Université de Montréal
Département d'informatique et de recherche opérationnelle

IFT6390
Fondements de l'apprentissage machine

Learning terminology

Professor: Ioannis Mitliagkas

Learning from examples

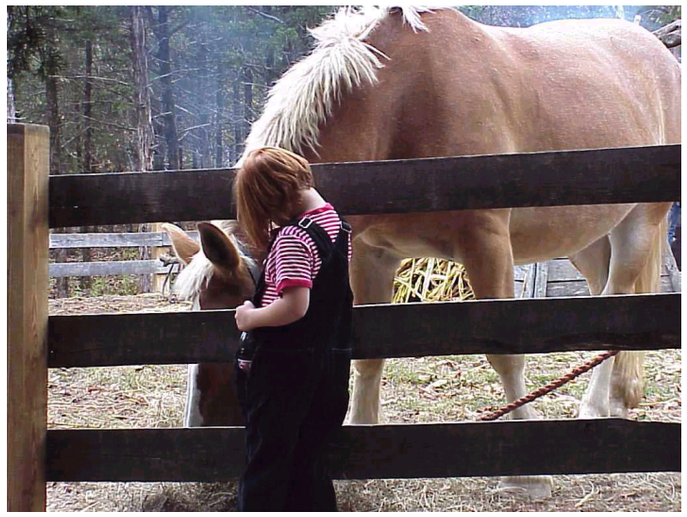
Learning from examples

Learning from examples

"horse"

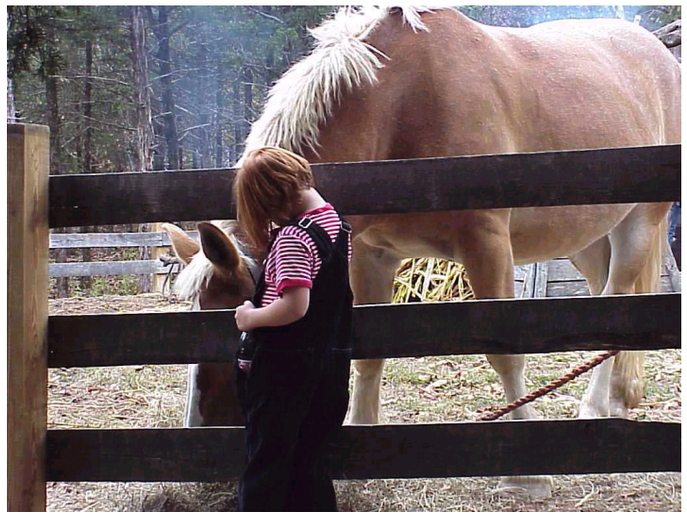
Learning from examples

"horse"



Learning from examples

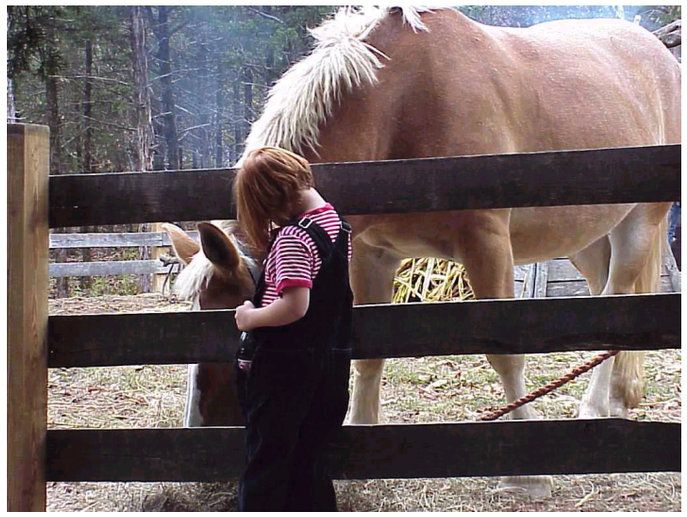
"horse"



"horse"

Learning from examples

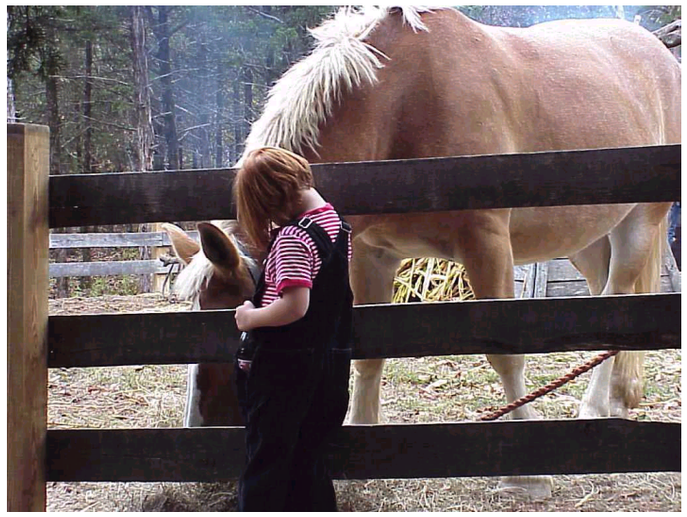
"horse"



"horse"

Learning from examples

"horse"

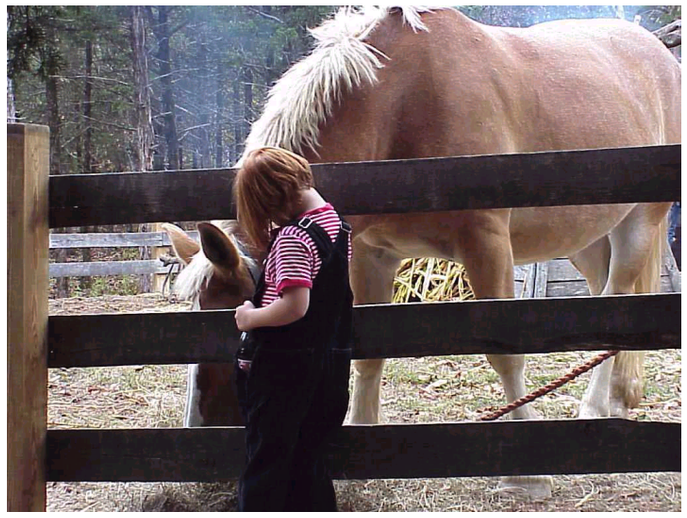


"horse"

"horse"

Learning from examples

"horse"



"horse"

"horse"

Principle much more general than to write by hand, starting from scratch, an algorithm to recognize a horse..

Categories of problems (tasks) in machine learning

Supervised learning

- Classification
- Regression

Non-supervised learning

- Density estimation (learn the distribution of examples)
- Clustering
- Dimensionality reduction
- Feature extraction

Reinforcement learning

(not scheduled for this class)

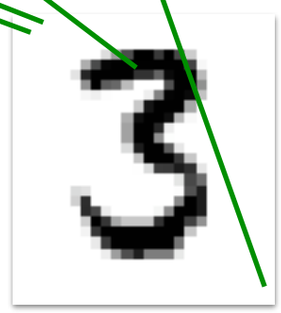
Example of classification problem

entry x_i

(vector representation)

$$x \in \mathbb{R}^d$$

$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$



Test point:
(new x)

Example of classification problem

training set

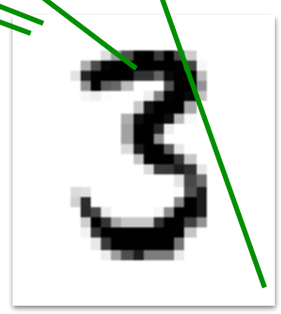
entry x_i

(vector representation)

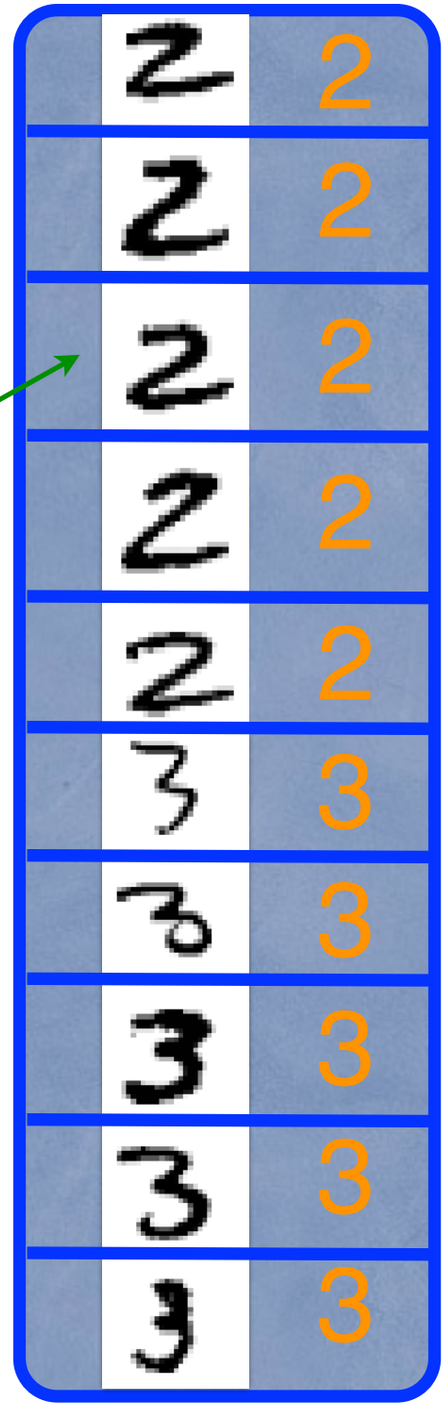
$$x \in \mathbb{R}^d$$

$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$

Test point:
(new x)



Example of classification problem



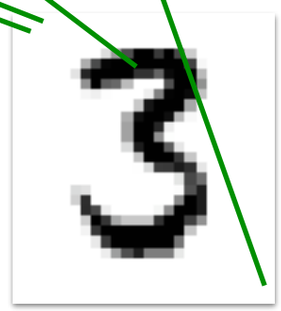
training set

entry x_i

(vector representation)

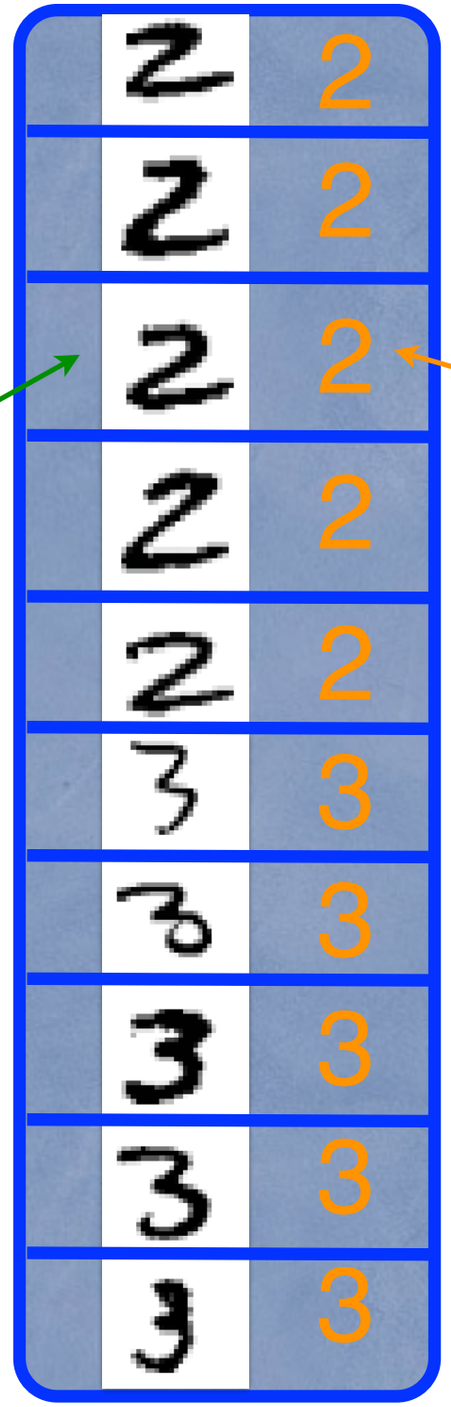
$$x \in \mathbb{R}^d$$

$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$



Test point:
(new x)

Example of classification problem



training set

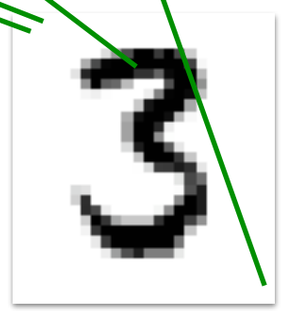
label y_i

entry x_i

(vector representation)

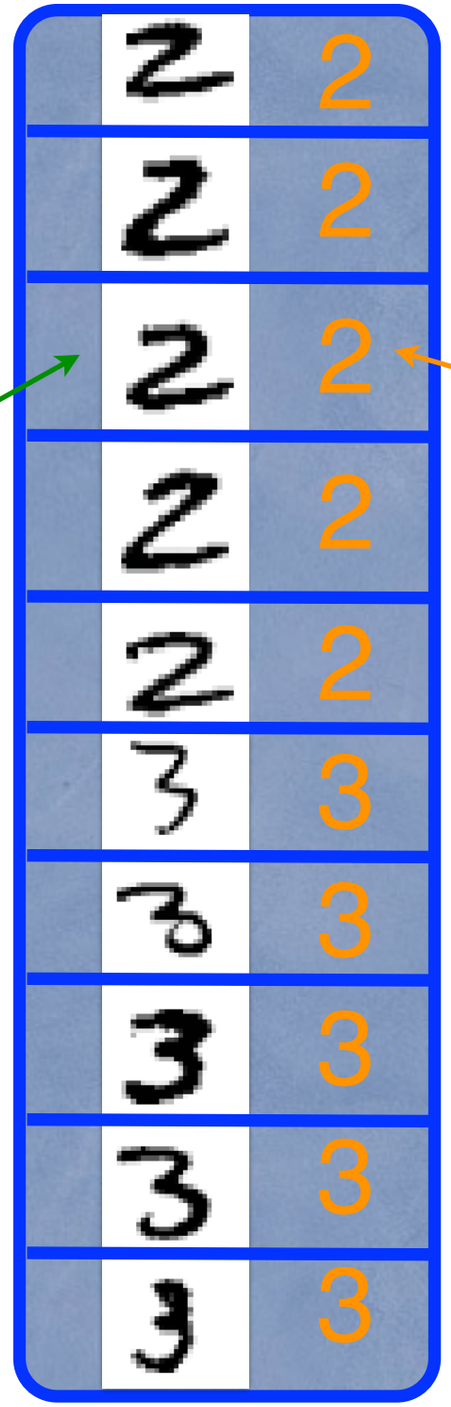
$$x \in \mathbb{R}^d$$

$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$



Test point:
(new x)

Example of classification problem



training set

entry x_i

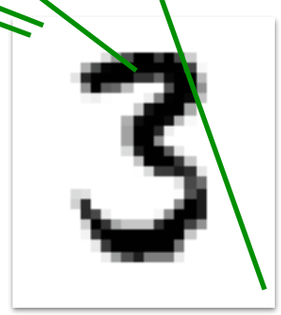
label y_i

(vector representation)

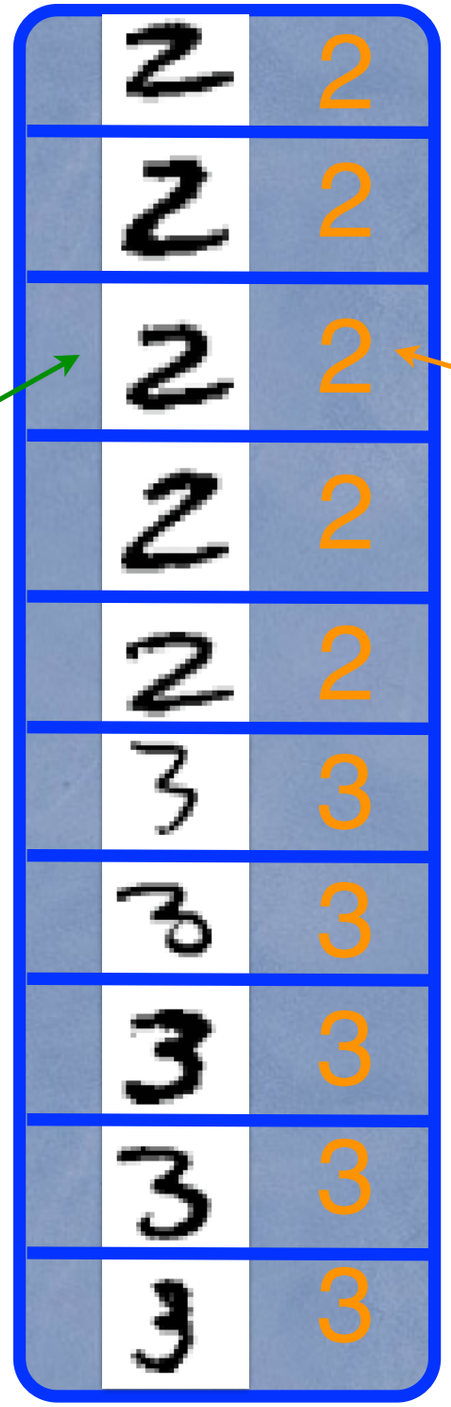
$$x \in \mathbb{R}^d$$

$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$

Test point:
(new x)



Example of classification problem



training set

entry x_i

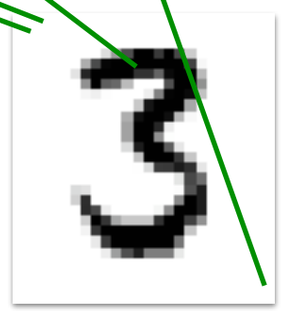
label y_i

(vector representation)

$$x \in \mathbb{R}^d$$

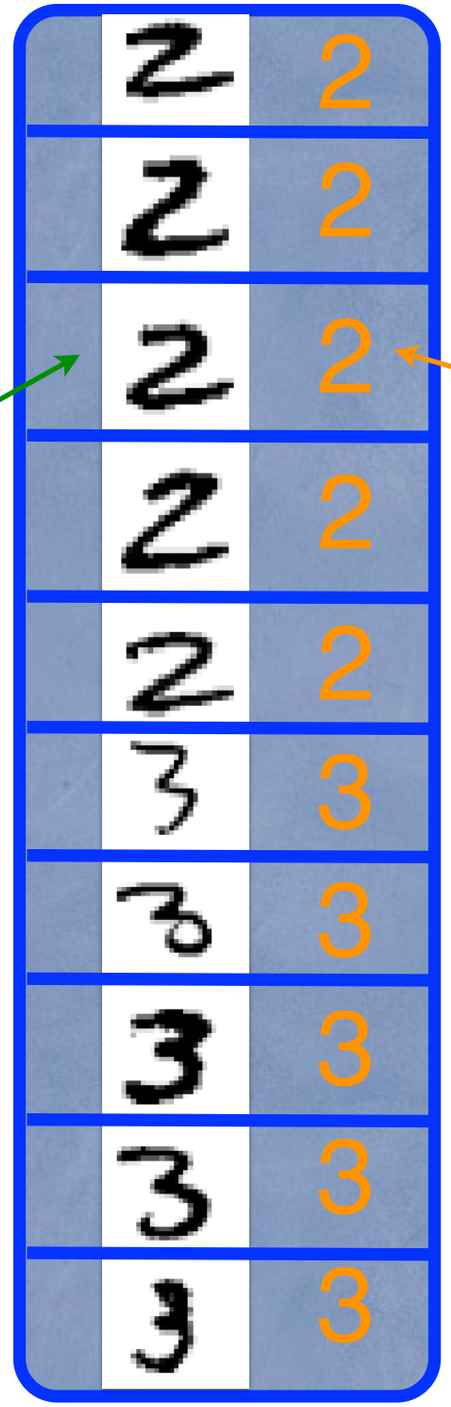
$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$

Test point:
(new x)



?

Example of classification problem



training set

entry x_i

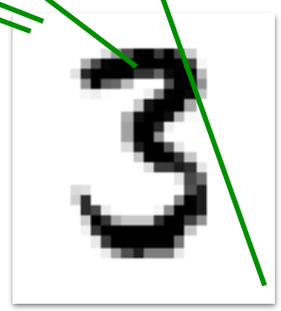
label y_i

(vector representation)

$$x \in \mathbb{R}^d$$

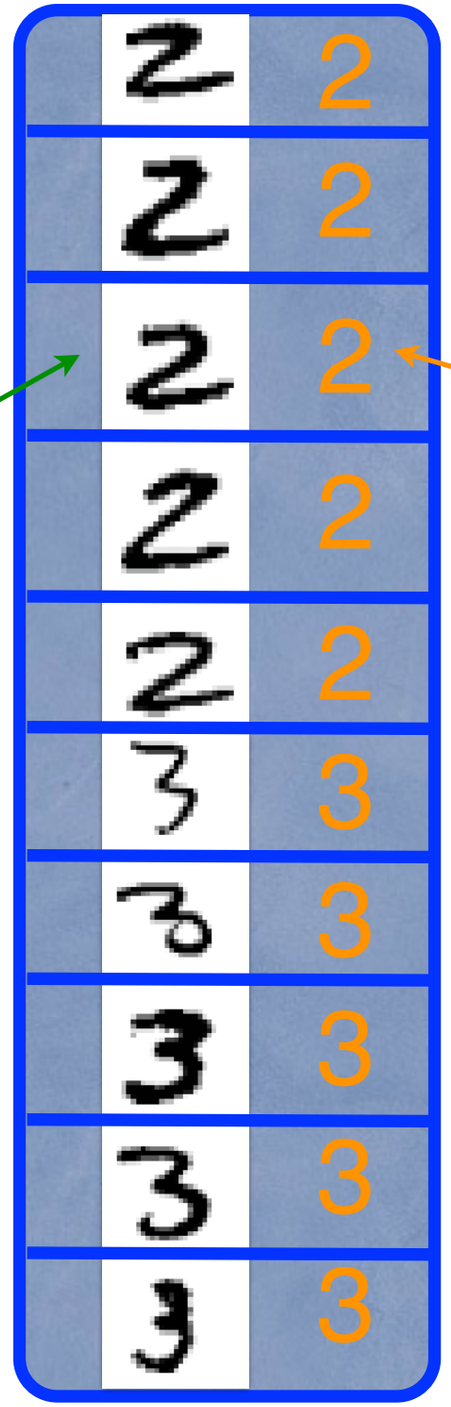
$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$

Test point:
(new x)



2 or 3?

Example of classification problem



training set

entry x_i

label y_i

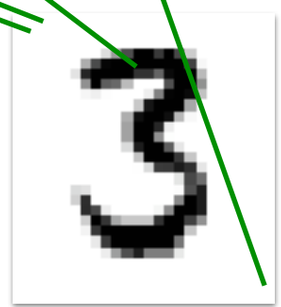
Learning is not memorizing

(vector representation)

$$x \in \mathbb{R}^d$$

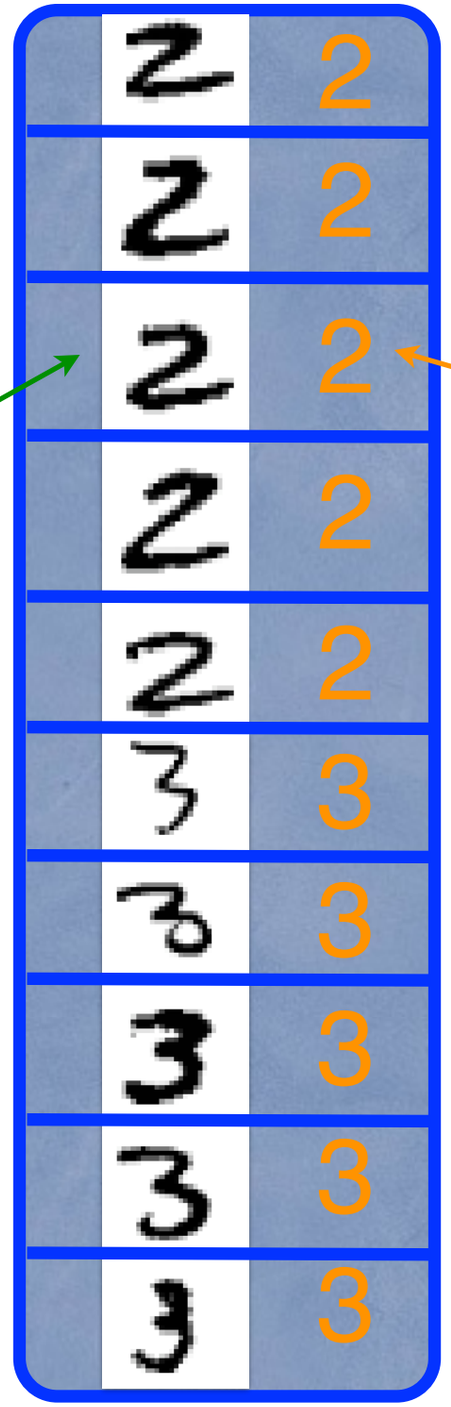
$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$

Test point:
(new x)



2 or 3?

Example of classification problem



training set

entry x_i

label y_i

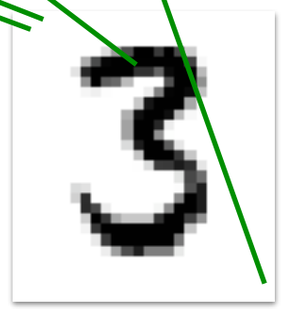
Learning is not memorizing

(vector representation)

$$x \in \mathbb{R}^d$$

$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$

Test point:
(new x)



2 or 3?

It is the ability to generalize to unseen cases!

Data representation

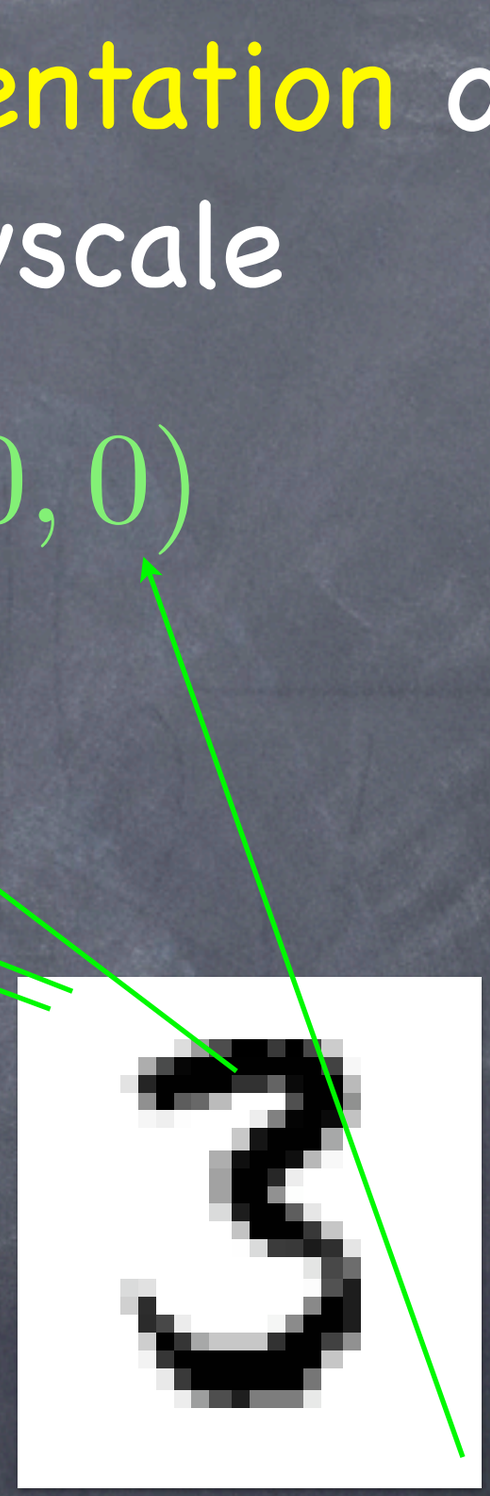
- Most learning algorithms require a **vector representation** of the examples
- Numerical vectors of fixed dimension **d**

$$x \in \mathbb{R}^d$$

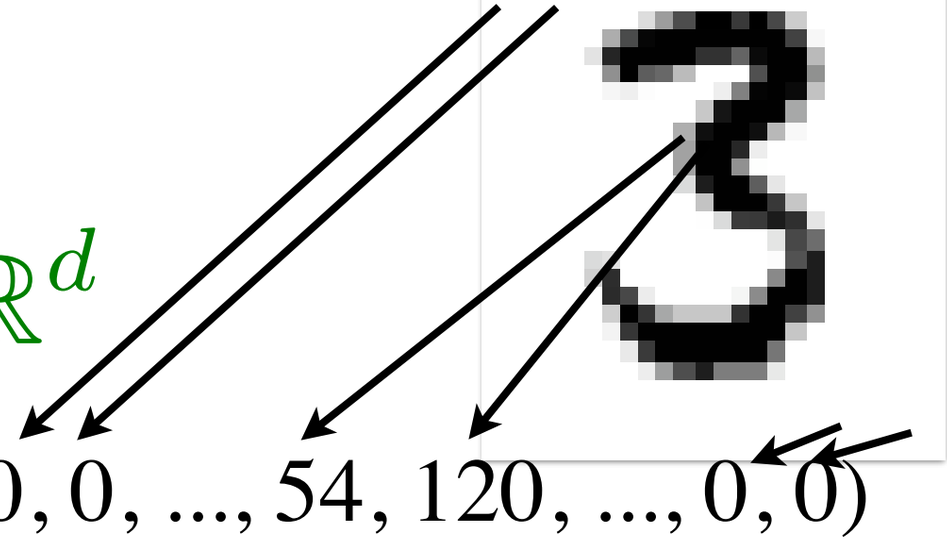
ex: $x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$

Example: raw vector representation of an image (bitmap) in grayscale

$$x = (0, 0, \ldots, 54, 120, \ldots, 0, 0)$$

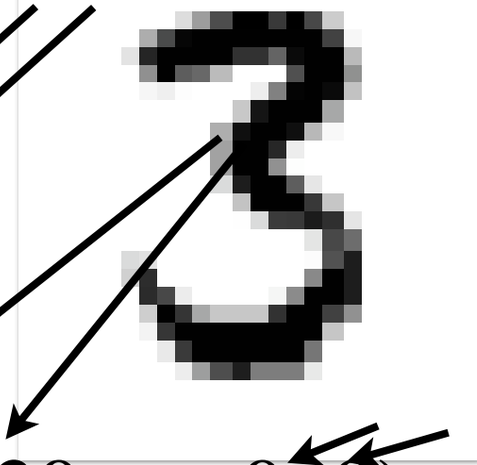


Transform an example into a vector representation $\mathbf{x} \in \mathbb{R}^d$



Raw representation:

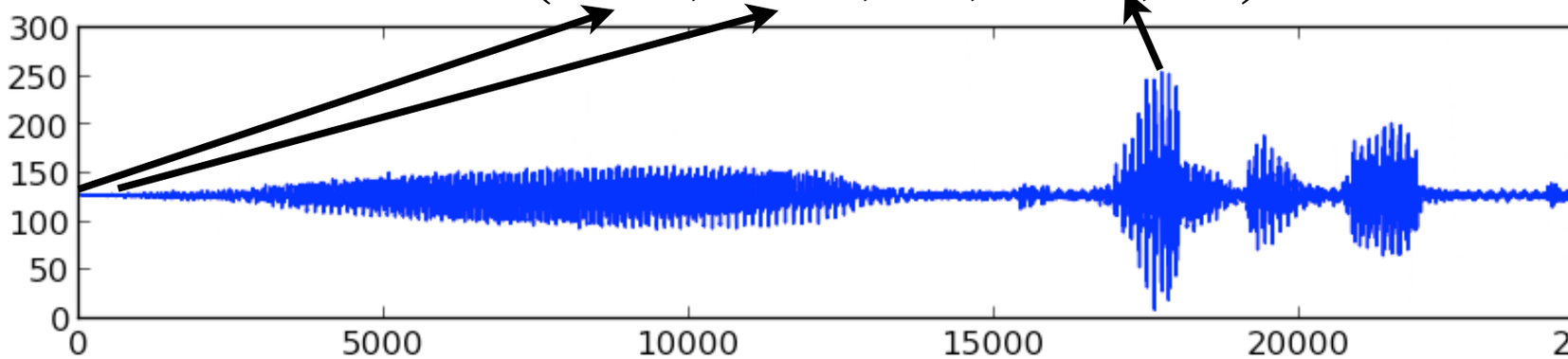
Transform an example into a vector representation $\mathbf{x} \in \mathbb{R}^d$



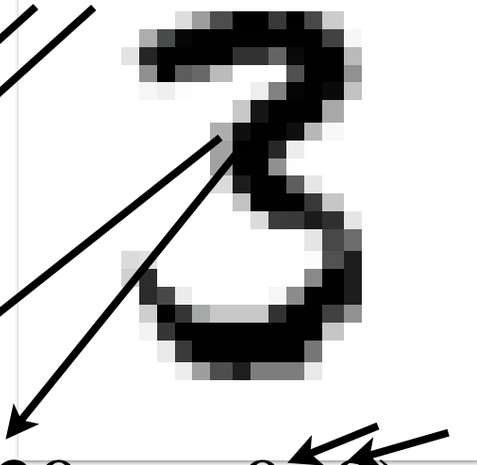
$x = (0, 0, ..., 54, 120, ..., 0, 0)$

Raw representation:

$x = (125, 125, ..., 250, ...)$



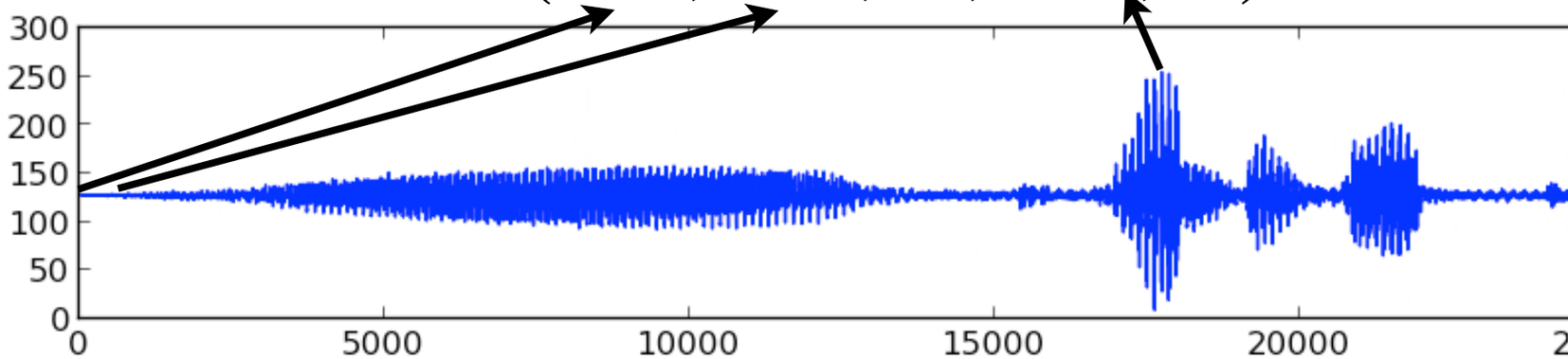
Transform an example into a vector representation $\mathbf{x} \in \mathbb{R}^d$



$x = (0, 0, ..., 54, 120, ..., 0, 0)$

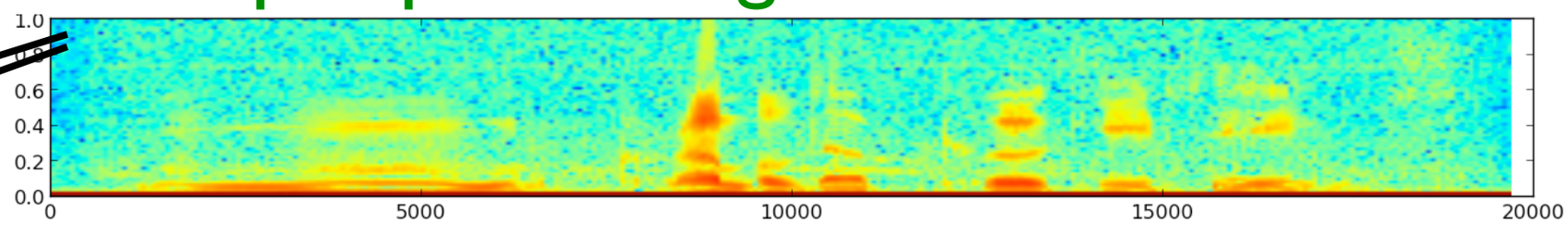
Raw representation:

$x = (125, 125, ..., 250, ...)$

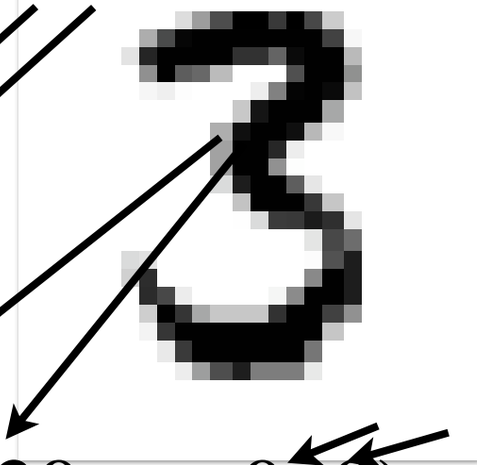


Or feature extraction from pre-processing:

$x = (, \quad , \quad , \quad , \quad ....)$



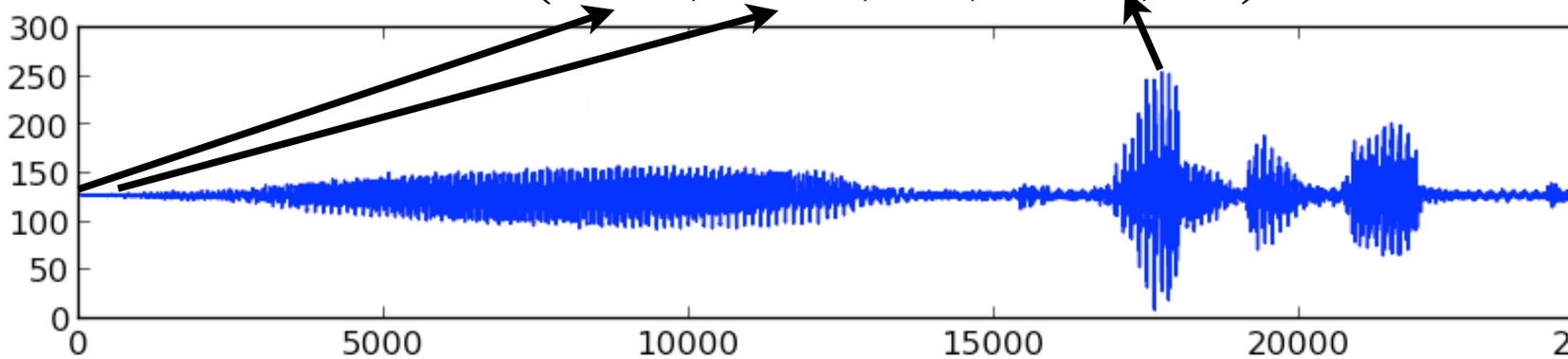
Transform an example into a vector representation $\mathbf{x} \in \mathbb{R}^d$



Raw representation:

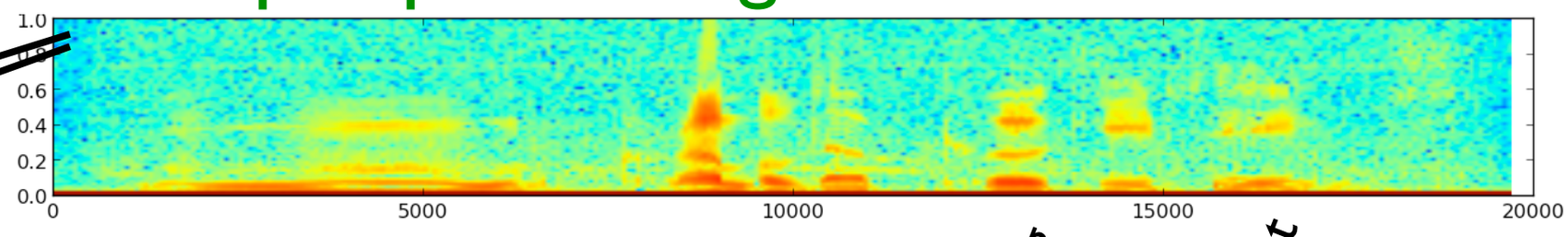
$x = (0, 0, ..., 54, 120, ..., 0, 0)$

$x = (125, 125, ..., 250, ...)$



Or feature extraction from pre-processing:

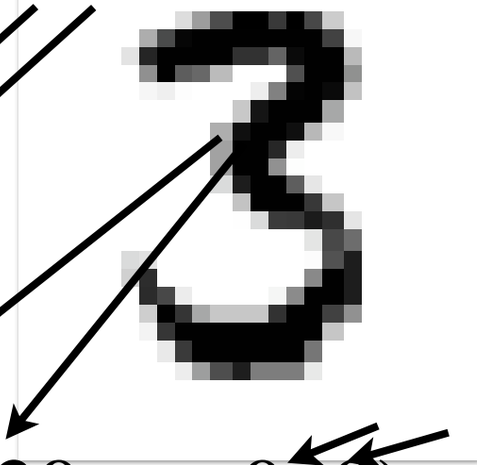
$x = (, \quad , \quad , \quad , \quad ....)$



Bag of words for «The cat jumped»: $x = (... 0 ... , 0, 1, ... 0 ... , 1, 0, 0, ..., 0, 0, 1, 0, ... 0 ..$

we, the, jumped, jumping, run, elephant, dog, cat, horse

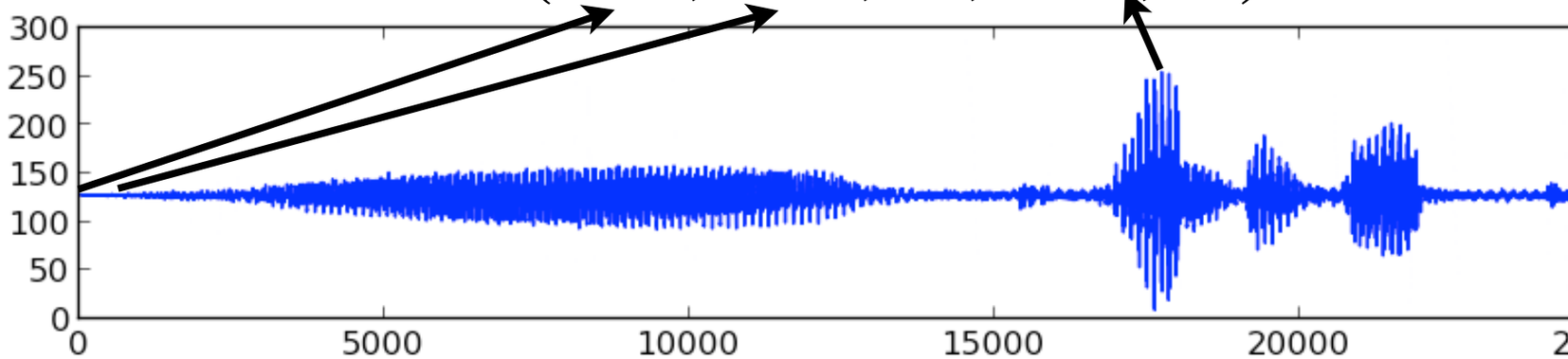
Transform an example into a vector representation $\mathbf{x} \in \mathbb{R}^d$



Raw representation:

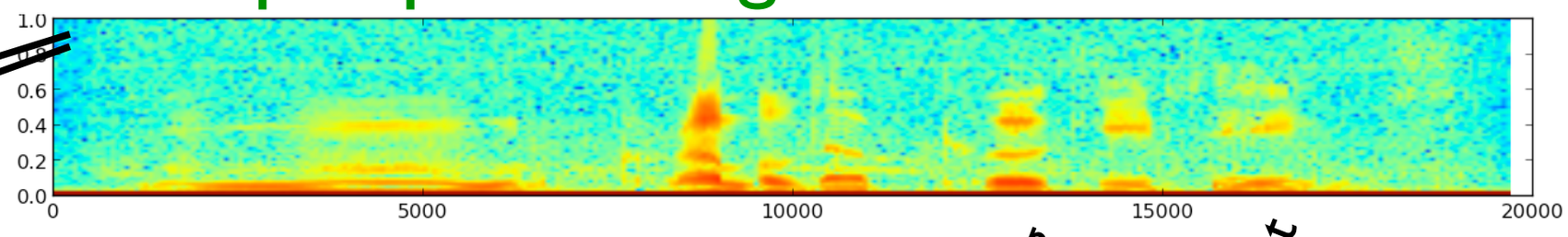
$x = (0, 0, ..., 54, 120, ..., 0, 0)$

$x = (125, 125, ..., 250, ...)$



Or feature extraction from pre-processing:

$x = (, \quad , \quad , \quad , \quad ....)$



Bag of words for «The cat jumped»: $x = (... 0 ... , 0, 1, ... 0 ... , 1, 0, 0, ..., 0, 0, 1, 0, ... 0 ..$

we the jumped jumping run elephant dog cat horse

OR vector of handmade features:

$x = (\text{feature } 1, \ \ldots \ , \text{feature } d)$

ex: Histograms of Oriented Gradients

Another possibility:

Characteristic traits

describe the inputs

(features, descriptors)

$x = (\text{feature 1}, \text{feature 2}, \ldots, \text{feature d})$

Another possibility:

Characteristic traits

describe the inputs

(features, descriptors)

$x = (\text{feature 1}, \text{feature 2}, \ldots, \text{feature d})$

- Often designed manually to facilitate the task of learning (feature engineering).
- eg. to represent a natural image (vision): SIFT descriptors.
- eg. to represent a document: frequency of occurrences of certain words (bag of words).

Another possibility:

Characteristic traits

describe the inputs

(features, descriptors)

$x = (\text{feature 1}, \text{feature 2}, \ldots, \text{feature d})$

- Often designed manually to facilitate the task of learning (feature engineering).
- eg. to represent a natural image (vision): SIFT descriptors.
- eg. to represent a document: frequency of occurrences of certain words (bag of words).

eg. feature k = how many times the word "plane" appears in the document.

Training set

Training set

input:

targets:

"horse"

"cat"

etc...

"horse"

Training set

number of examples

n

input:	targets:
	"horse"
	"cat"
etc...	
	"horse"

Training set

number of examples

n

input:

targets:

"horse"

"cat"

etc...

"horse"

preprocessing, feature extraction

Training set

number of examples n

input:

targets:

"horse"

"cat"

etc...

"horse"

preprocessing, feature extraction

inputs: X
(vector of features)

targets: Y

X_1 (3.5, -2, ... , 127, 0, ...) +1 Y_1

(-9.2, 32, ... , 24, 1, ...) -1

etc...

$X_{n,2}$

X_n (6.8, 54, ... , 17, -3, ...) +1 Y_n

Training set

number of examples n

input: | targets:

"horse"

"cat"

etc...

"horse"

preprocessing, feature extraction

Dimensionality of input d

inputs: X (*vector of features*) | targets: Y

X_1 (3.5, -2, ... , 127, 0, ...) +1 Y_1

(-9.2, 32, ... , 24, 1, ...) -1

etc...

$X_{n,2}$

X_n (6.8, 54, ... , 17, -3, ...) +1 Y_n

Training set

number of examples n

input: | targets:

"horse"

"cat"

etc...

"horse"

preprocessing, feature extraction

Dimensionality of input d

inputs: X (*vector of features*) | targets: Y

X_1 (3.5, -2, ... , 127, 0, ...) +1 Y_1

(-9.2, 32, ... , 24, 1, ...) -1

etc...

$X_{n,2}$

X_n (6.8, 54, ... , 17, -3, ...) +1 Y_n

Test point:

Training set

number of examples n

input:

targets:

"horse"

"cat"

etc...

"horse"

preprocessing, feature extraction

Dimensionality of input d

inputs: X (*vector of features*)

targets: Y

X_1 (3.5, -2, ... , 127, 0, ...) +1 Y_1

(-9.2, 32, ... , 24, 1, ...) -1

etc...

$X_{n,2}$

X_n (6.8, 54, ... , 17, -3, ...) +1 Y_n

Test point:

Training set

number of examples n

input: | targets:

"horse"

"cat"

etc...

"horse"

preprocessing, feature extraction

Dimensionality of input d

inputs: X (*vector of features*) | targets: Y

X_1 (3.5, -2, ... , 127, 0, ...) +1 Y_1

(-9.2, 32, ... , 24, 1, ...) -1

etc...

$X_{n,2}$

X_n (6.8, 54, ... , 17, -3, ...) +1 Y_n

Test point:

→ ?

Training set

number of examples n

input: | targets:

"horse"

"cat"

etc...

"horse"

preprocessing, feature extraction

Dimensionality of input d

inputs: X (*vector of features*) | targets: Y

X_1 (3.5, -2, ... , 127, 0, ...) +1 Y_1

(-9.2, 32, ... , 24, 1, ...) -1

etc...

$X_{n,2}$

X_n (6.8, 54, ... , 17, -3, ...) +1 Y_n

Test point:

→ ?

$x =$ (5.7, -27, ... , 64, 0, ...) →

Training set

number of examples n

input: | targets:

"horse"

"cat"

etc...

"horse"

preprocessing, feature extraction

Dimensionality of input d

inputs: X (*vector of features*) | targets: Y

X_1 (3.5, -2, ... , 127, 0, ...) +1 Y_1

(-9.2, 32, ... , 24, 1, ...) -1

etc...

$X_{n,2}$

X_n (6.8, 54, ... , 17, -3, ...) +1 Y_n

Test point:

→ ?

$x =$ (5.7, -27, ... , 64, 0, ...) → ?

Importance of the problem dimension

⇨ Determines which learning algorithms are applicable or not (because of their algorithmic complexity).

- Number of examples: **n**
  (often in the millions of more)
- Dimension of input: **d**
  how many features we are using
  (often in the order of 100 to 1000, sometimes in the 10s of thousands)
- Dimensions of the target we want to predict eg., number of classes m (often low, sometimes thousands)

→ A dataset will often be organized
in a matrix n x (d+1) or n x (d+m)

Terminology of supervised learning

One input generally represented by a vector of dimension d.

$x \in \mathbb{R}^d$

dimension of entry

1	observation, *input*, x_1	*target*, desired output, y_1
2	observation, *input*, x_2	*target*, desired output, y_2
3	observation, *input*, x_3	*target*, desired output, y_3
⋮	*etc…* ⋮	⋮
n	observation, *input*, x_n	*target*, desired output, y_n

size of the dataset, number of examples, or "samples"

? ? ?

Terminology of supervised learning

Training set

1	observation, *input*, x_1	*target*, desired output, y_1
2	observation, *input*, x_2	*target*, desired output, y_2
3	observation, *input*, x_3	*target*, desired output, y_3
⋮	*etc…* ⋮	⋮
n	observation, *input*, x_n	*target*, desired output, y_n

One input generally represented by a vector of dimension d.

$x \in \mathbb{R}^d$

dimension of entry

size of the dataset, number of examples, or "samples"

→ ? ? ?

Terminology of supervised learning

Training set

One input generally represented by a vector of dimension d.

$x \in \mathbb{R}^d$

dimension of entry

1	observation, *input*, x_1	*target*, desired output, y_1
2	observation, *input*, x_2	*target*, desired output, y_2
3	observation, *input*, x_3	*target*, desired output, y_3
⋮	*etc…* ⋮	⋮
n	observation, *input*, x_n	*target*, desired output, y_n

size of the dataset, number of examples, or "samples"

test point → ? ? ?

Terminology of supervised learning

Training set

One input generally represented by a vector of dimension d.

$x \in \mathbb{R}^d$

dimension of entry

1	observation, *input*, x_1	*target*, desired output, y_1
2	observation, *input*, x_2	*target*, desired output, y_2
3	observation, *input*, x_3	*target*, desired output, y_3
⋮	*etc…*	⋮
n	observation, *input*, x_n	*target*, desired output, y_n

size of the dataset, number of examples, or "samples"

test point

? ? ?

Terminology of supervised learning

Training set

One input generally represented by a vector of dimension d.

$x \in \mathbb{R}^d$

dimension of entry

1	observation, *input*, x_1	*target*, desired output, y_1
2	observation, *input*, x_2	*target*, desired output, y_2
3	observation, *input*, x_3	*target*, desired output, y_3
⋮	*etc…*	⋮
n	observation, *input*, x_n	*target*, desired output, y_n

size of the dataset, number of examples, or "samples"

test point  ? ? ?

We are looking for an algorithm that produces an **output** that is a good prediction of the target.

This algorithm finds a good **function x -> y**

Terminology of supervised learning

- When the target is a class label, a categorical variable (indicating which class or category the entry belongs to, among many) is said to be a **CLASSIFICATION** problem. (we often use an integer as a label).
- When the target is one (or more) real values to predict, we talk about a **REGRESSION** problem.

When we do not have an explicit target, we are in the context of **unsupervised learning.**

Learning tasks

Supervised learning = predict a target y from the input x

Learning tasks

Supervised learning = predict a target y from the input x

Unsupervised learning: no explicit target (label) y

Learning tasks

Supervised learning = predict a target y from the input x

- y represents a category or "class"
  ➠classification (binary or multiclass)
- y is a real value
  ➠ regression

Unsupervised learning: no explicit target (label) y

Learning tasks

Supervised learning = predict a target y from the input x

- y represents a category or "class"
  ➠classification (binary or multiclass)

- y is a real value
  ➠ regression

Unsupervised learning: no explicit target (label) y

- modeling the distribution of x
  ➠density estimation

- discover an underlying structure in x
  ➠ clustering
  ➠ dimensionality reduction (eg. for visualisation)

Learning tasks

Supervised learning = predict a target y from the input x

- y represents a category or "class"
  ➠classification (binary or multiclass)
- y is a real value
  ➠ regression

} predictive models

Unsupervised learning: no explicit target (label) y

- modeling the distribution of x
  ➠density estimation
- discover an underlying structure in x
  ➠ clustering
  ➠ dimensionality reduction (eg. for visualisation)

Learning tasks

Supervised learning = predict a target y from the input x

- y represents a category or "class"
  ➠classification (binary or multiclass)
- y is a real value
  ➠ regression

} predictive models

Unsupervised learning: no explicit target (label) y

- modeling the distribution of x
  ➠density estimation
- discover an underlying structure in x
  ➠ clustering
  ➠ dimensionality reduction (eg. for visualisation)

} descriptive models

Learning tasks

Semi-supervised learning:
same goal as supervised learning, but using both labeled (with target) and unlabeled examples.

Reinforcement learning
an artificial agent must learn to decide what actions to perform in a changing environment to maximize a total reward.

Data set seen as a scatter plot in a high-dimensional vector space

(input) $\mathbf{x} \in \mathbb{R}^d$ | (label) y

n examples

$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	
0.32	-0.27	+1	0	0.82	1
-0.12	0.42	-1	1	0.22	0
0.06	0.35	-1	1	-0.37	1
0.91	-0.72	+1	0	-0.63	1
...	...	...	...	...	...

Each example (row) is a vector of d+1-dimensions

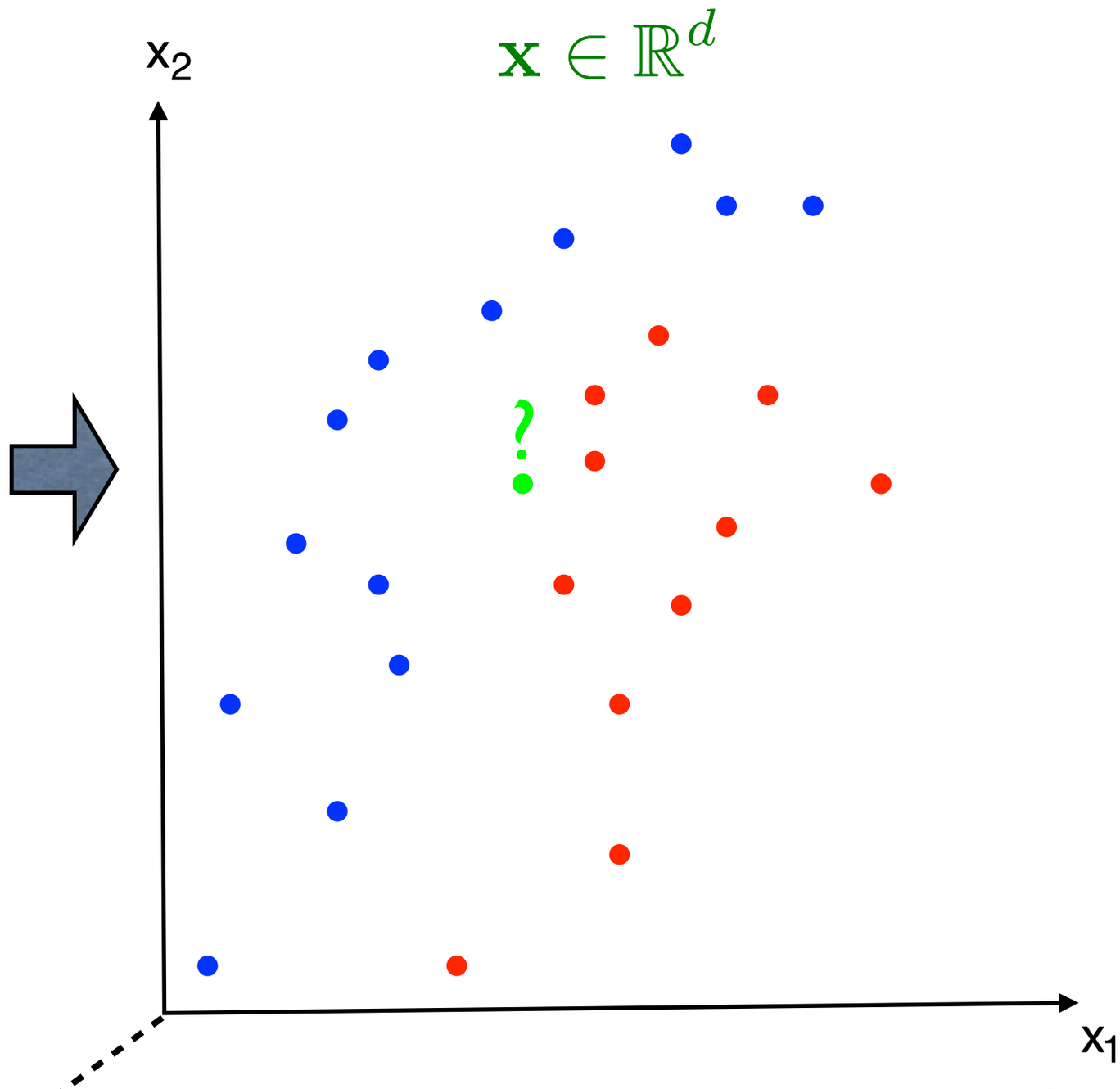
Data set seen as a scatter plot in a high-dimensional vector space

(input) $\mathbf{x} \in \mathbb{R}^d$ (label) y

n examples

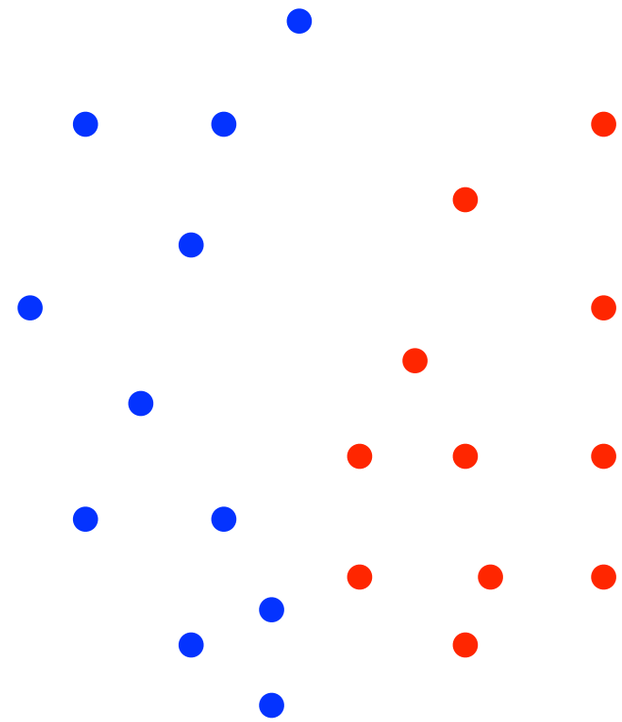
$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	
0.32	-0.27	+1	0	0.82	1
-0.12	0.42	-1	1	0.22	0
0.06	0.35	-1	1	-0.37	1
0.91	-0.72	+1	0	-0.63	1
...	...	...	...	...	...

Each example (row) is a vector of d+1-dimensions



Each entry is a point in a space of d-dimensions

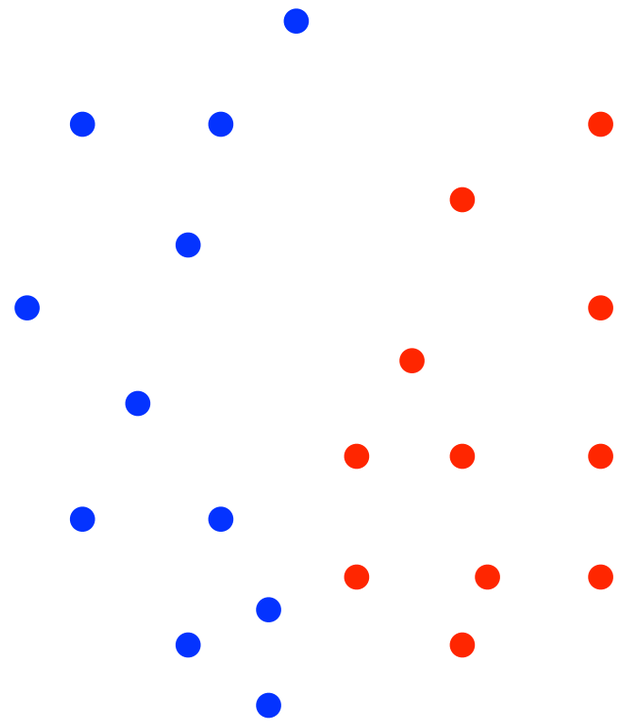
The nearest neighbor algorithm



Training set

The nearest neighbor algorithm

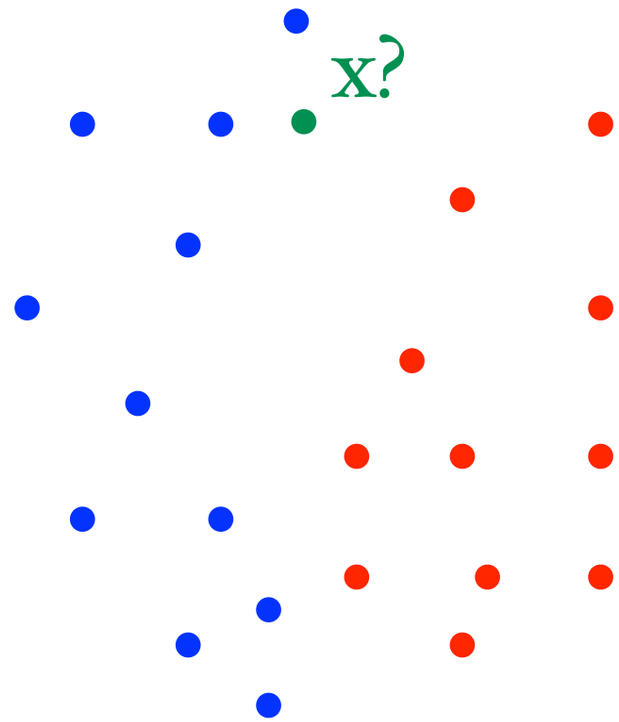
For a test point **x**:



Training set

The nearest neighbor algorithm

For a test point x:

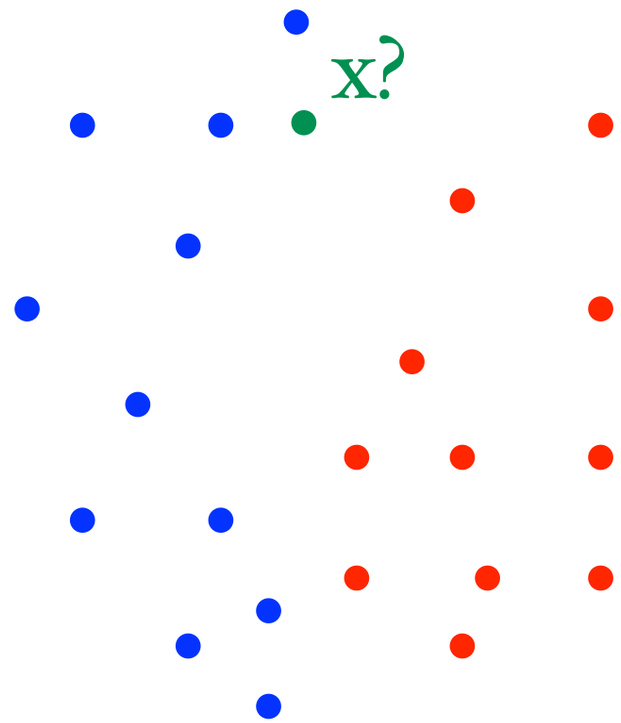


Training set

The nearest neighbor algorithm

For a test point x:

- We find the **nearest neighbour** of x within the training set by some measure of distance (eg Euclidean distance).

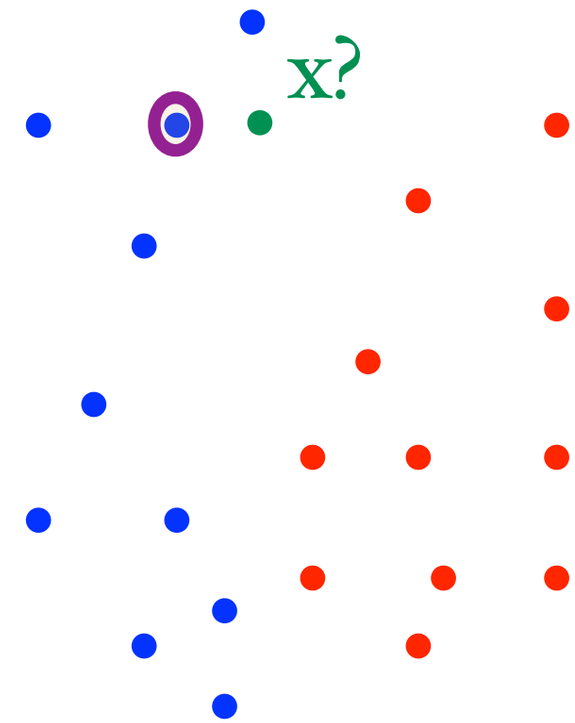


Training set

The nearest neighbor algorithm

For a test point x:

- We find the **nearest neighbour** of x within the training set by some measure of distance (eg Euclidean distance).



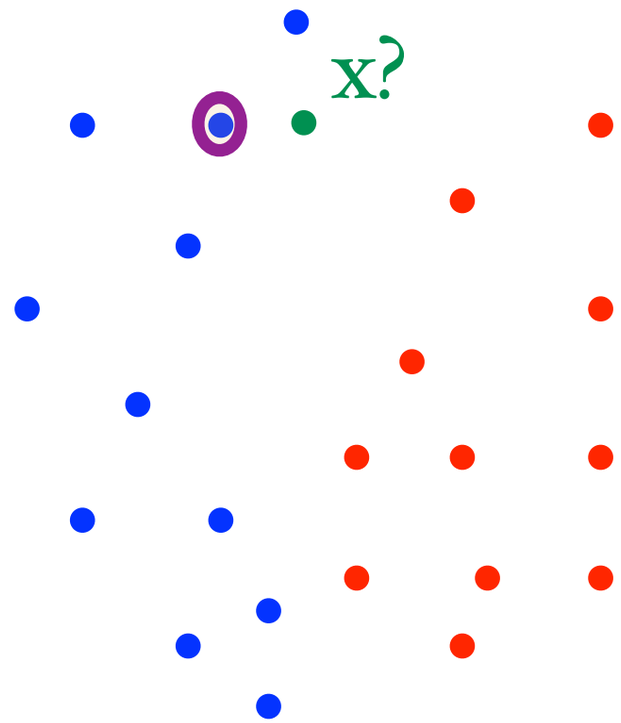
Training set

The nearest neighbor algorithm

For a test point x:

- We find the **nearest neighbour** of x within the training set by some measure of distance (eg Euclidean distance).

- We associate x with the class of this nearest neighbor.



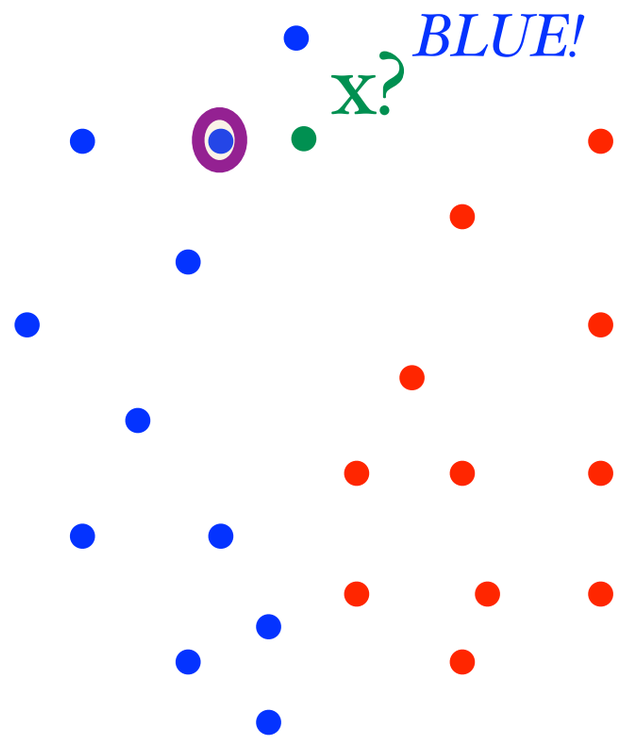
Training set

The nearest neighbor algorithm

For a test point x:

- We find the **nearest neighbour** of x within the training set by some measure of distance (eg Euclidean distance).

- We associate x with the class of this nearest neighbor.



Training set

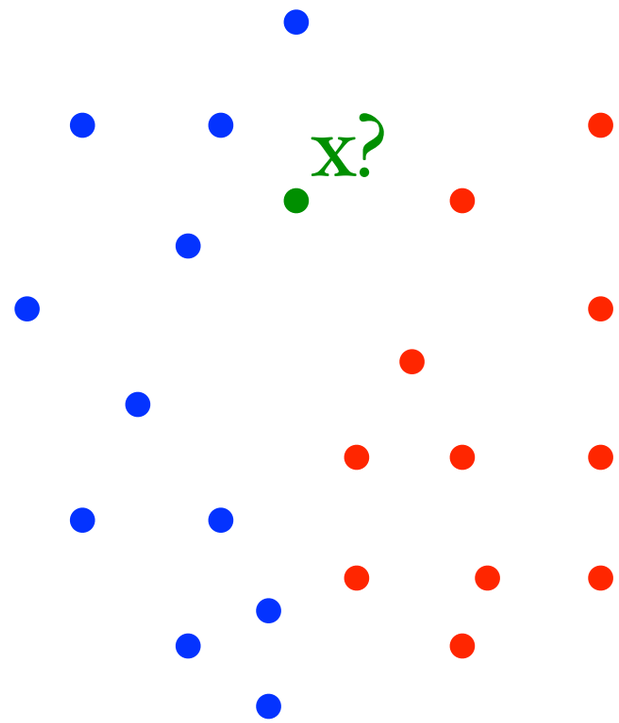
The k-nearest neighbors algorithm (k-NN)

For a test point x:

k=3

- We find the **k nearest neighbors** of x within the training set apprentissage by some measure of distance (eg Euclidean distance).

- We associate x with the class of the **majority** of neighbors.



Training set

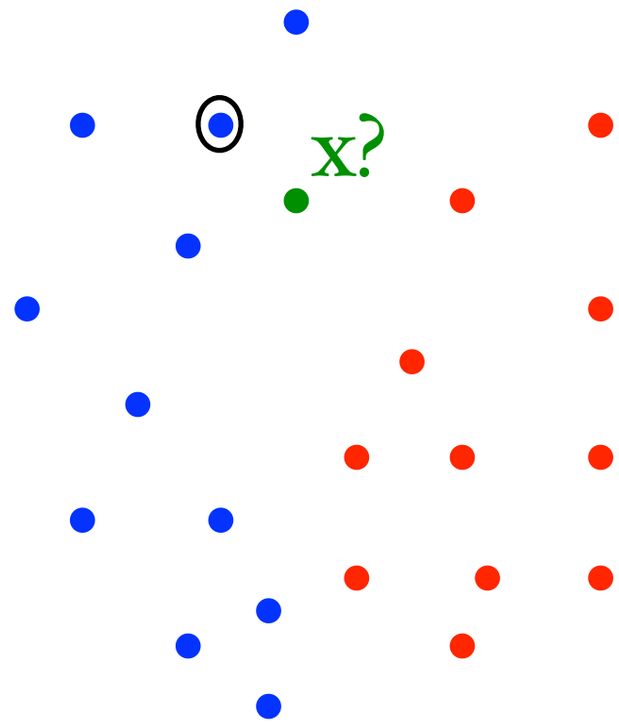
The k-nearest neighbors algorithm (k-NN)

For a test point x:

k=3

- We find the **k nearest neighbors** of x within the training set apprentissage by some measure of distance (eg Euclidean distance).

- We associate x with the class of the **majority** of neighbors.



Training set

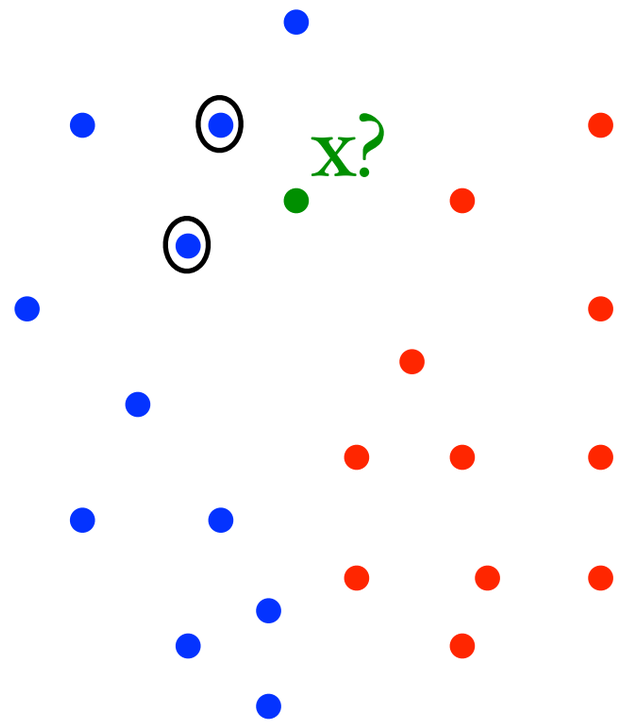
The k-nearest neighbors algorithm (k-NN)

For a test point x:

k=3

- We find the **k nearest neighbors** of x within the training set apprentissage by some measure of distance (eg Euclidean distance).

- We associate x with the class of the **majority** of neighbors.



Training set

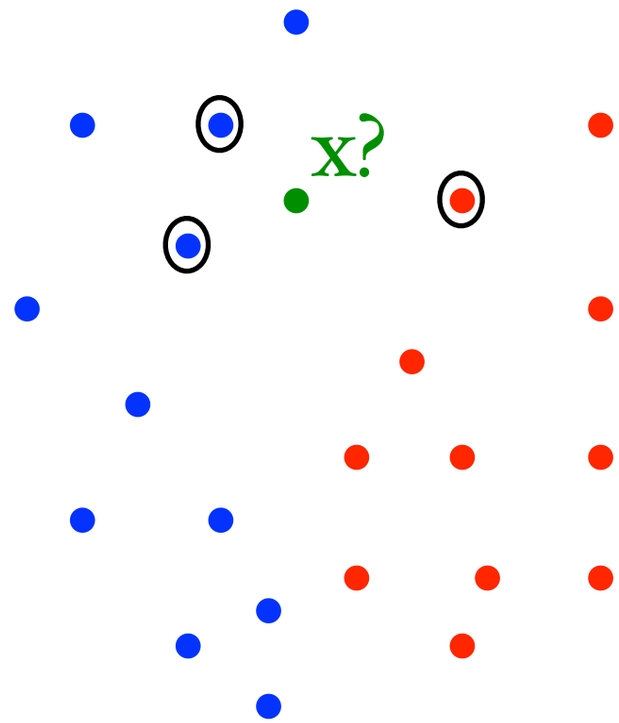
The k-nearest neighbors algorithm (k-NN)

For a test point x:

k=3

- We find the **k nearest neighbors** of x within the training set apprentissage by some measure of distance (eg Euclidean distance).

- We associate x with the class of the **majority** of neighbors.



Training set

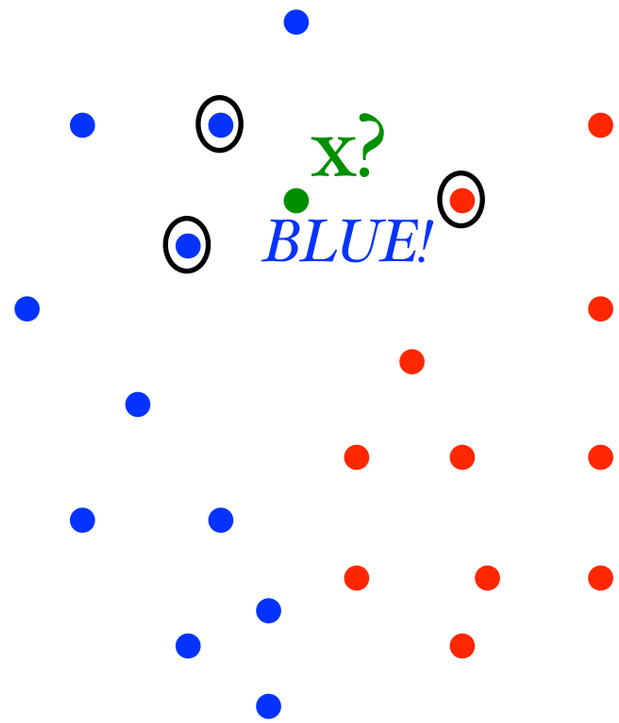
The k-nearest neighbors algorithm (k-NN)

For a test point x:

k=3

- We find the **k nearest neighbors** of x within the training set apprentissage by some measure of distance (eg Euclidean distance).

- We associate x with the class of the **majority** of neighbors.



Training set

Short review of vectors

(of dimension d)

- what is a vector? mathematical, computer, graphic representations ...
- Euclidean distance
- norms
- scalar product
- efficient calculation of the distance from a point to a set of points

The notion of level of representation