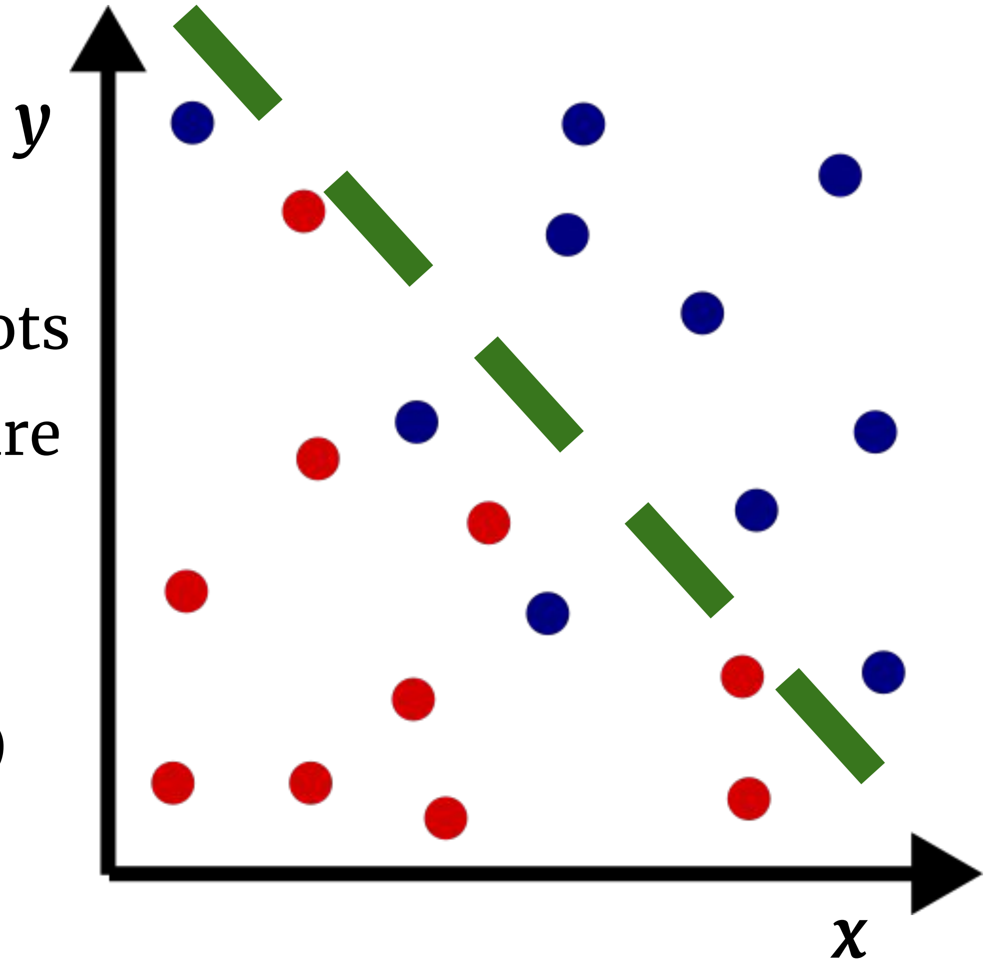


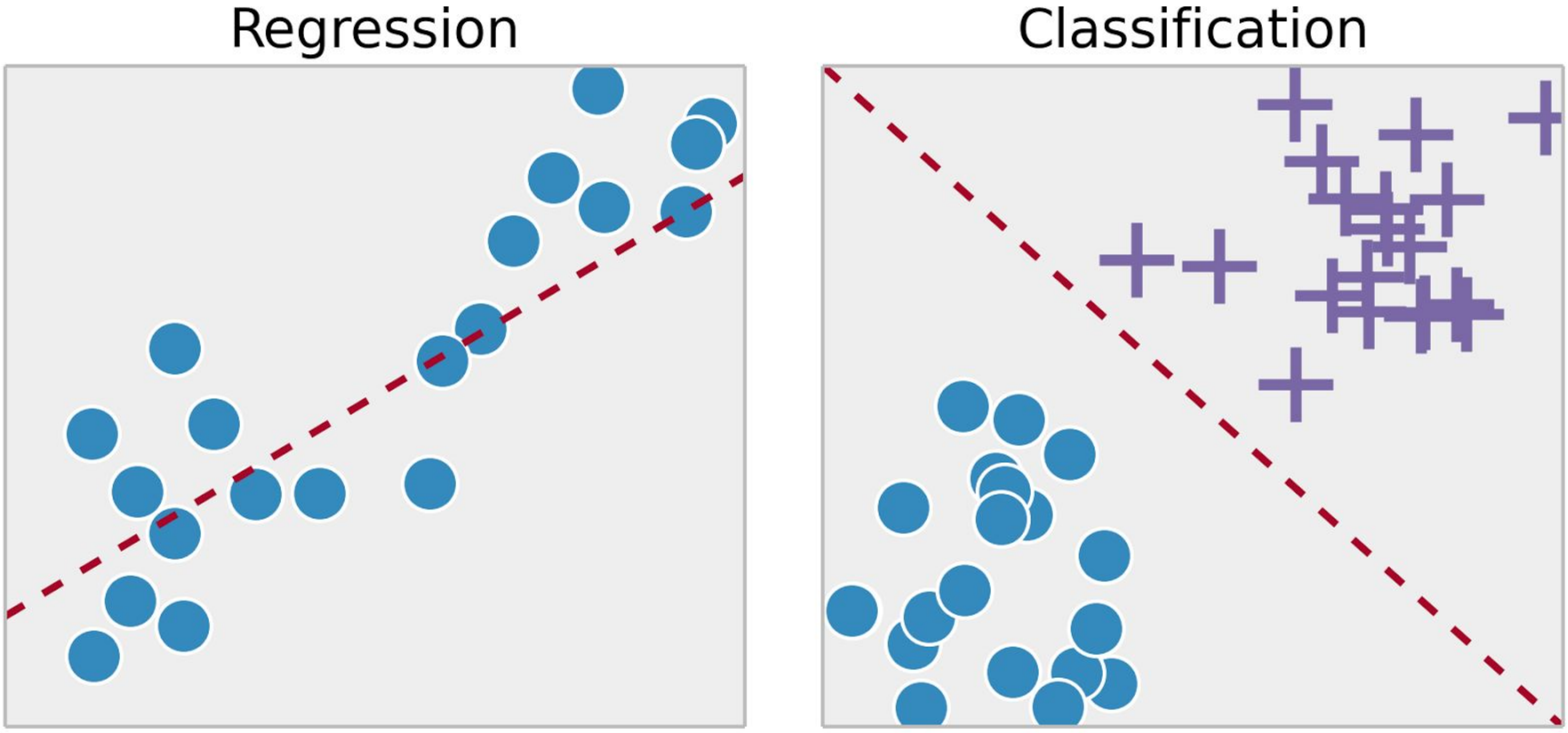
Classification

- Using x and y only
- Can you predict which dots are **red** and which ones are **blue**?

OR:

- Can you *classify* the (x , y) pairs into red or blue?

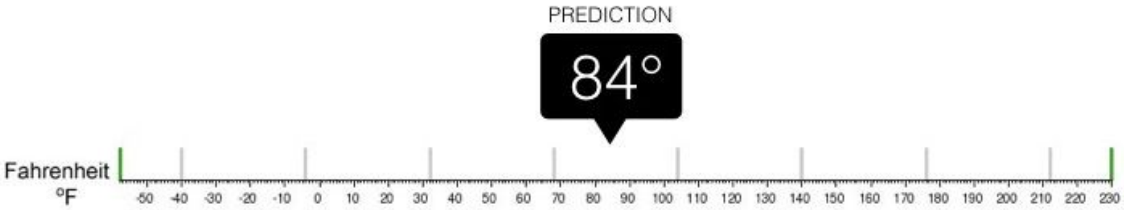



Regression
Classification

DATA

Regression

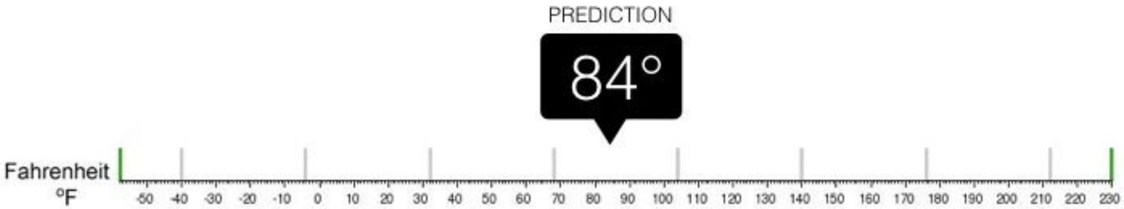
What is the temperature going to be tomorrow?



DATA

Regression

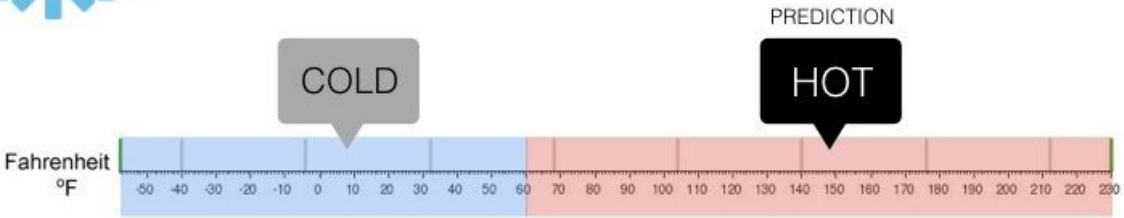
What is the temperature going to be tomorrow?



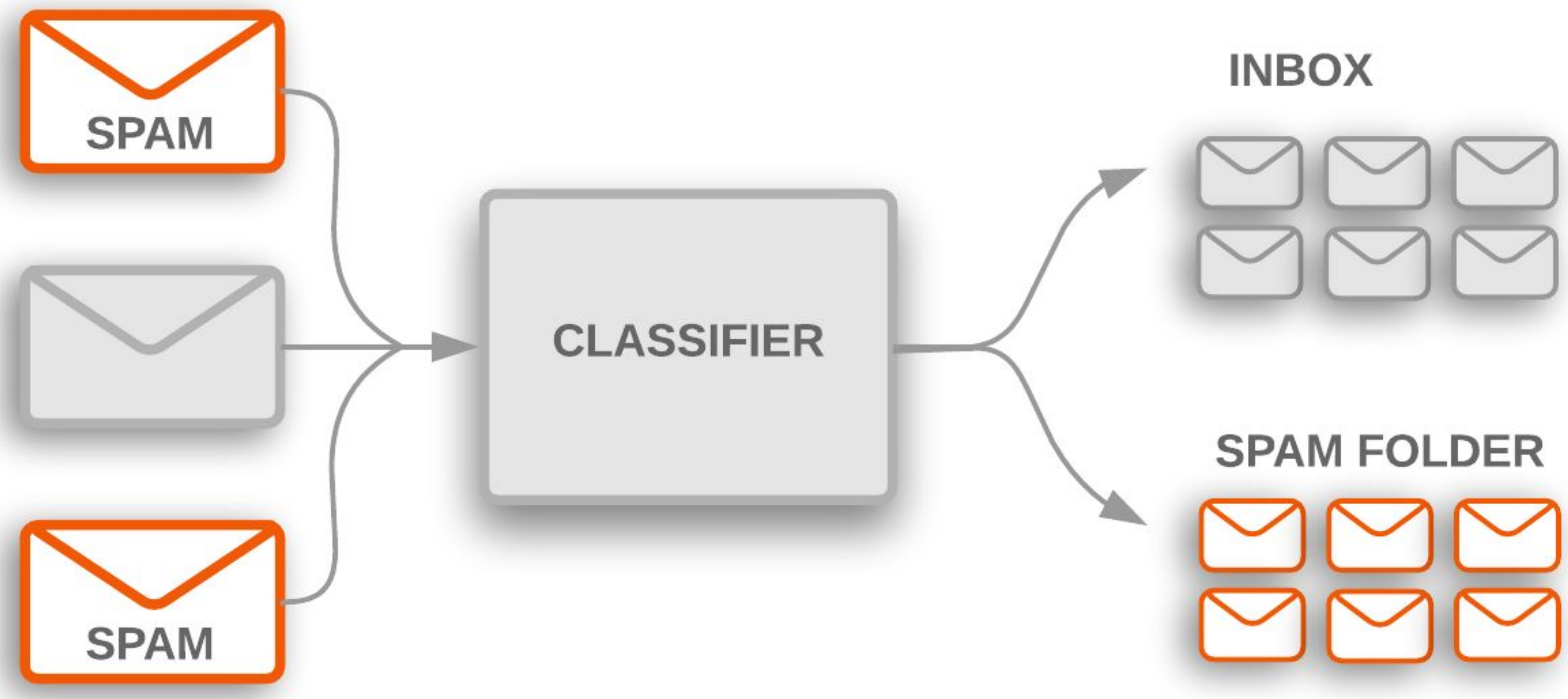
DATA

Classification

Will it be Cold or Hot tomorrow?



Classification is used everywhere



Classification is used everywhere

New York

New York

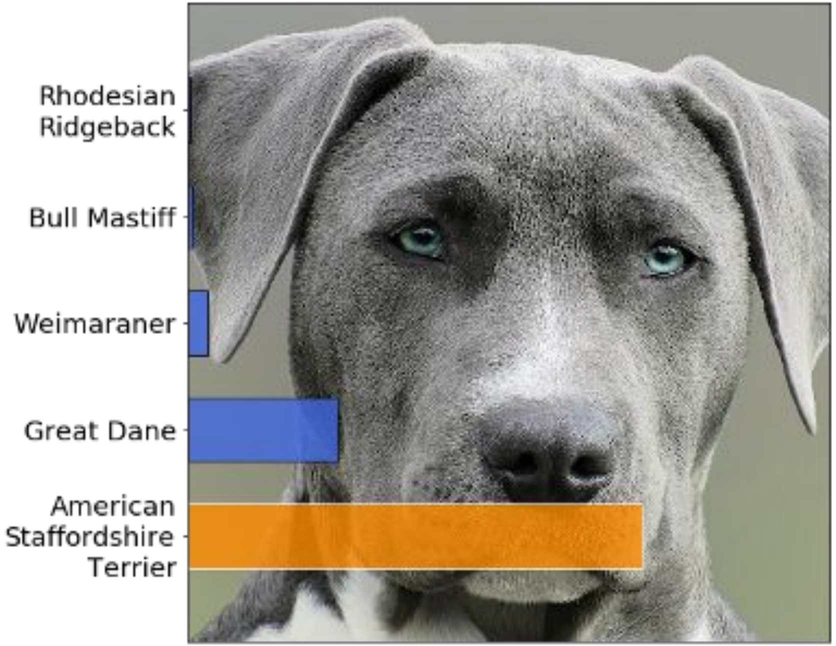
San Francisco

San Francisco

beds	bath	price	year_built	sqft	price_per_sqft	elevation	city
2.0	1.0	999000	1960	1000	999	10	New York
2.0	2.0	2750000	2006	1418	1939	0	New York
2.0	1.0	695000	1923	1045	665	106	San Francisco
3.0	2.0	1650000	1922	1483	1113	106	San Francisco
1.0	1.0	649000	1983	850	764	163	?

?

Image classification



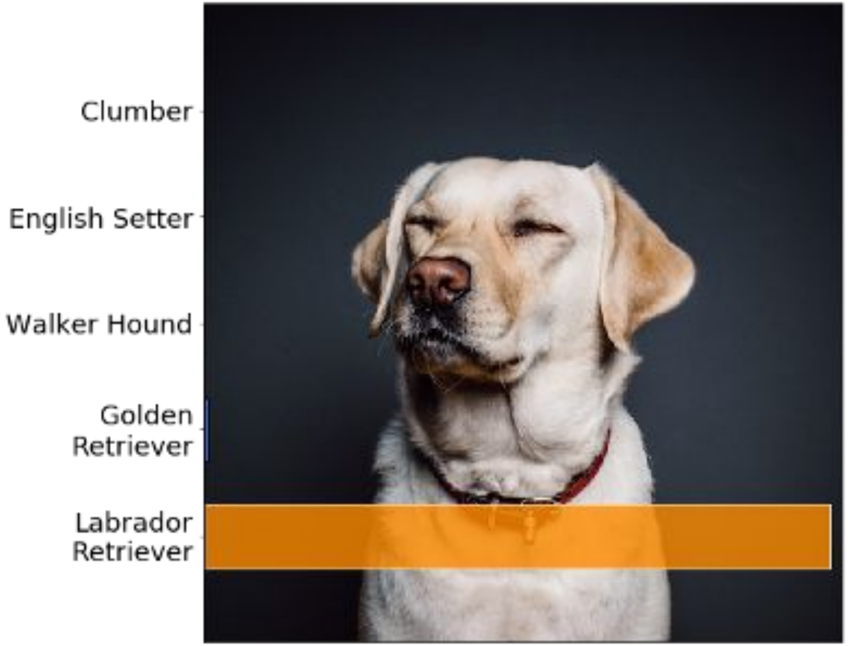


Image classification

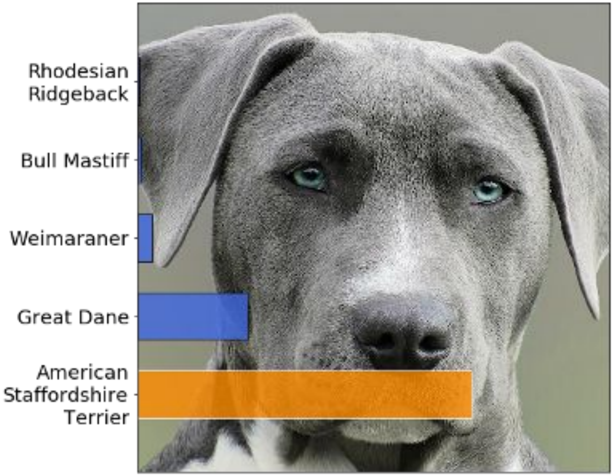
Chihuahua or blueberry muffin

Image classification

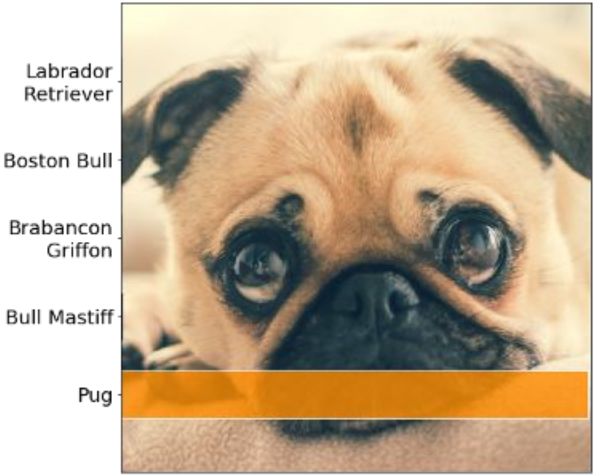
Chihuahua or blueberry muffin

Sheepdog or mop

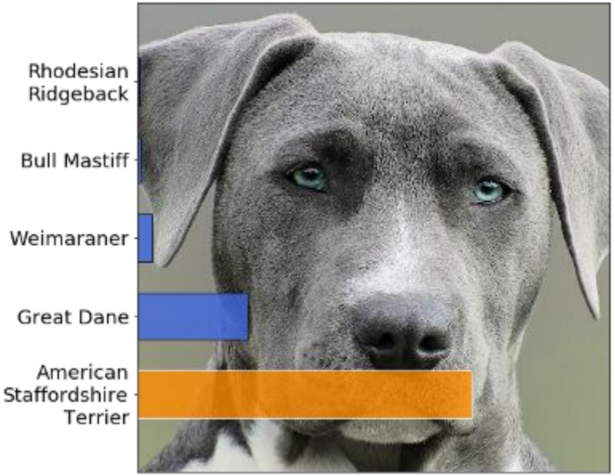
Image classification

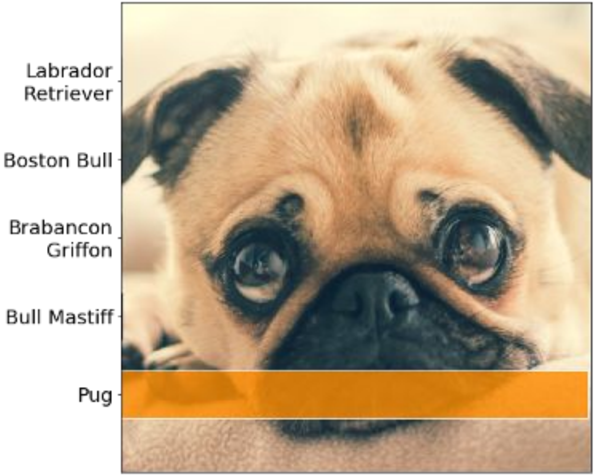


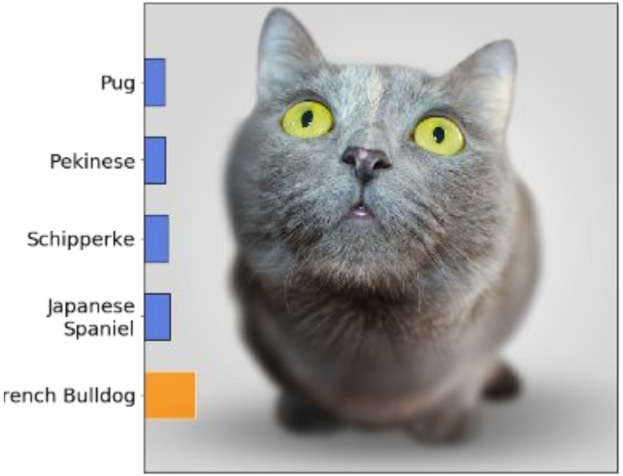
Rhodesian Ridgeback
Bull Mastiff
Weimaraner
Great Dane
American Staffordshire Terrier



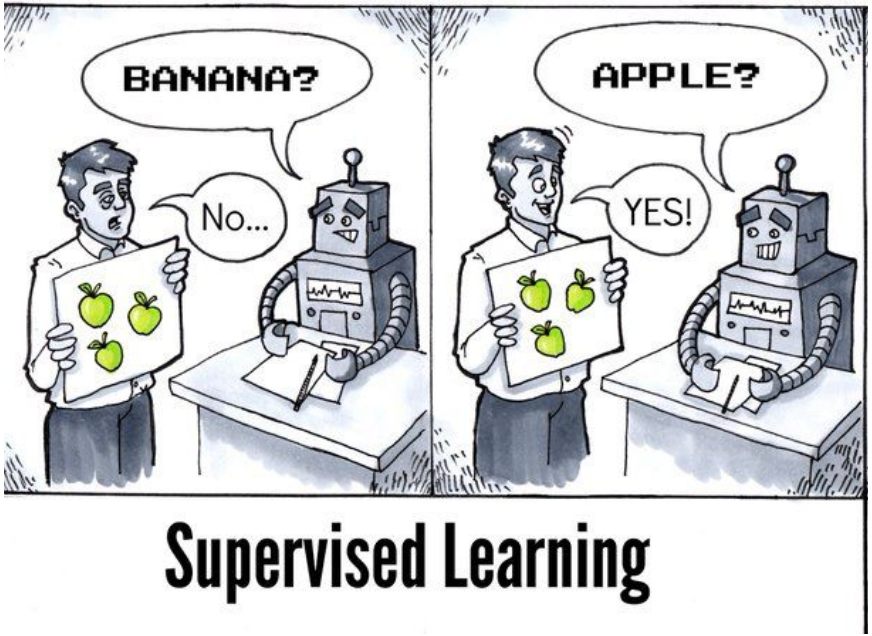
Labrador Retriever
Boston Bull
Brabancon Griffon
Bull Mastiff
Pug


Rhodesian Ridgeback
Bull Mastiff
Weimaraner
Great Dane
American Staffordshire Terrier


Labrador Retriever
Boston Bull
Brabancon Griffon
Bull Mastiff
Pug


Pug
Pekinese
Schipperke
Japanese Spaniel
rench Bulldog

Classification is supervised learning.



1. You train the algorithm using some correct examples, or “ground truth data”

2. A trained algorithm can make predictions about new data

Illustration by @Ciaraioch

Classification is supervised learning.

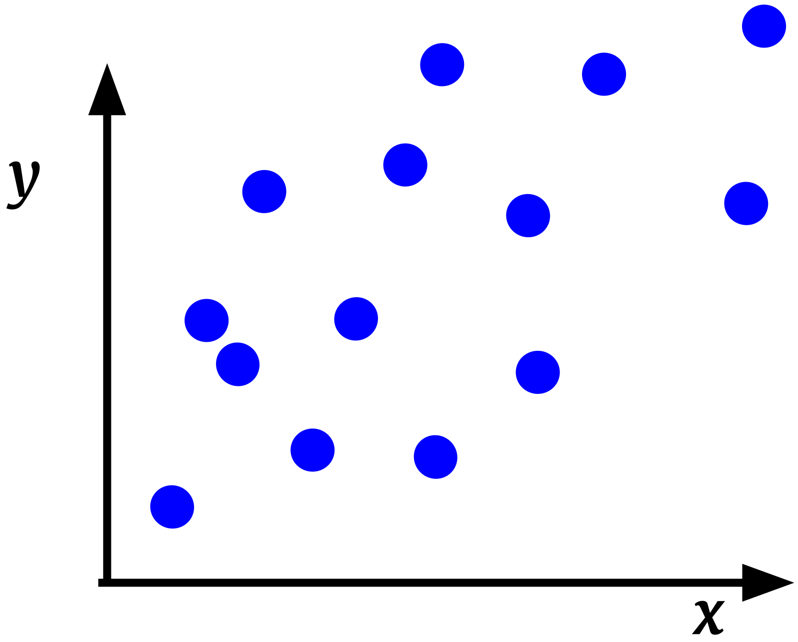
Input: smells

Output: banana or apple?

1. You train the algorithm using some correct examples, or “ground truth data”

2. A trained algorithm can make predictions about new data

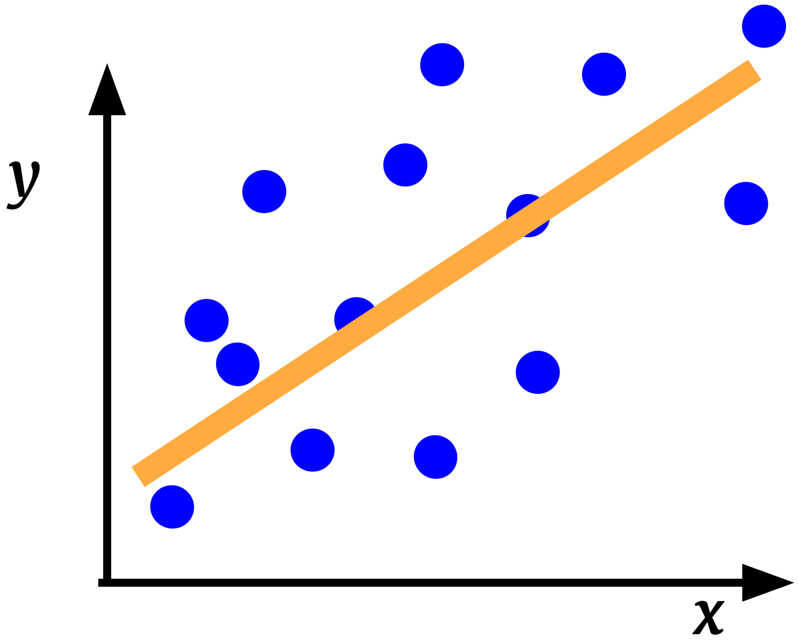
Regression is also supervised learning.



1. You train the algorithm using some correct examples, or "ground truth data"

2. A trained algorithm can make predictions about new data

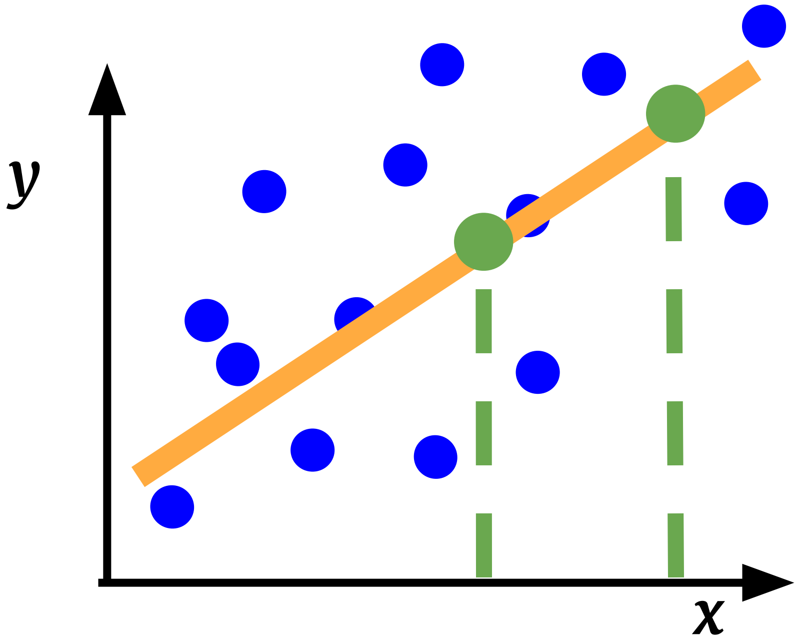
Regression is also supervised learning.



1. You train the algorithm using some correct examples, or “ground truth data”

2. A trained algorithm can make predictions about new data

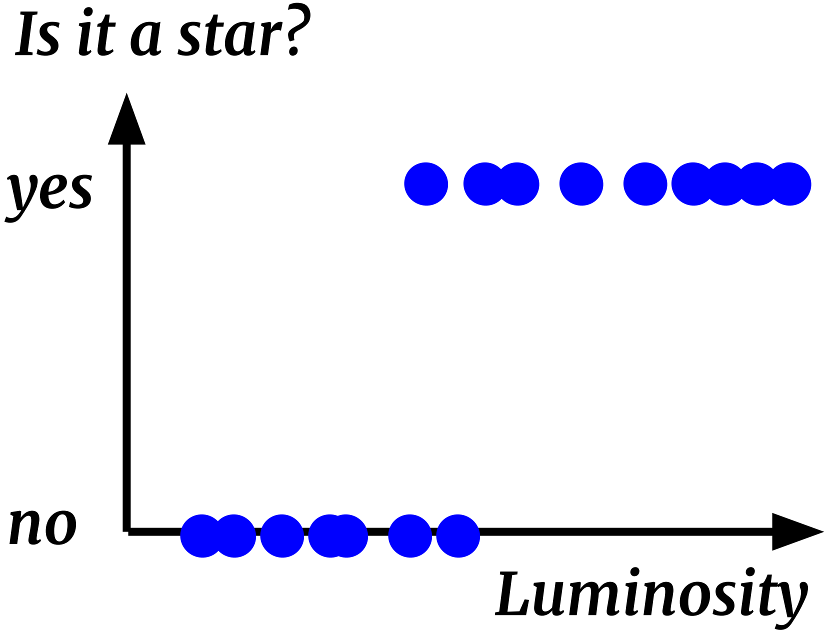
Regression is also supervised learning.



1. You train the algorithm using some correct examples, or “ground truth data”

2. A trained algorithm can make predictions about new data

First classifier: Logistic Regression

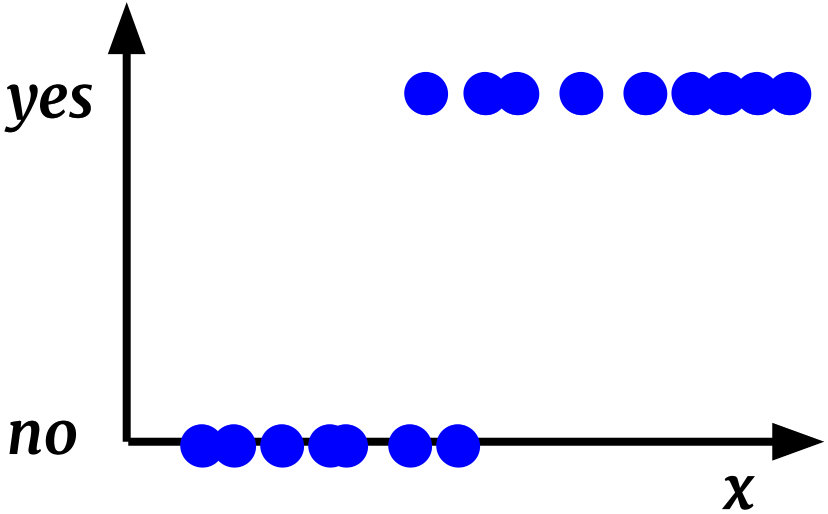


The input:

x = continuous data

y = binary data

Logistic Regression

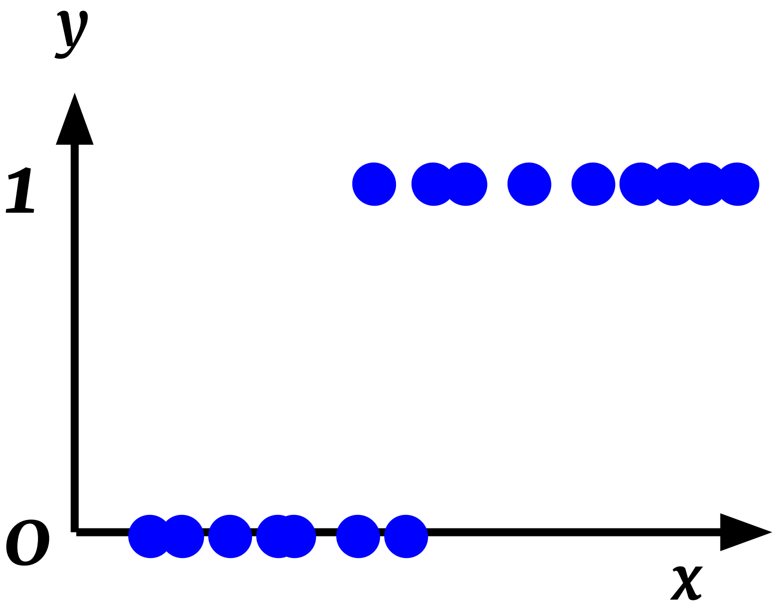


The input:

x = continuous data

y = binary data

Logistic Regression

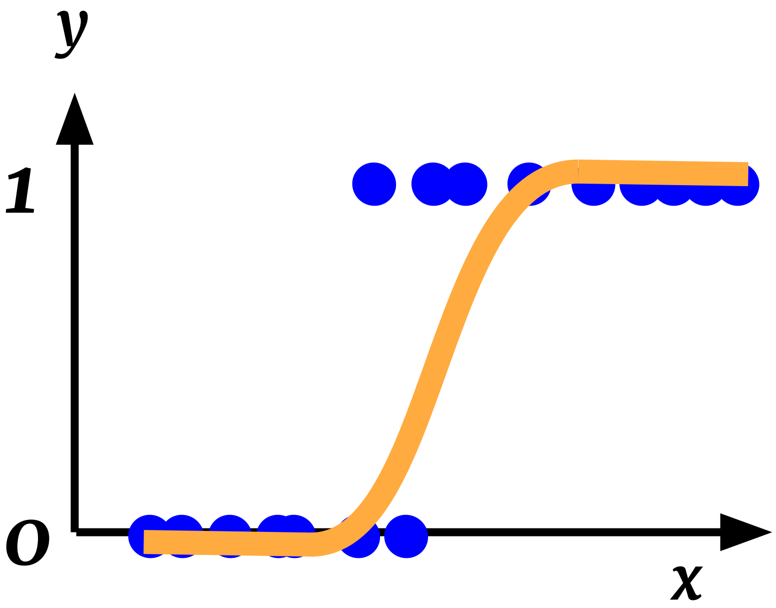


The input:

x = continuous data

y = binary data

Logistic Regression



The input:

x = continuous data

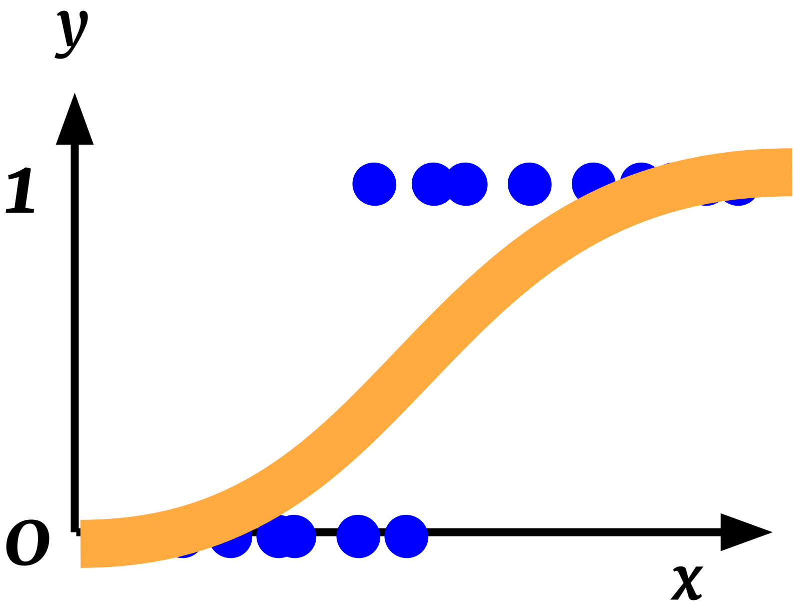
y = binary data

The model:

$$y = \frac{1}{1 + e^{-(ax+b)}}$$

Play with the parameters at: desmos.com/calculator

Logistic Regression



The input:

x = continuous data

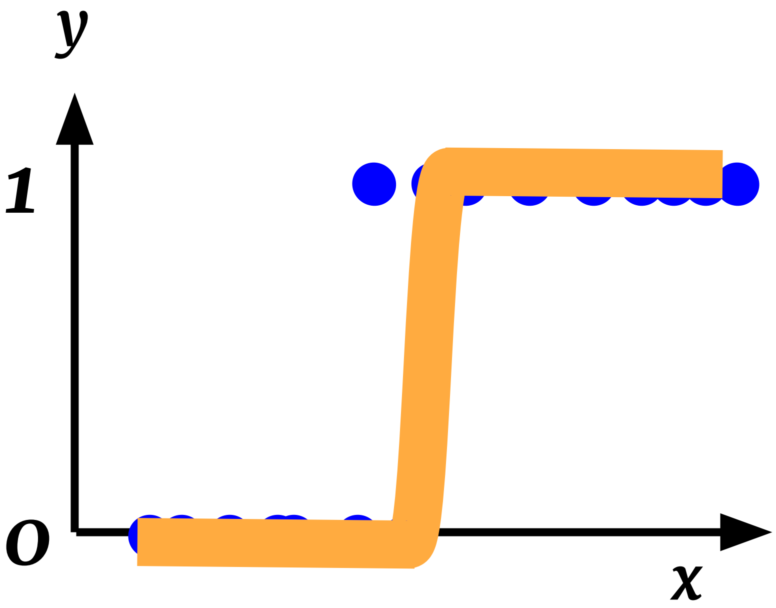
y = binary data

The model:

$$y = \frac{1}{1 + e^{-(ax+b)}}$$

Play with the parameters at: desmos.com/calculator

Logistic Regression



The input:

x = continuous data

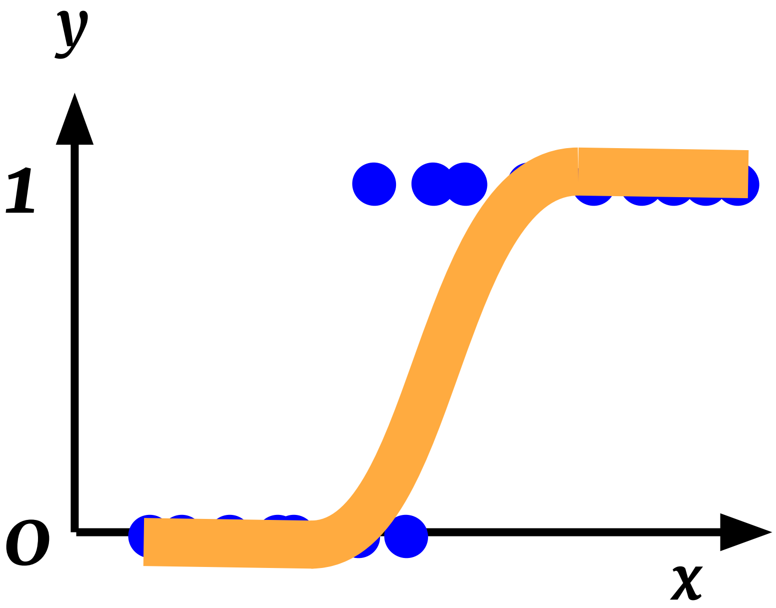
y = binary data

The model:

$$y = \frac{1}{1 + e^{-(ax+b)}}$$

Play with the parameters at: desmos.com/calculator

Logistic Regression



The input:

x = continuous data

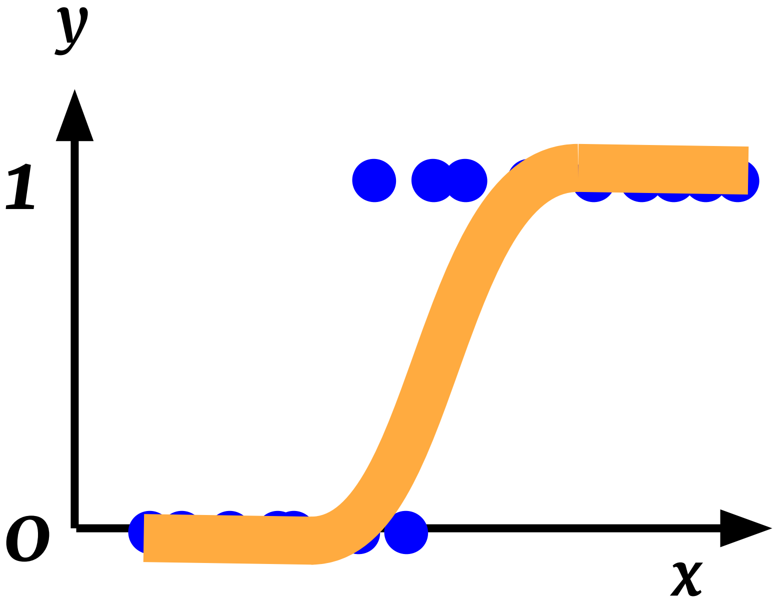
y = binary data

The model:

$$y = \frac{1}{1 + e^{-(ax+b)}}$$

Play with the parameters at: desmos.com/calculator

Logistic Regression



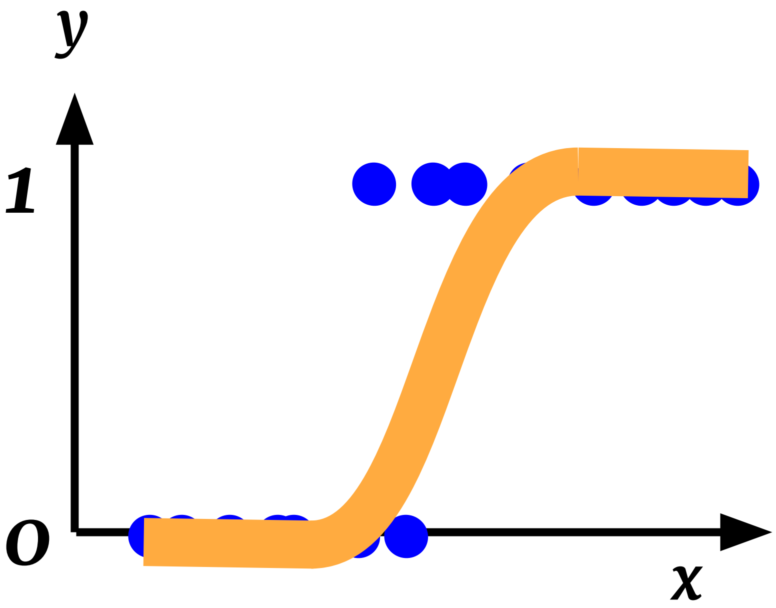
The model:

$$y = \frac{1}{1 + e^{-(ax+b)}}$$

The interpretation:

The model gives the probability of ***y* = 1**, for a given value of ***x***

Logistic Regression



The model:

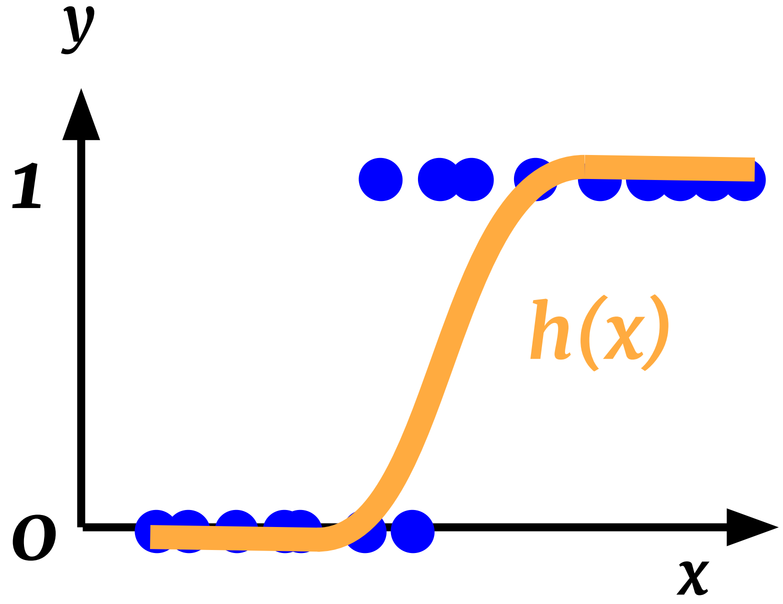
$$y = \frac{1}{1 + e^{-(\sum_i a_i\, x_i + b)}}$$

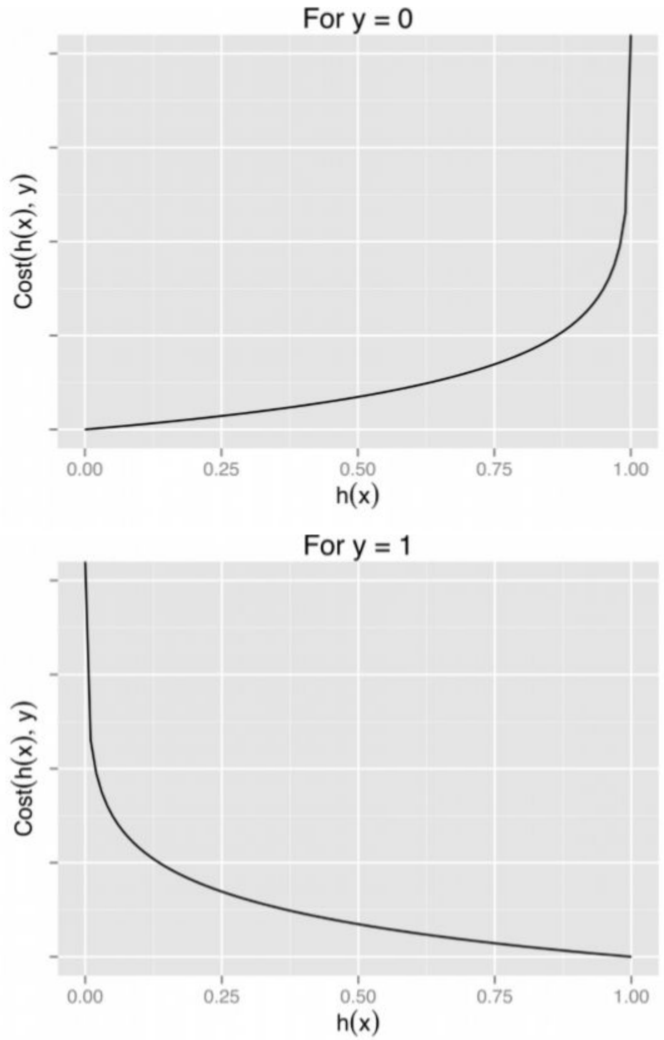
The interpretation:

The model gives the probability of ***y* = 1**, for a given value of x_i

Cost function:

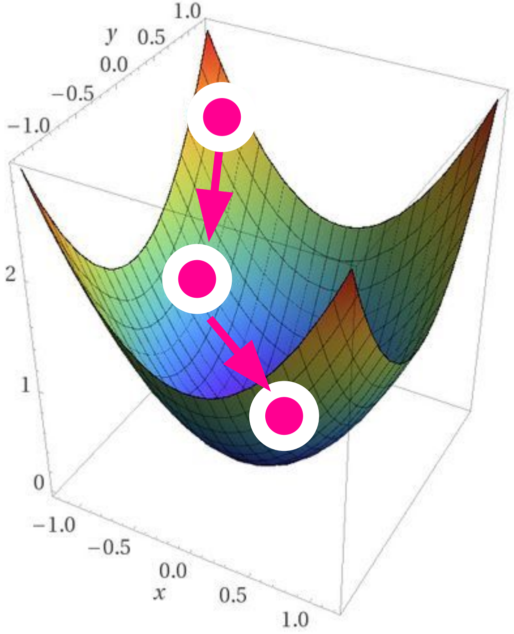
```
log_reg.fit(X_train, y_train)
```

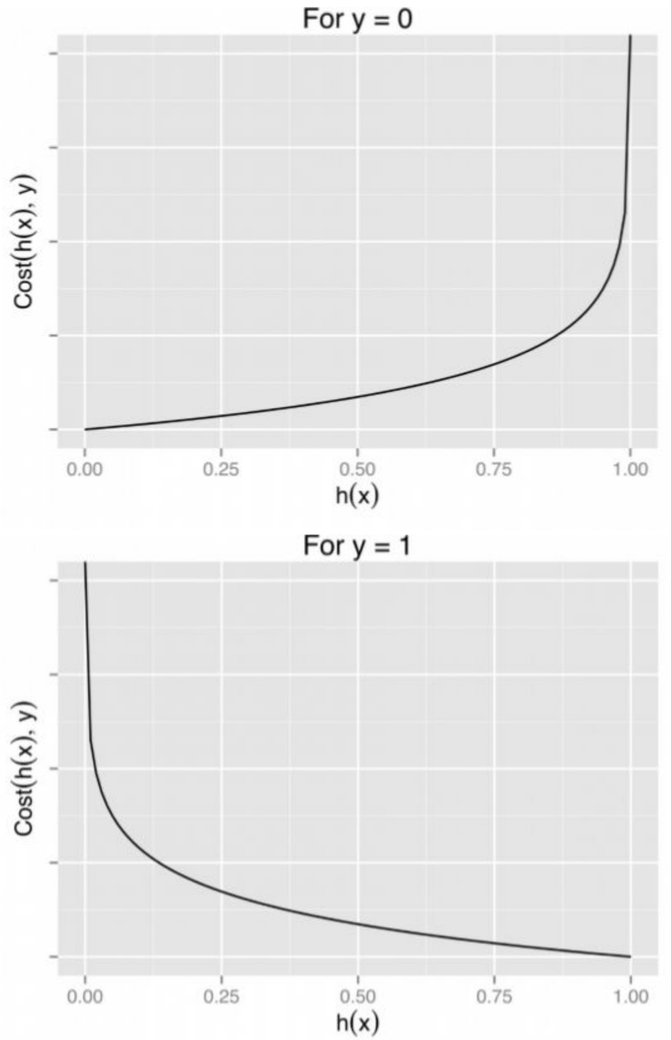




Cost function:

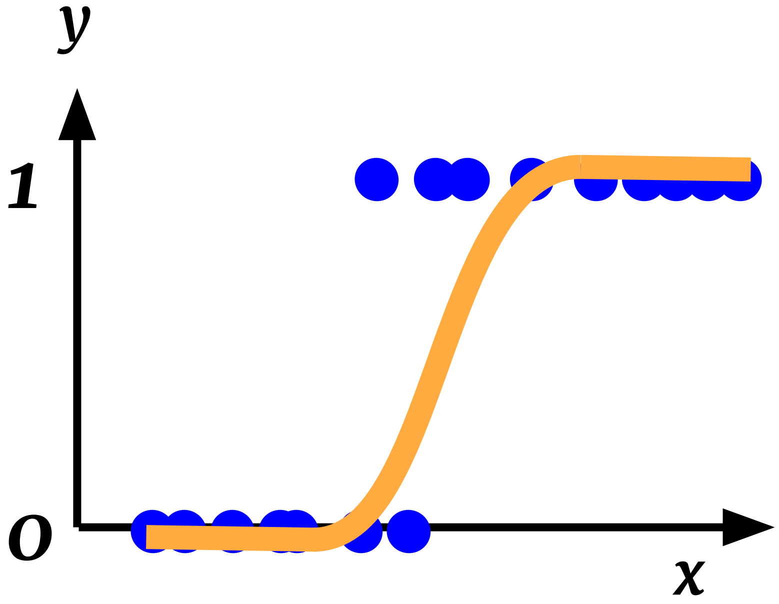
```
log_reg.fit(X_train, y_train)
```

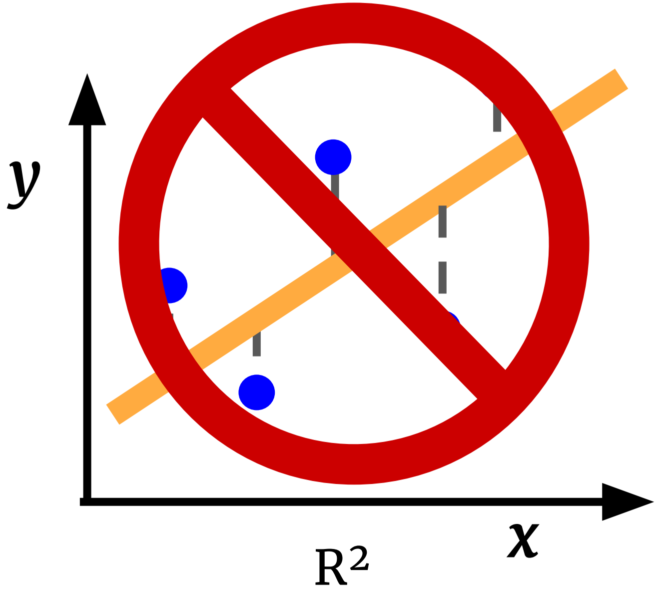




The score is also different: no more R^2

```
score = log_reg.score(X_test, y_test)
```





The score is also different: no more R^2

```
score = log_reg.score(X_test, y_test)
```

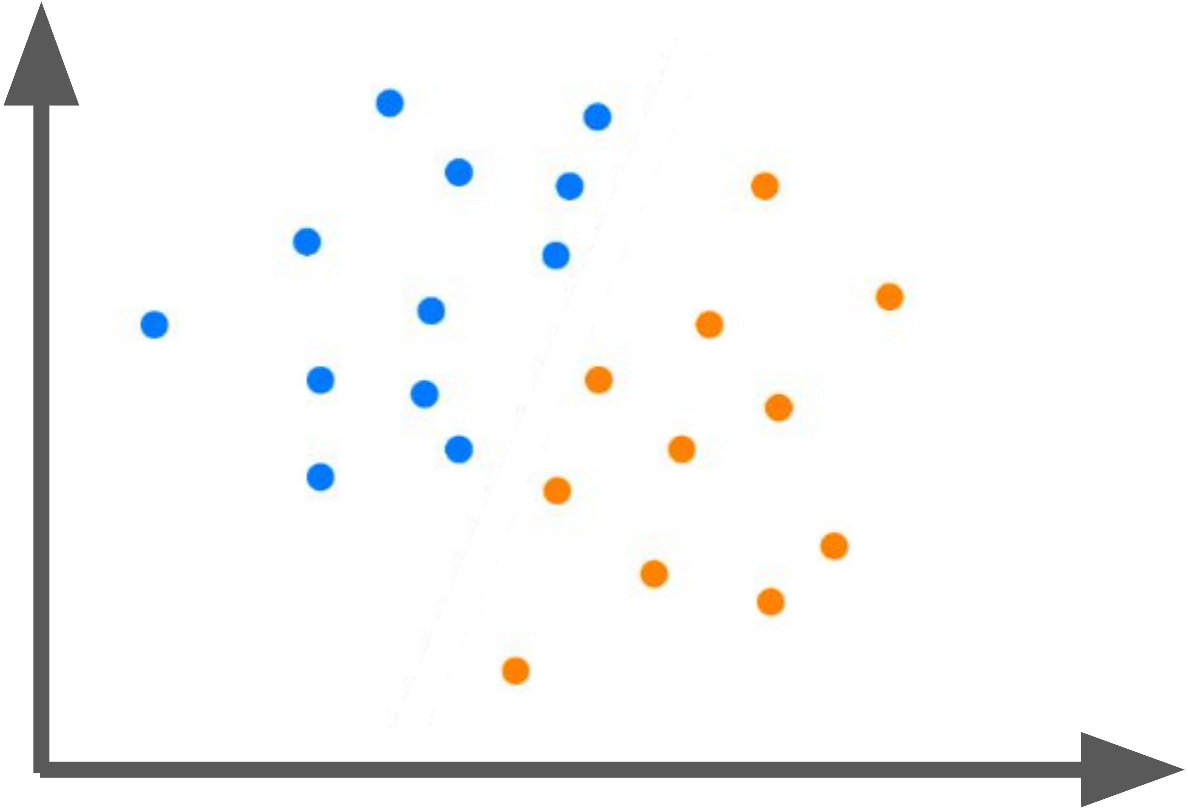
decision_function(X)	Predict confidence scores for samples.
densify()	Convert coefficient matrix to dense array format.
fit(X, y[, sample_weight])	Fit the model according to the given training data.
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict class labels for samples in X.
predict_log_proba(X)	Predict logarithm of probability estimates.
predict_proba(X)	Probability estimates.
score(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_params(**params)	Set the parameters of this estimator.
sparsify()	Convert coefficient matrix to sparse format.

	Actual Positives	Actual Negatives
Positive Predictions	True Positives (TP)	False Positives (FP)
Negative Predictions	False Negatives (FN)	True Negatives (TN)

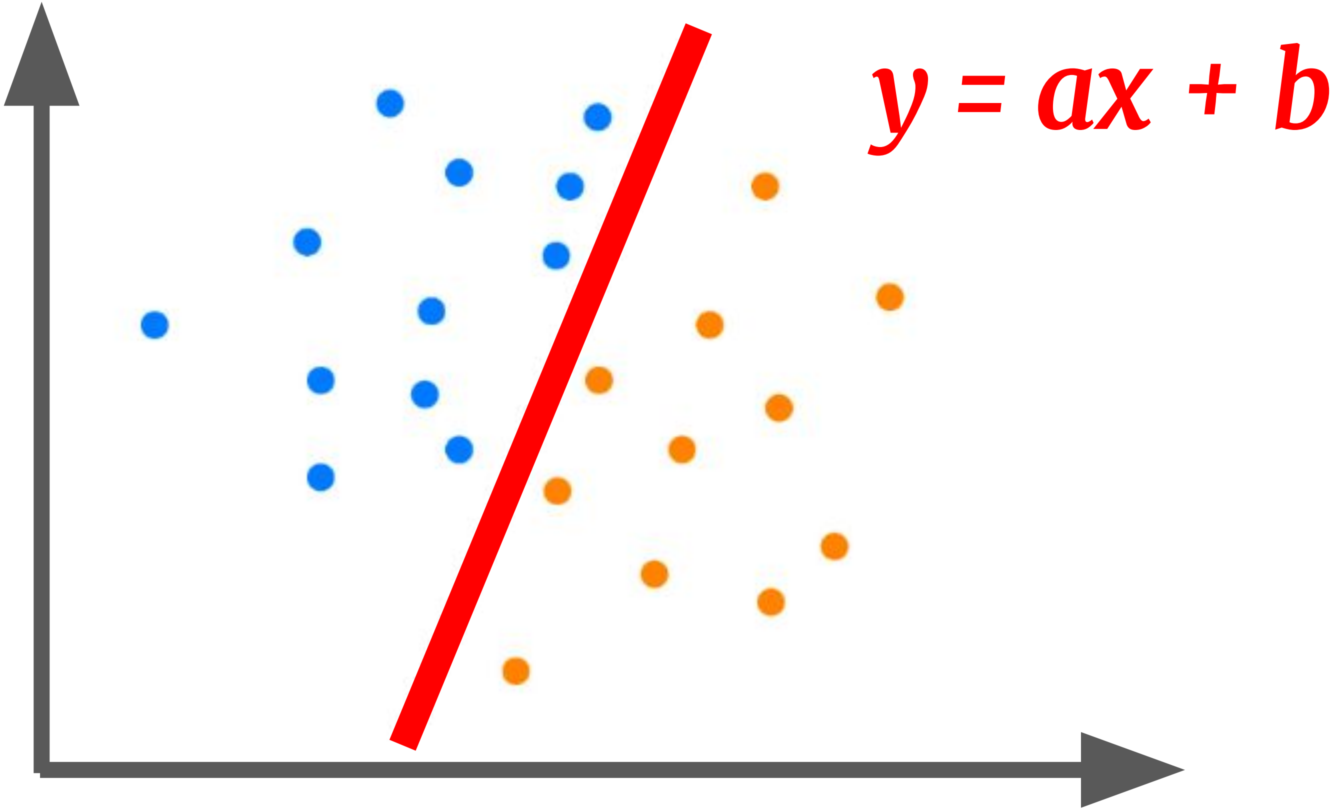
Here's our score:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

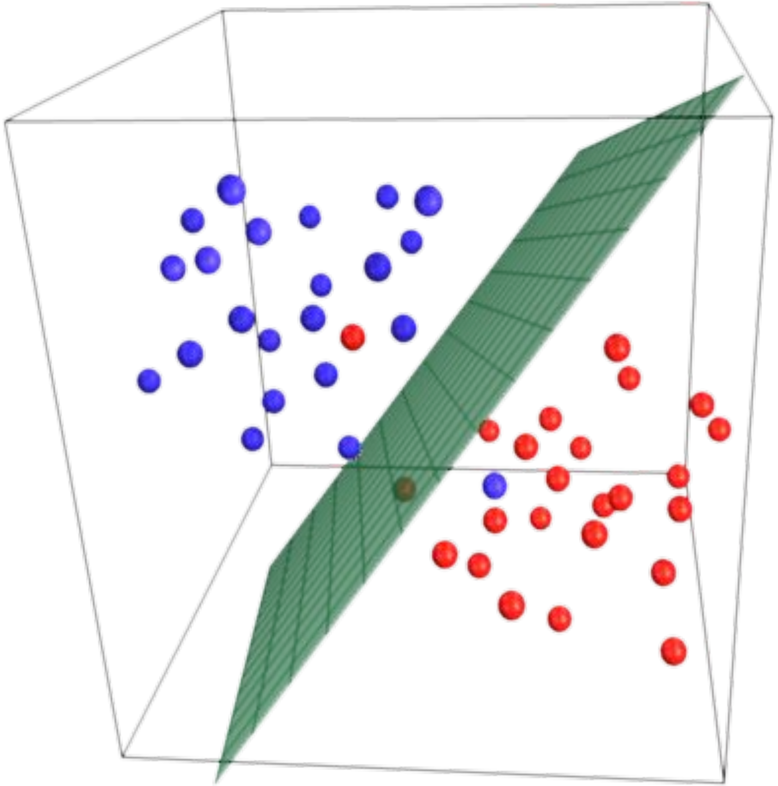
Second example: the Perceptron



The Perceptron



Perceptron in more dimensions

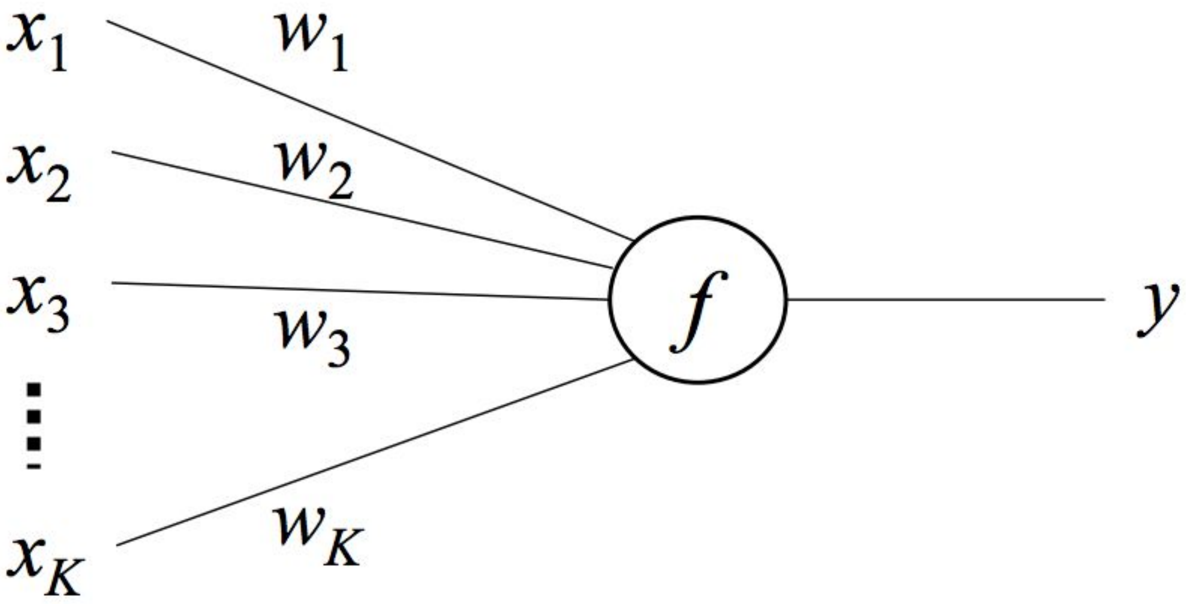


$$z = ax + by$$

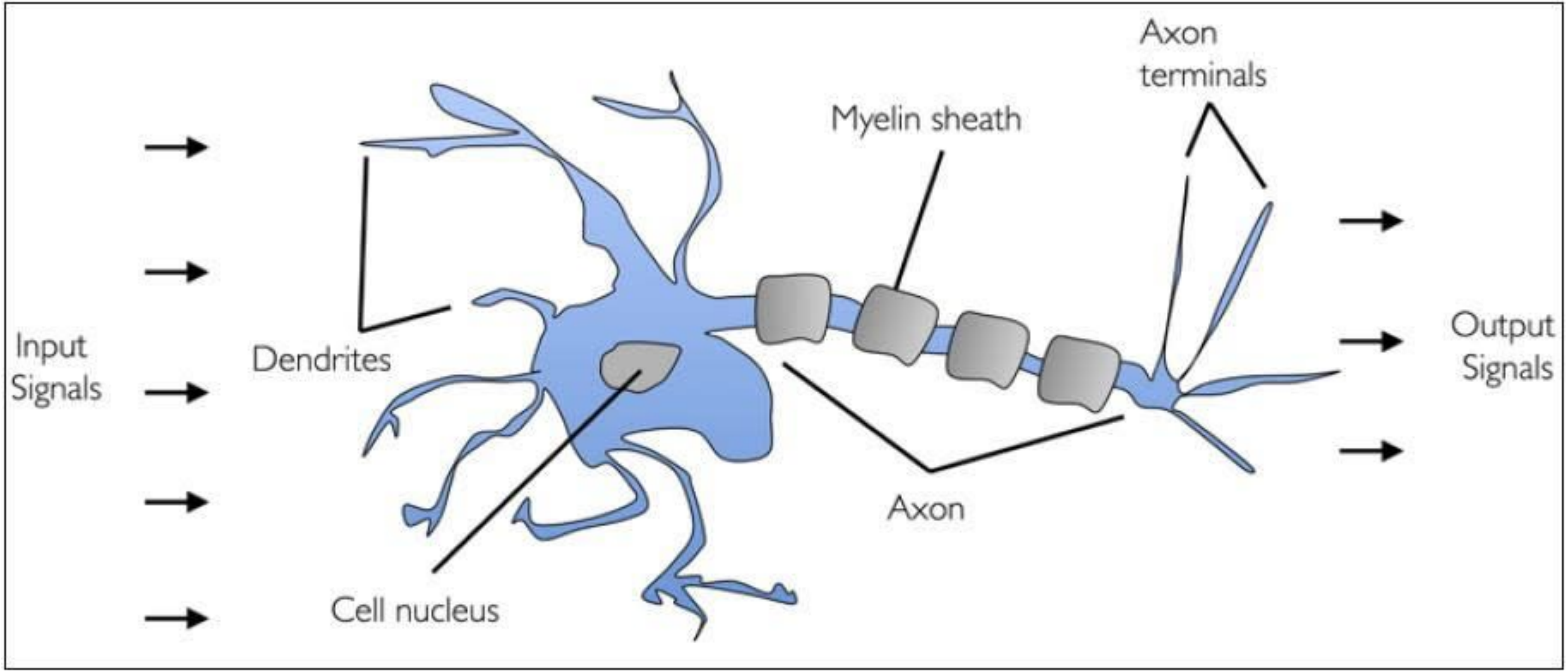
or

$$y = a_1x_1 + a_2x_2$$

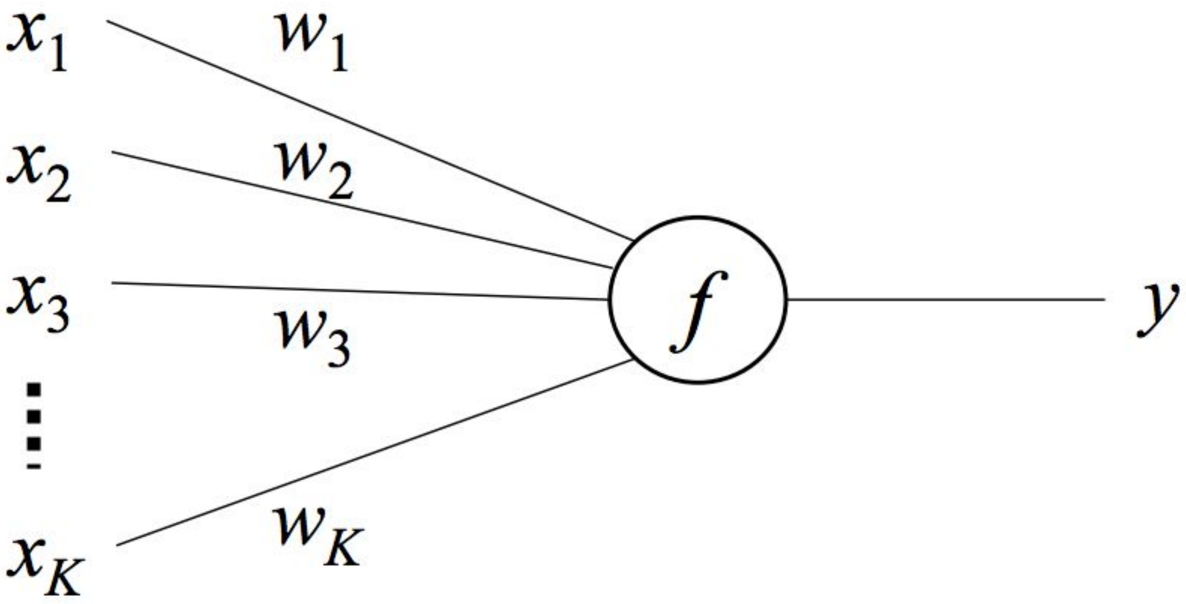
The Perceptron



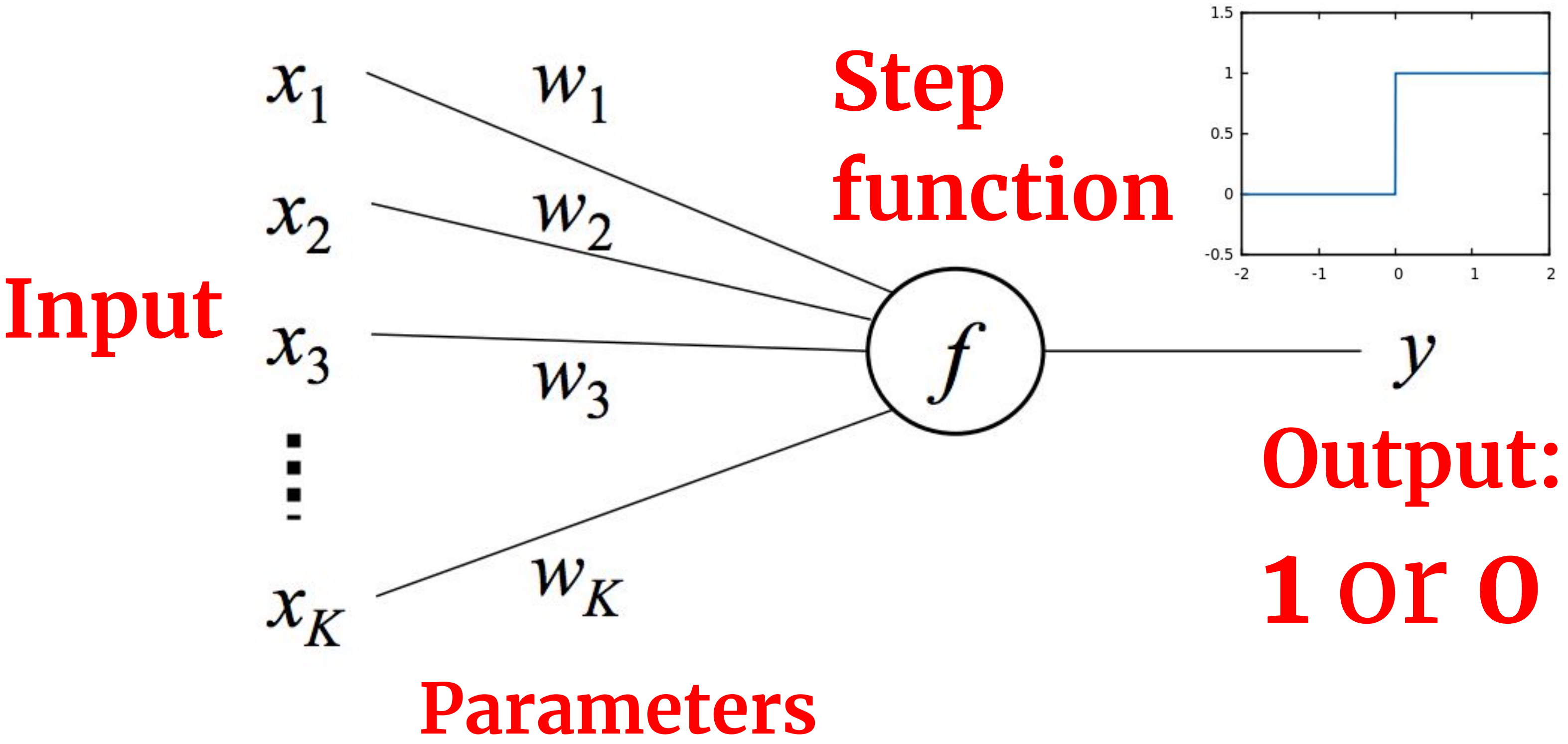
The Perceptron



The Perceptron



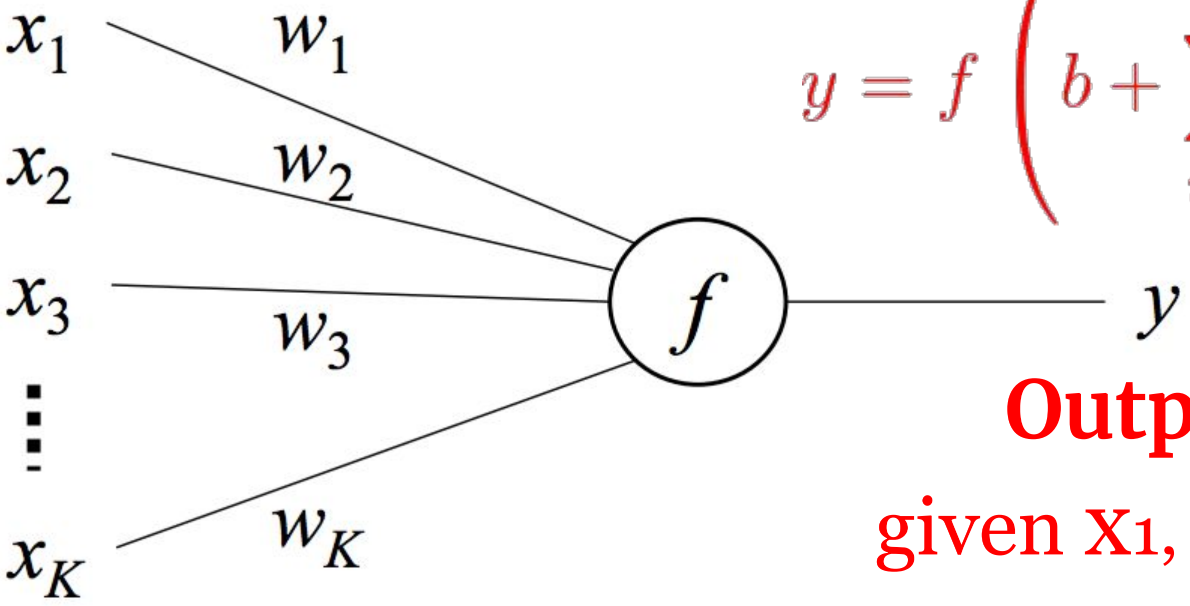
The Perceptron



The Perceptron

$$y = f\left(b + \sum_{i=1}^{K} w_i\, x_i\right)$$

Input

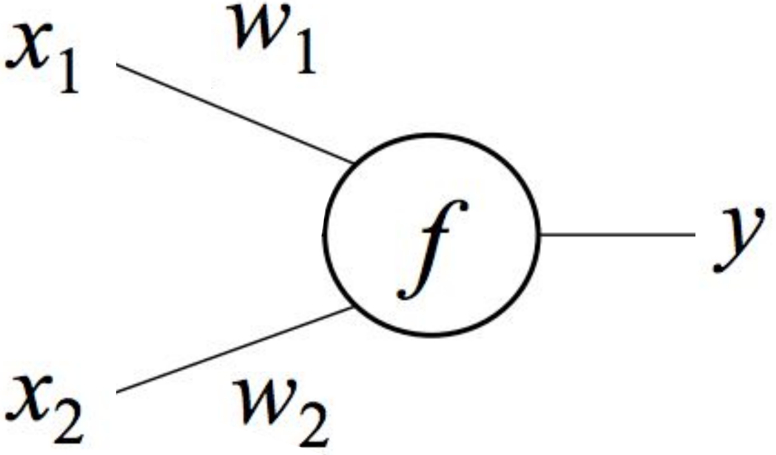


Output:

given $x_1, \ldots, x_K$,

$y=0$ or $y=1$

Training a perceptron



1. Initialise the vector ***w*** at 0.
2. Keep cycling through the data
3. For every ***x***, try classifying it:

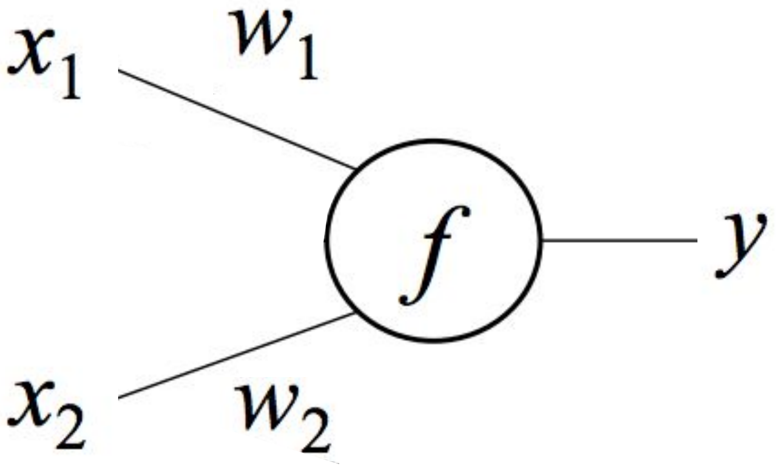
   $y^{pred} = f(w \cdot x + b)$

4. Update ***w***:

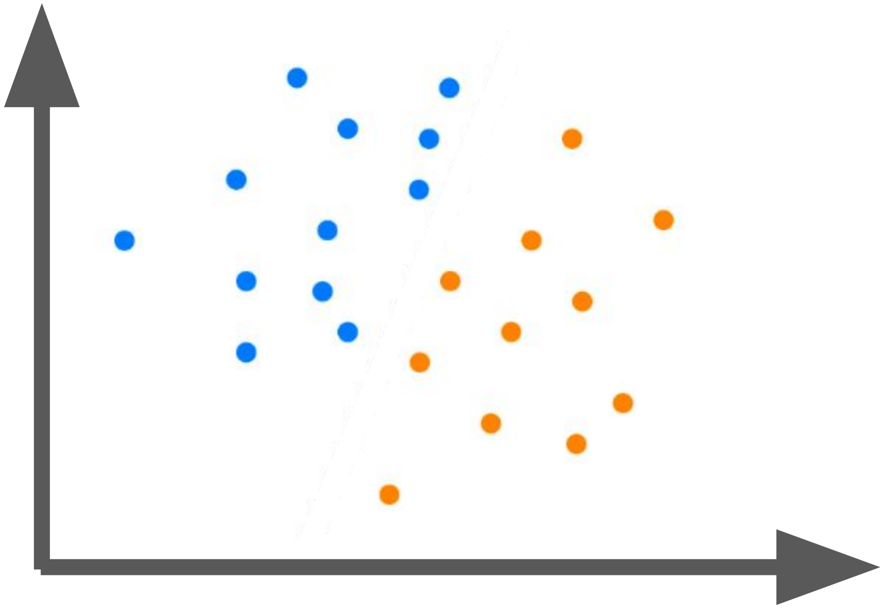
   $w_{t+1} = w_t + \alpha(y^{data} - y^{pred})x$

5. If the prediction is correct, ***w*** is not updated.

Training a perceptron



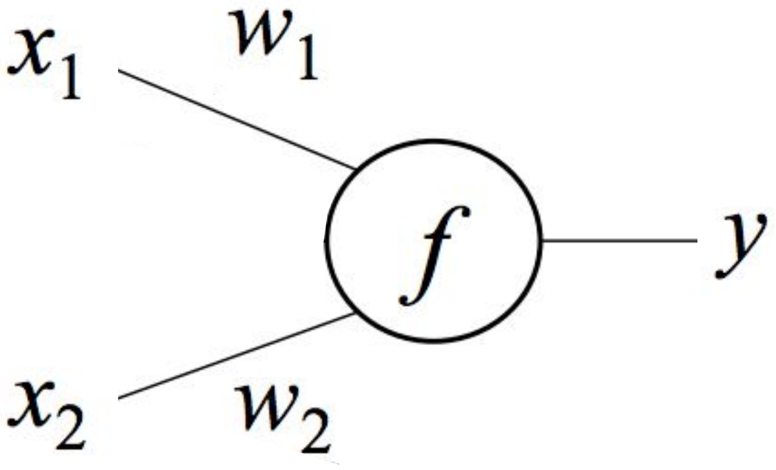
$(x_1, x_2) = (x, y)$

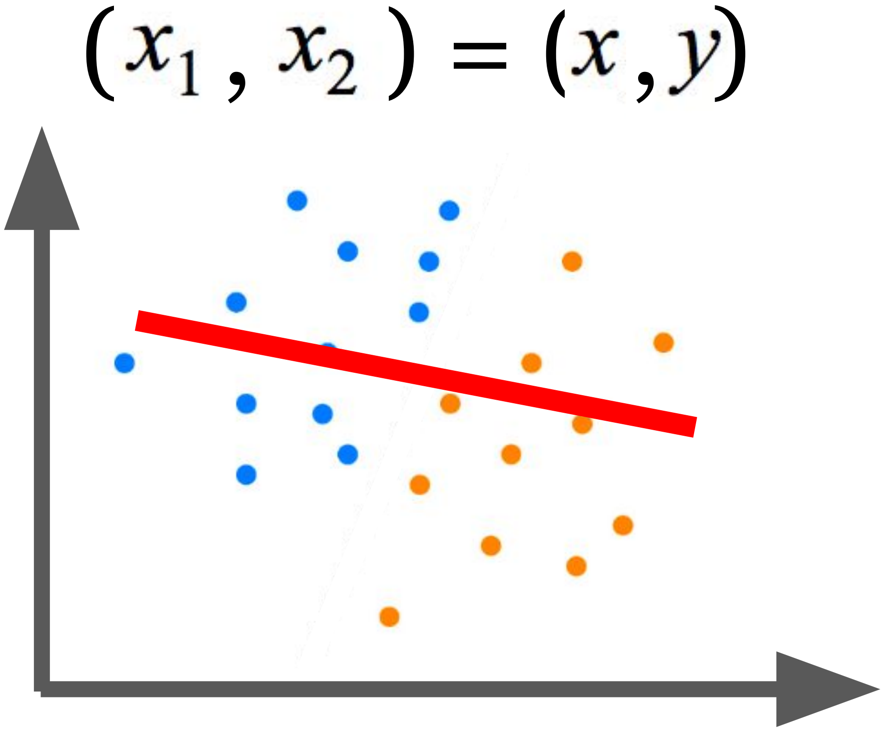


Learning the $\mathbf{w_i}$ weights

= learning the coefficients of the line

Training a perceptron

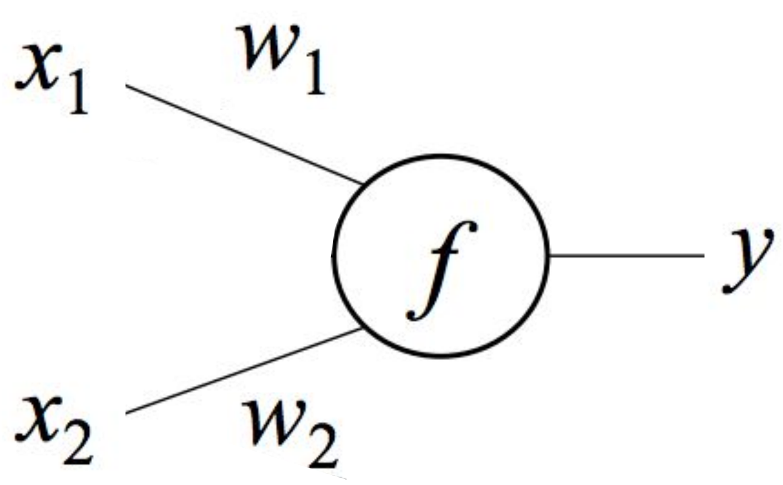




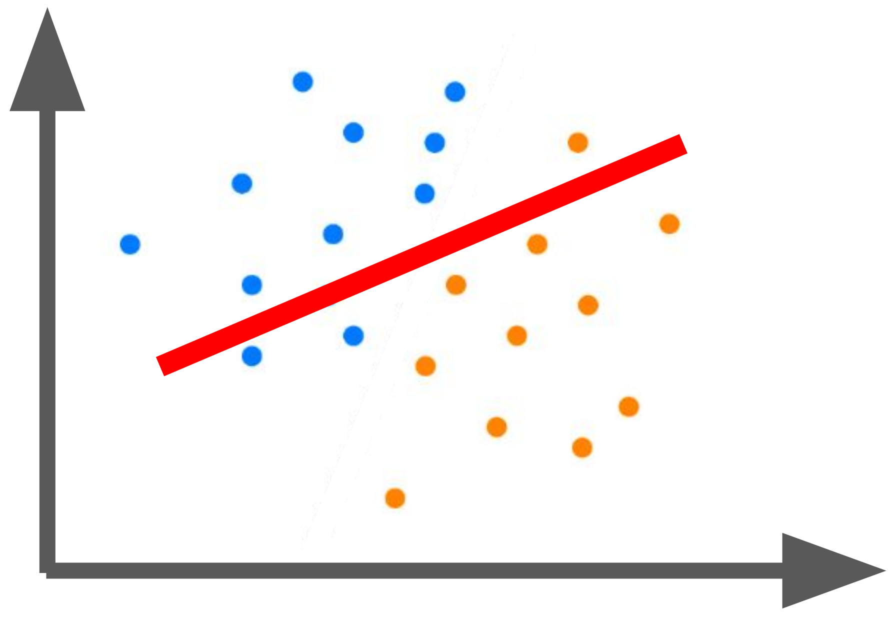
Learning the **w**ᵢ weights

= learning the coefficients of the line

Training a perceptron



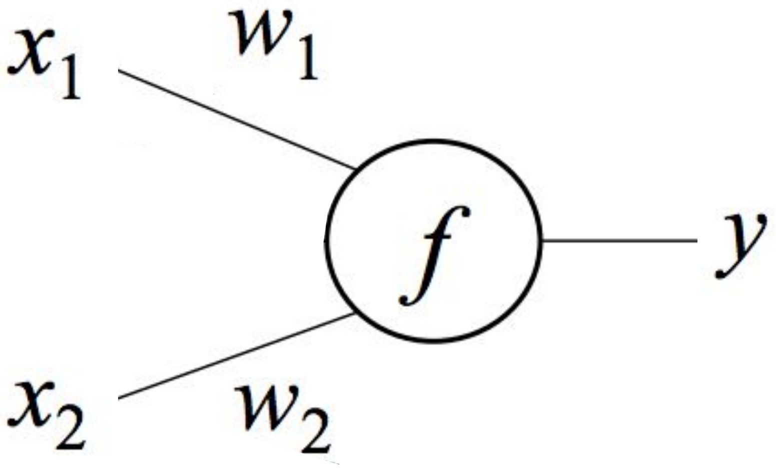
$(x_1, x_2) = (x, y)$

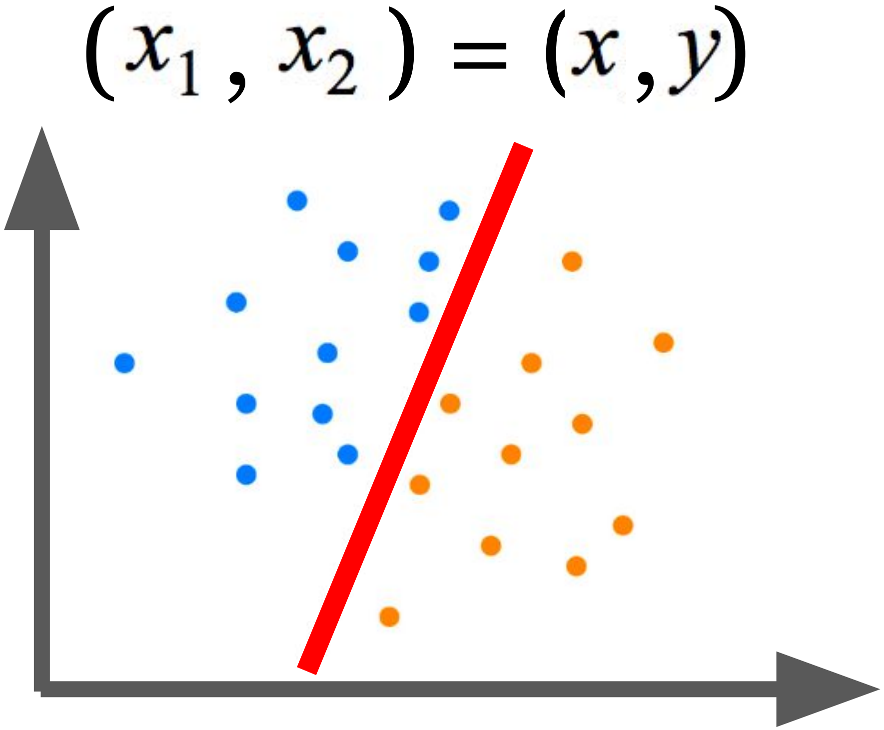


Learning the $\mathbf{w_i}$ weights

= learning the coefficients of the line

Training a perceptron

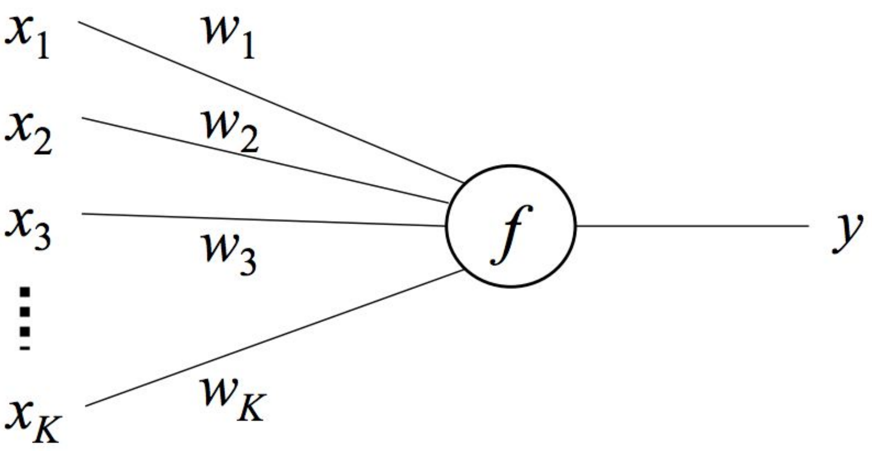




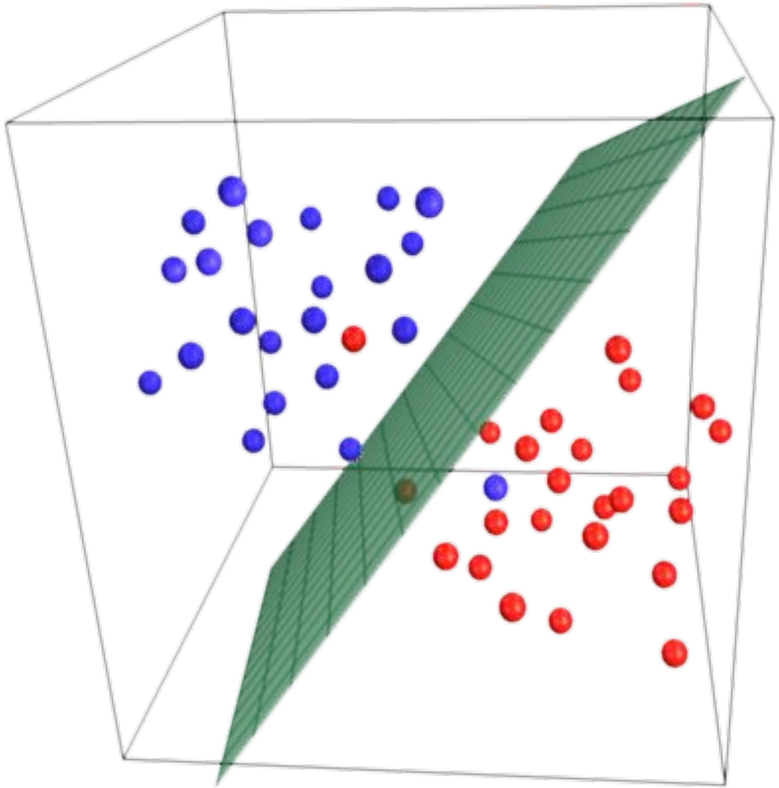
Learning the $\mathbf{w}_i$ weights

= learning the coefficients of the line

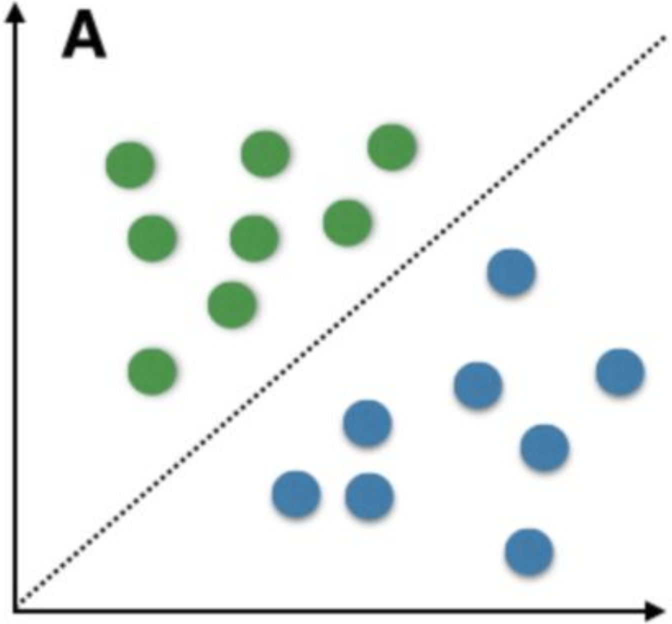
Perceptron in more dimensions



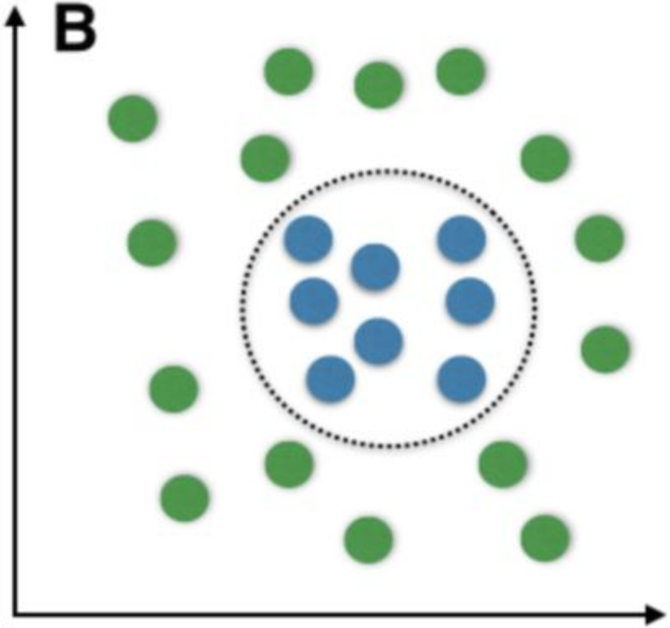
$$y = f\left(b + \sum_{i=1}^{K} w_i \, x_i\right)$$



The weakness of perceptrons: linear separability

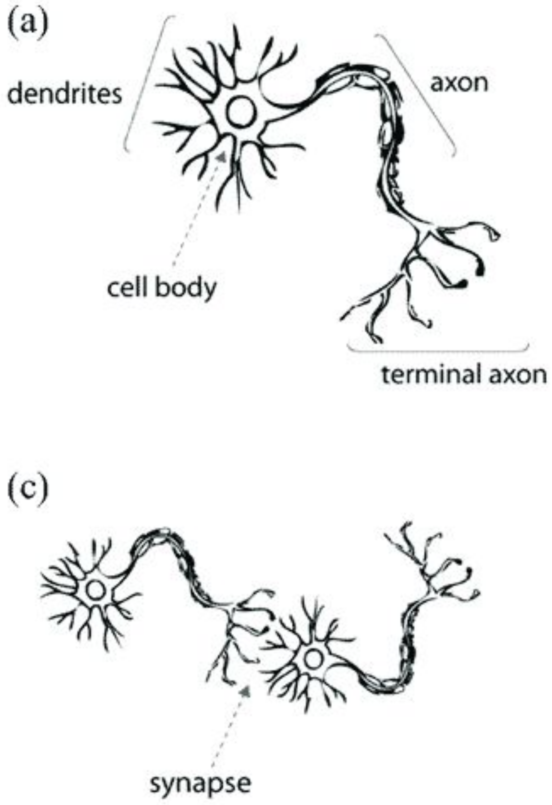


Perceptron can do

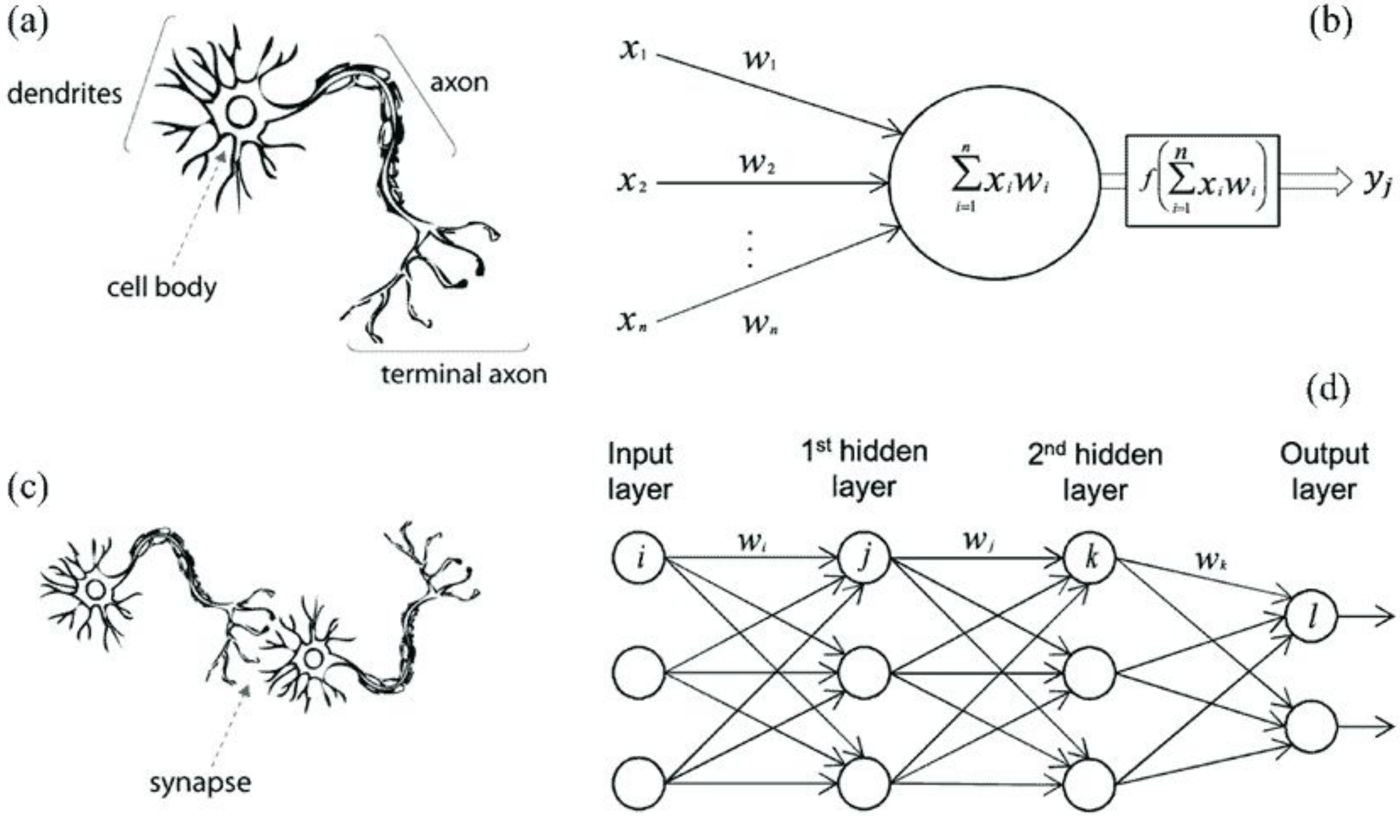


Perceptron can’t do

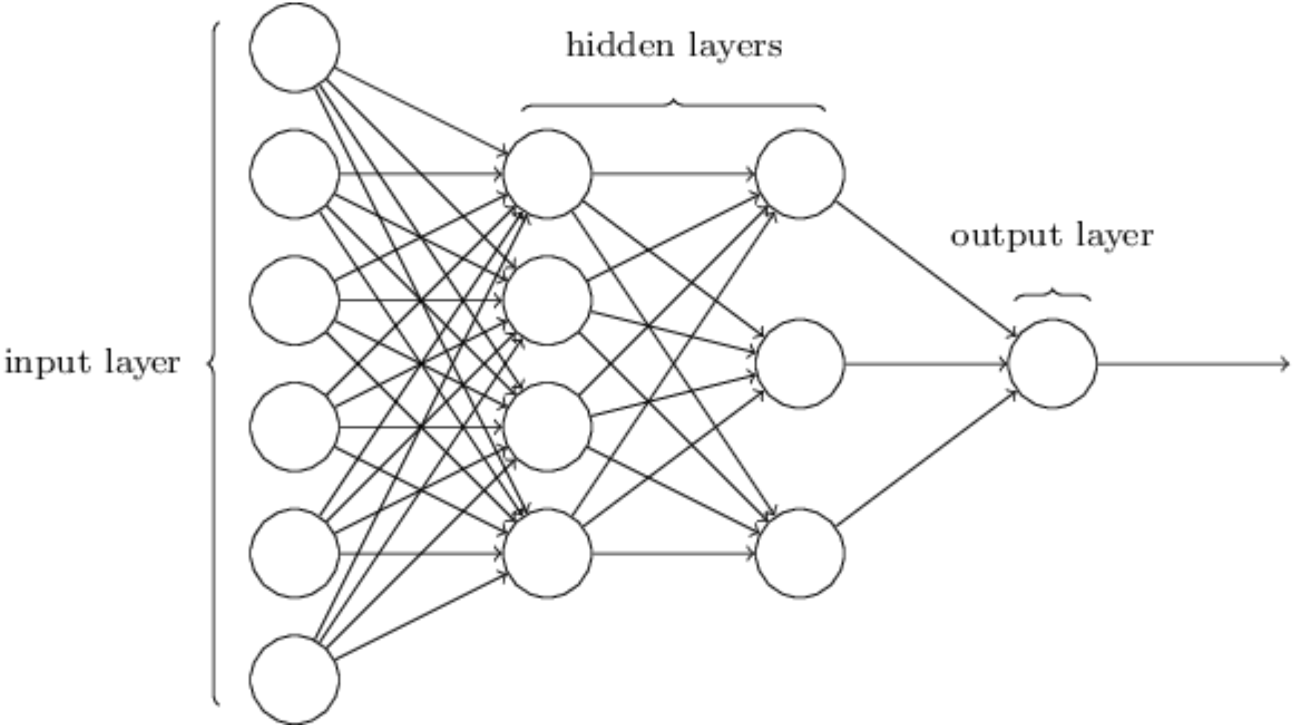
The solution: Multi Layer Perceptron



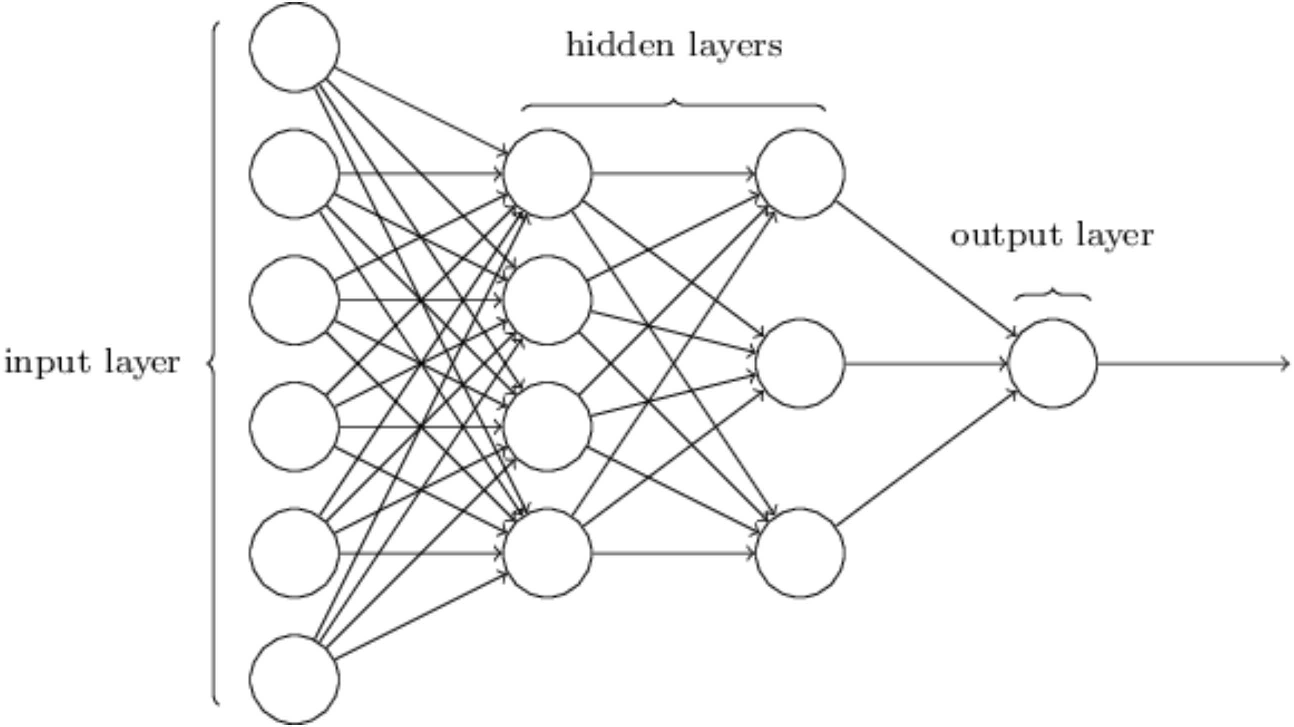
The solution: Multi Layer Perceptron



The MLP is the simplest neural network.

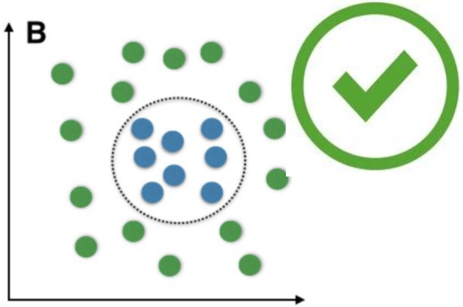


The MLP is the simplest neural network.

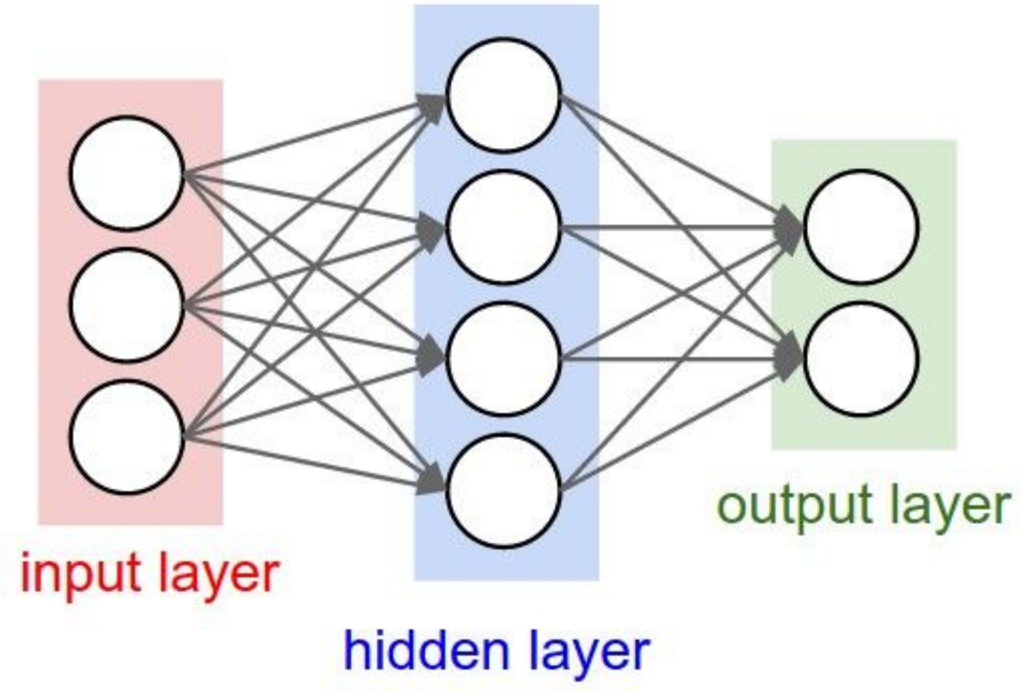


every neuron:

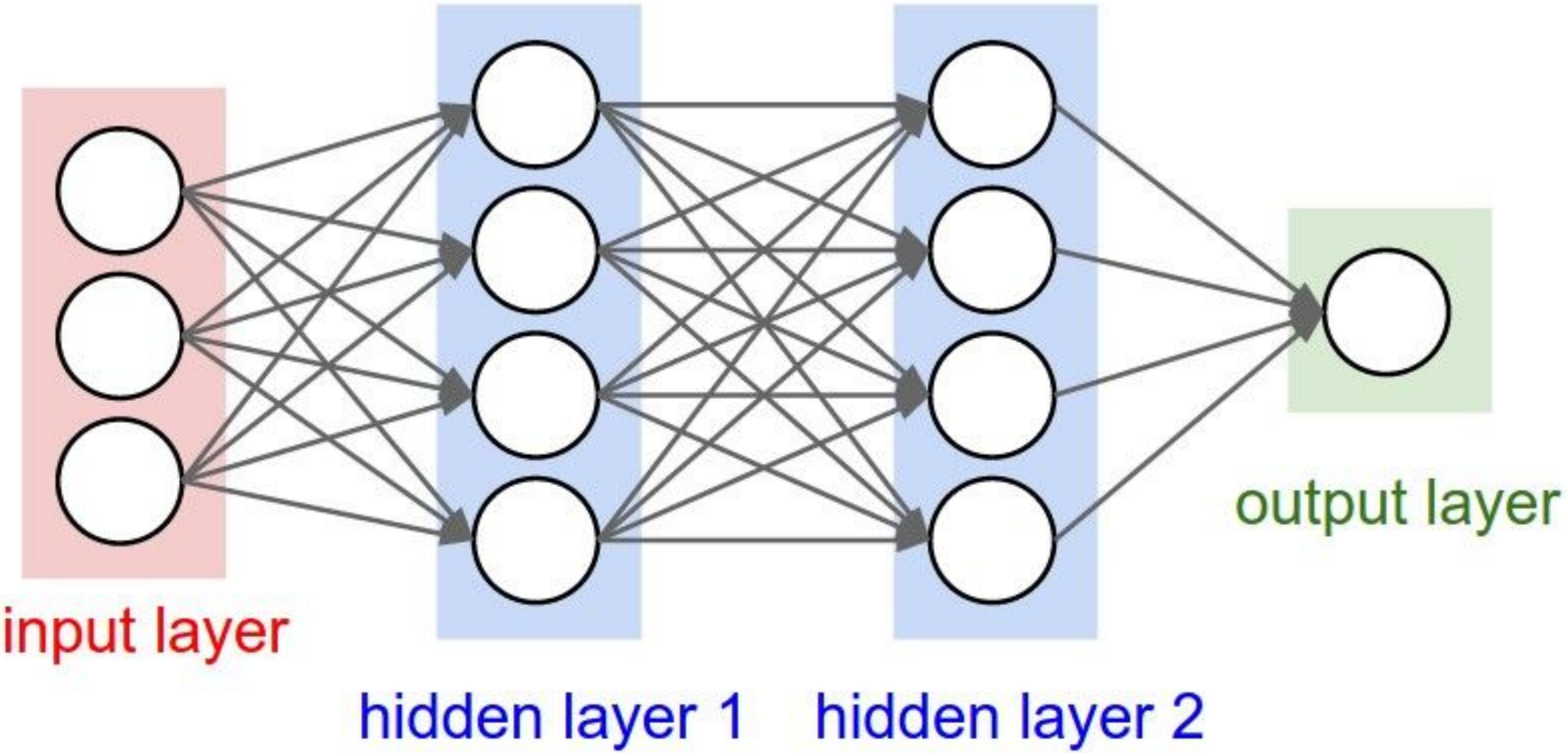
$$y = f\left(b + \sum_{i=1}^{K} w_i\, x_i\right)$$

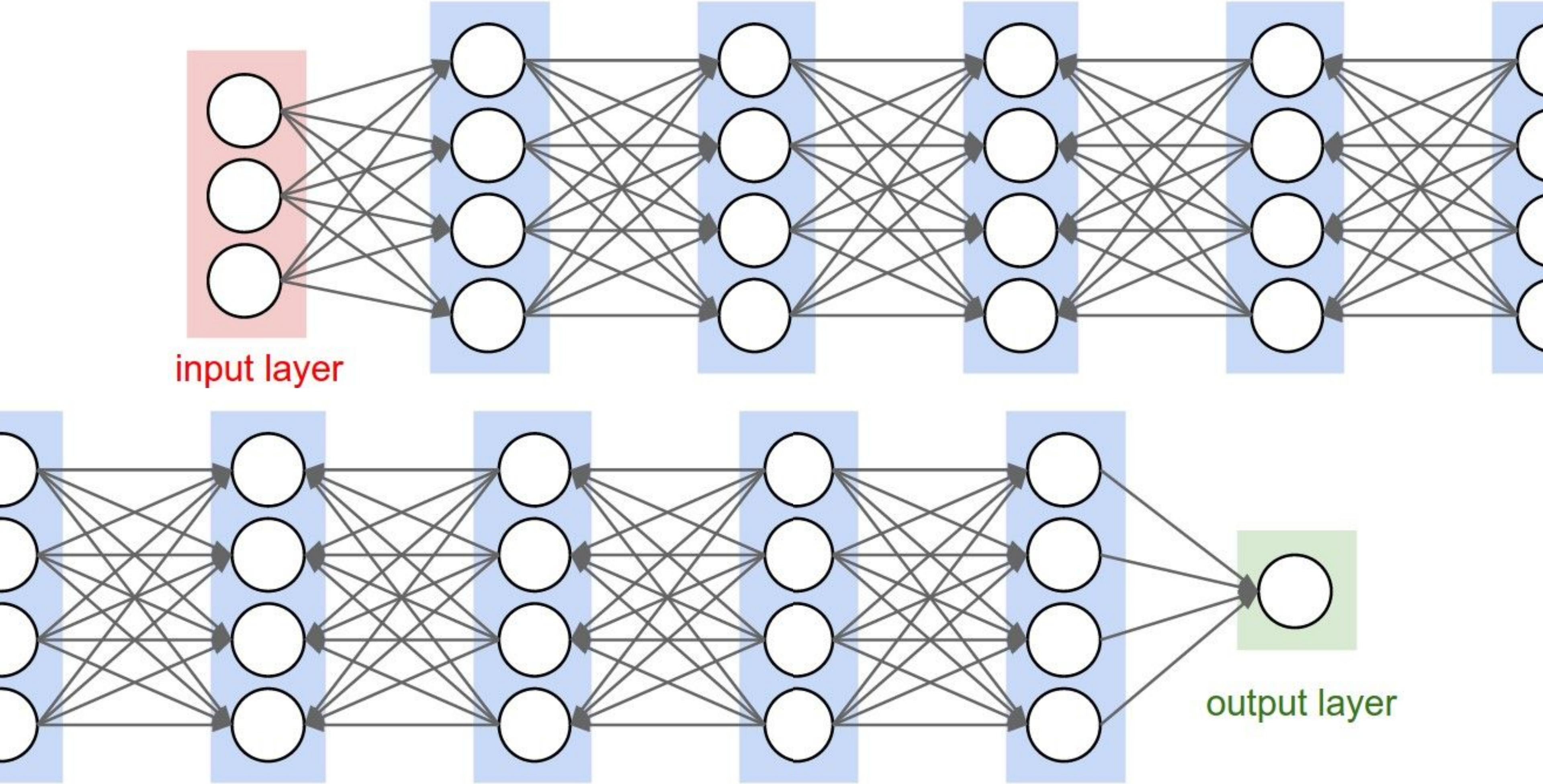


Neural networks



Neural networks




input layer
output layer

A mostly complete chart of

Neural Networks