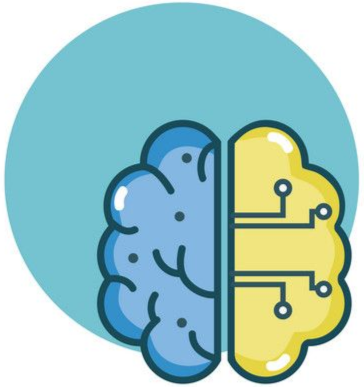
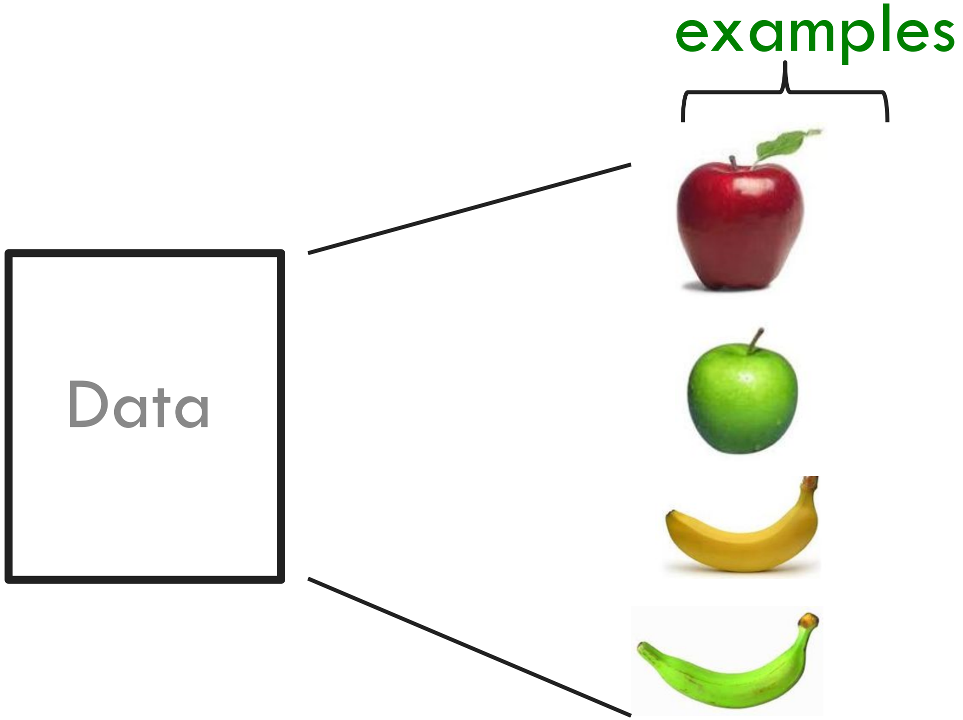
INTRODUCTION TO MACHINE LEARNING

K-Nearest neighbor

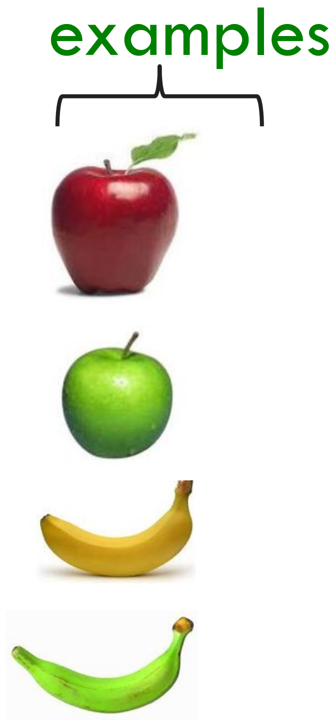


Elisa Ricci

Recap: Data



Recap: Representing examples



What is an an example?

How is it represented?

Recap: Features

examples

features

$f_1, f_2, f_3, \ldots, f_n$

$f_1, f_2, f_3, \ldots, f_n$

$f_1, f_2, f_3, \ldots, f_n$

$f_1, f_2, f_3, \ldots, f_n$

How our algorithms actually "view" the data

Recap: Features

examples

features

red, round, leaf, 3oz, …

green, round, no leaf, 4oz, …

yellow, curved, no leaf, 8oz, …

green, curved, no leaf, 7oz, …

How our algorithms actually “view” the data

Features are the questions we can ask about the examples

Features in generally are represented with vectors

Apples vs. Bananas

Weight	Color	Label
4	Red	Apple
5	Yellow	Apple
6	Yellow	Banana
3	Red	Apple
7	Yellow	Banana
8	Yellow	Banana
6	Yellow	Apple

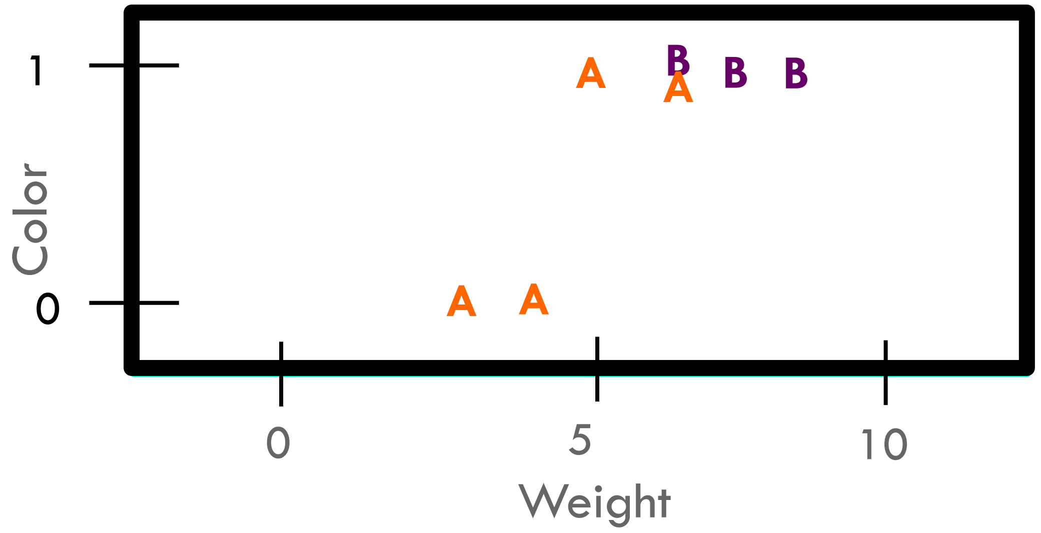
Can we visualize this data?

Apples vs. Bananas

We can turn features into numerical values

peso in grammi

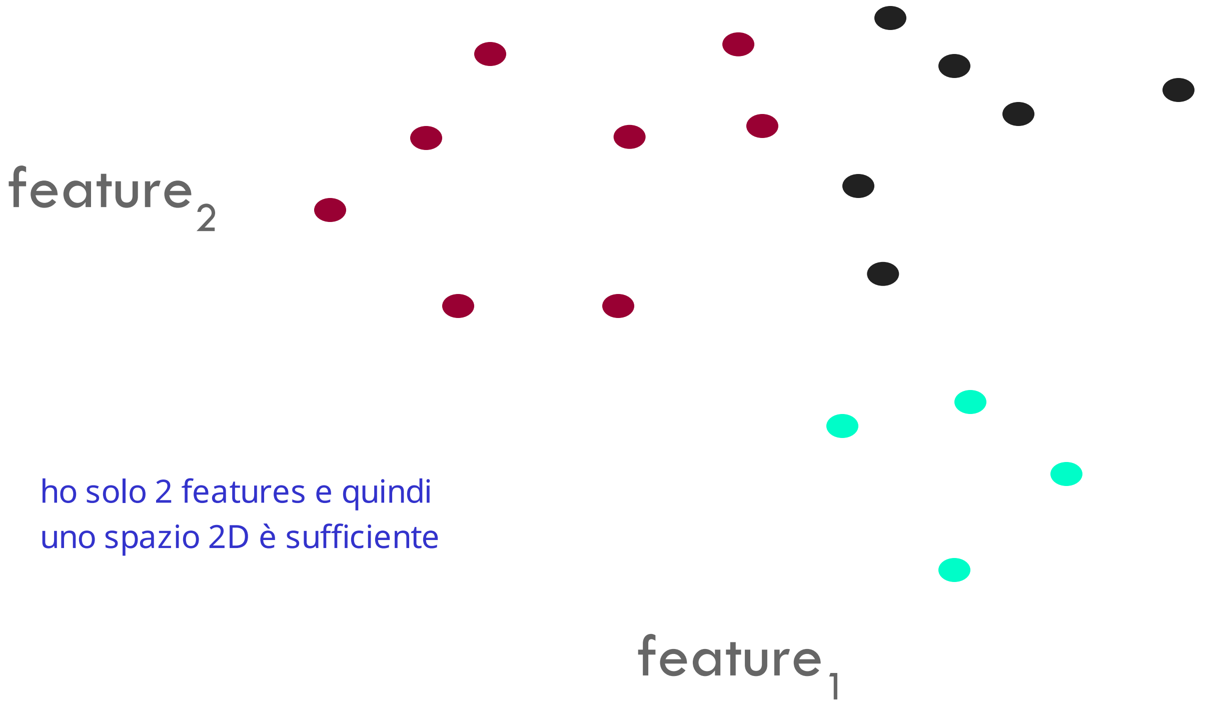
Weight	Color	Label
4	0	Apple
5	1	Apple
6	1	Banana
3	0	Apple
7	1	Banana
8	1	Banana
6	1	Apple



Examples are points in an *n*-dimensional space where *n* is the number of features

Examples in a feature space

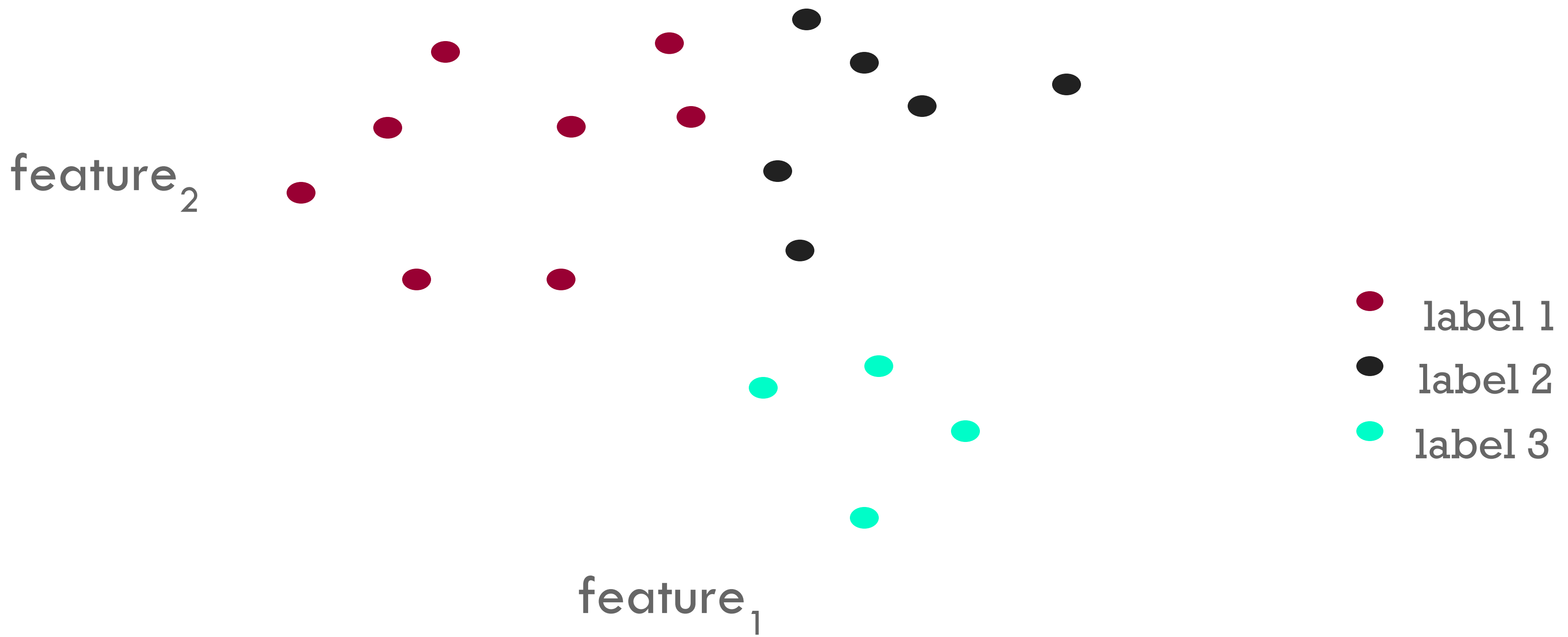
We can imagine to have 3 categories of interest



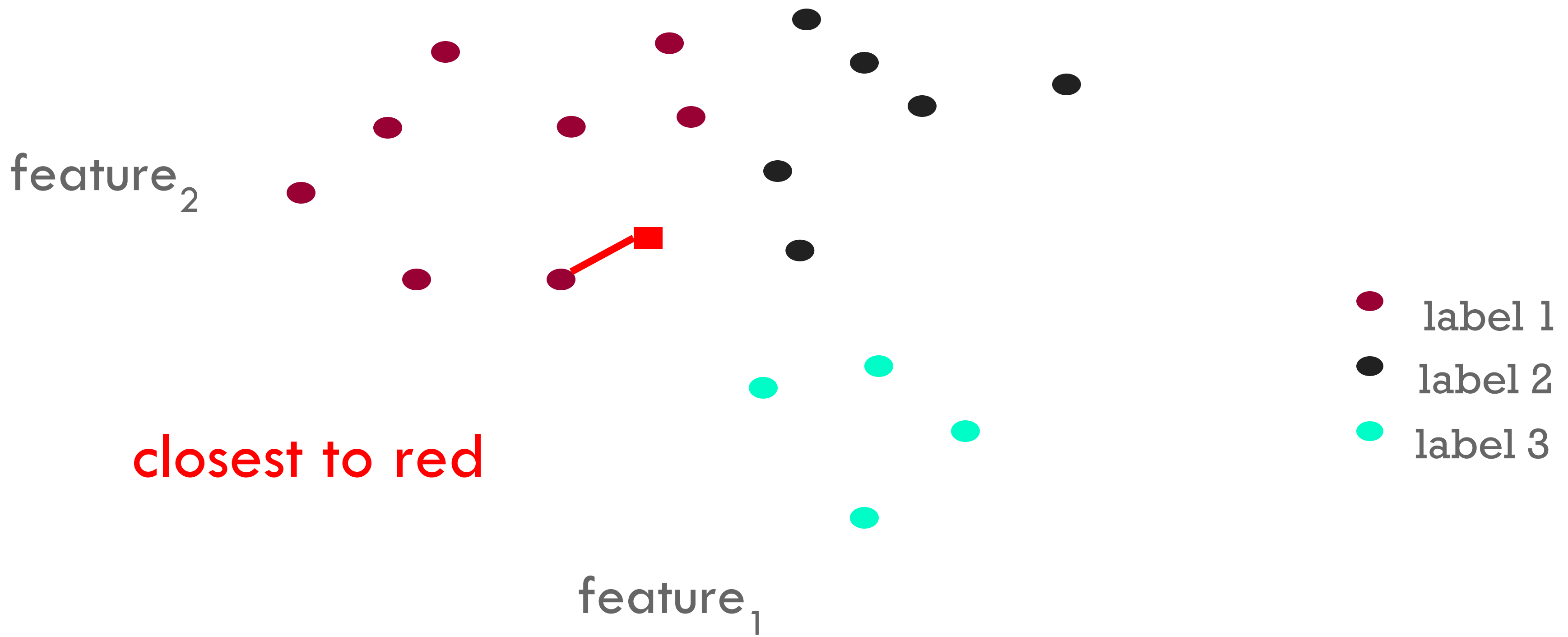
● label 1
● label 2
● label 3

ho solo 2 features e quindi
uno spazio 2D è sufficiente

Training set

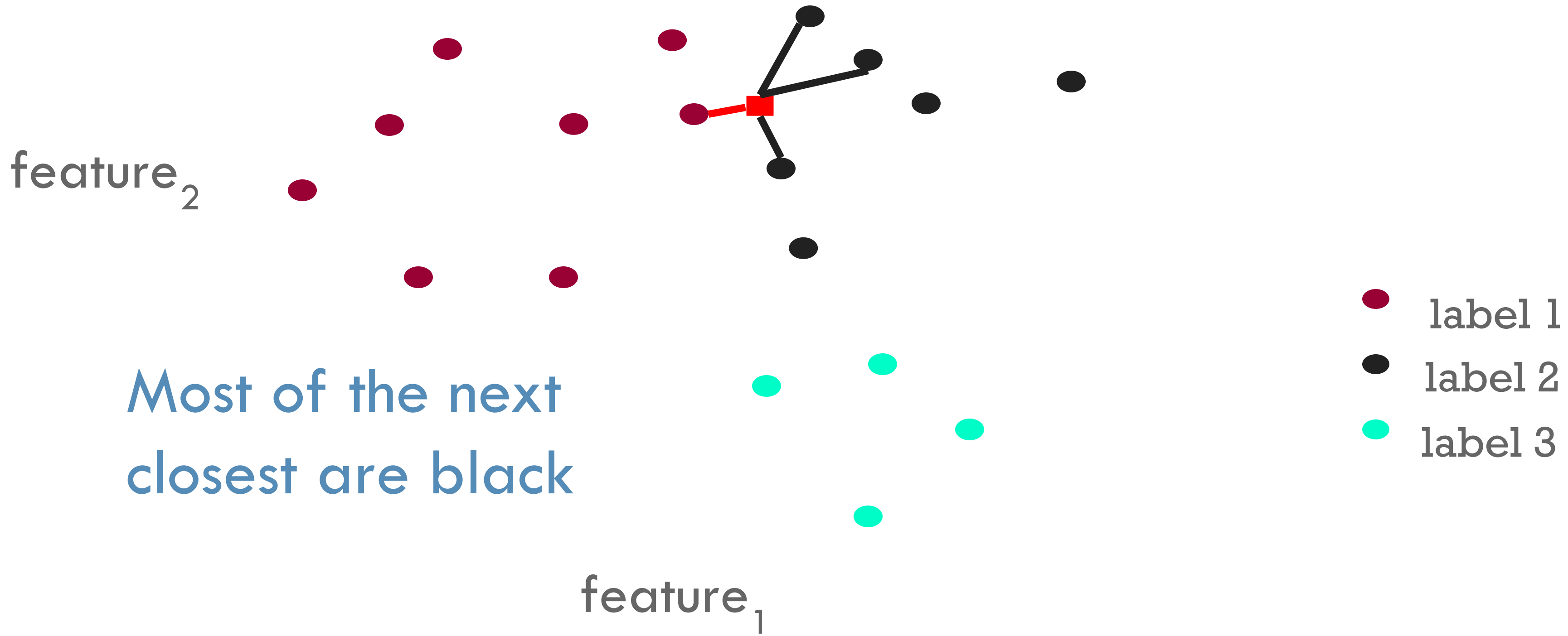


Test example: which class?



closest to red

What about this example?



Most of the next closest are black

k-Nearest Neighbor (k-NN)

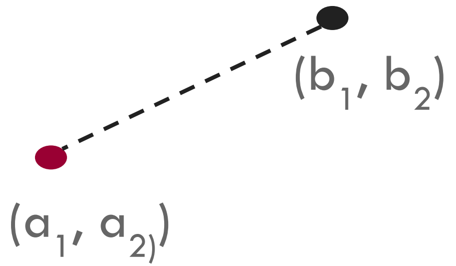
To classify an example ***d***:

- Find ***k*** nearest neighbors of ***d***
- Choose as the label the majority label within the ***k*** nearest neighbors

Robust to noisy data by considering k-nearest neighbors

Euclidean distance

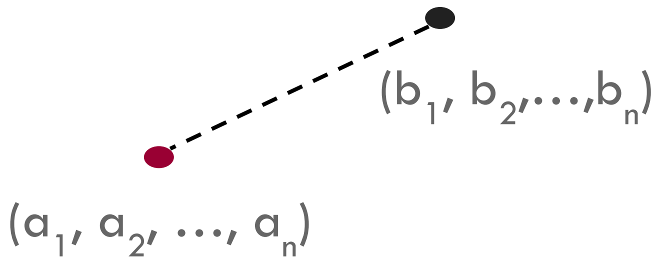
In two dimensions, how do we compute the distance?



$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

Euclidean distance

In n-dimensions, how do we compute the distance?



$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

Euclidean distance

- Euclidean Distance makes sense when different features are comparable (e.g. each is variable measured in the same units).
    - For instance, if the measurements are different, say length and weight, it is not clear.

data pre-processing

- **Standardization:** When features are not comparable we can standardize them by dividing by their corresponding standard deviation. This makes them all equally important.

Standardization & Scaling

- **Standardization or Z-score normalization**
  - Rescale the data so that the mean is 0 and the standard deviation from the mean (standard scores) is 1

$$x_{norm} = \frac{x - \mu}{\sigma}$$

μ is mean, σ is a standard deviation from the mean (standard score)

- **Min-Max scaling**
  - Scale the data to a fixed range – between 0 and 1

$$x_{\mathrm{morm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Distance and Similarity

- Measuring distance/similarity is a domain-specific problem and there are many, many different variations
- Similarity
    - Numerical measure of how alike two data objects are.
    - Is higher when objects are more alike.
    - Often falls in the range [0,1]
- Distance
    - Numerical measure of how different are two data objects
    - Lower when objects are more alike
    - Minimum dissimilarity is often 0
    - Upper limit varies

Minkowski Distance

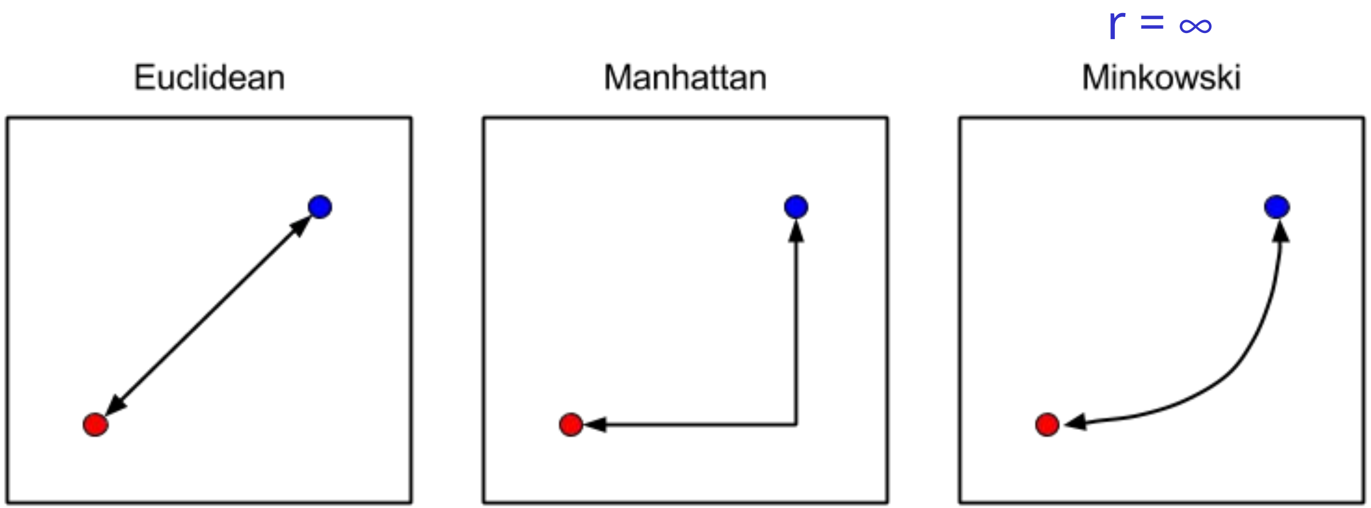
- Minkowski Distance is a generalization of Euclidean Distance

a, b = vettori

$$D(a,b) = \sum_{k=1}^{p} |a_k - b_k|^{r} \; {}^{1/r}$$

- r is a parameter, p is the number of dimensions
- $r = 1$. City block (Manhattan, L1 norm) distance.
- $r = 2$. Euclidean distance
- $r \rightarrow \infty$. “supremum” (L_∞ norm) distance.
  - This is the maximum difference between any component of the vectors

Distances

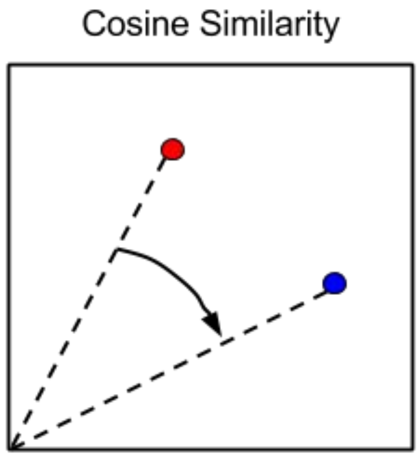


Cosine Similarity

Given two document vectors:

$$\cos(d_1, d_2) = (d_1 \cdot d_2) / ||d_1|| \, ||d_2||,$$

Where $\cdot$ indicates vector dot product and $||d||$ is the length of vector *d*.



Example:

d_1= 3 2 0 5 0 0 0 2 0 0

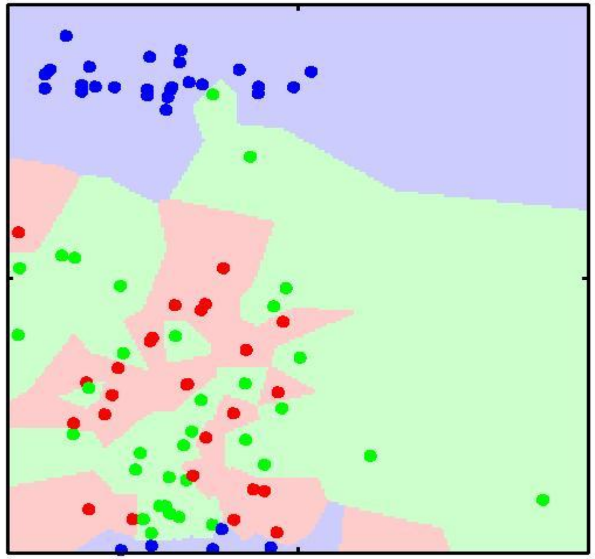
d_2= 1 0 0 0 0 0 0 1 0 2

$d_1 \cdot d_2$= 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5

$||d_1||$ = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)**0.5**= (42)^**0.5**= 6.481

$||d_1||$ = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)**0.5**= (6)^**0.5**= 2.245

cos(*d1*, *d2*) = .3150

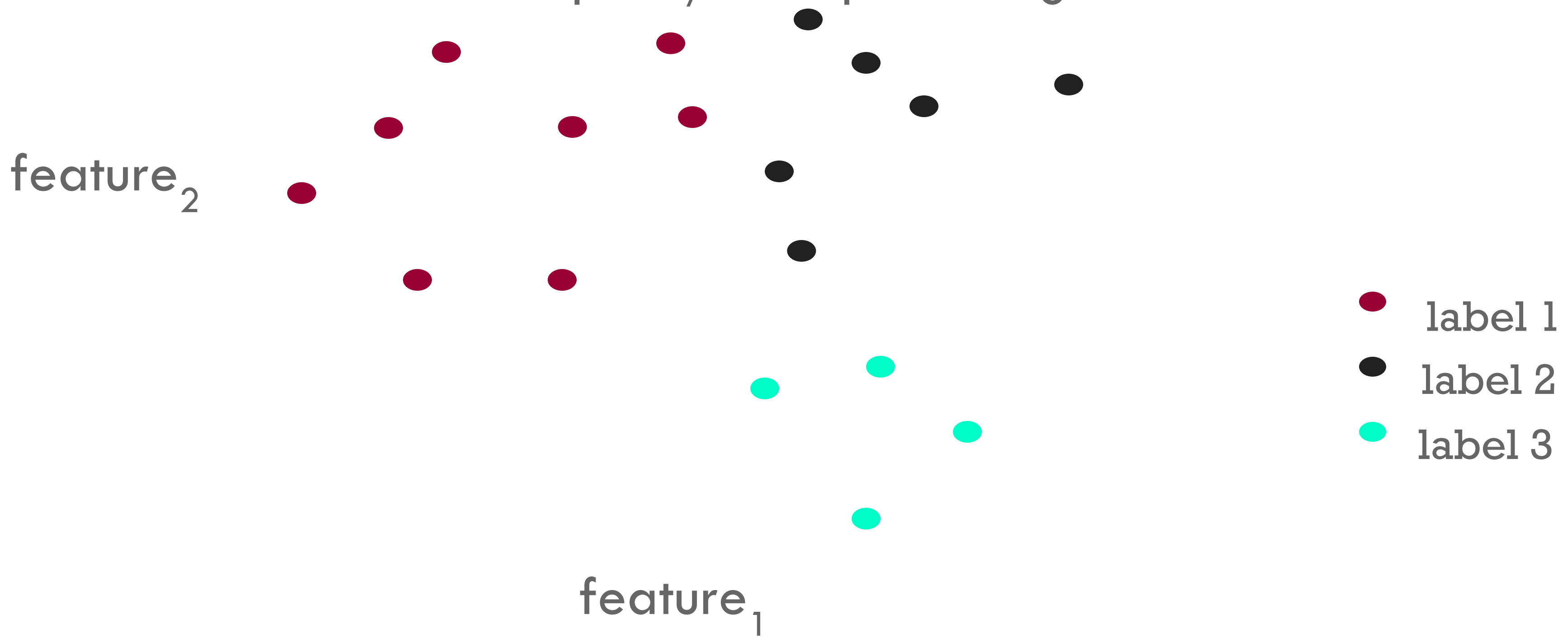


Decision Boundaries

Hyper-parameters

Decision boundaries

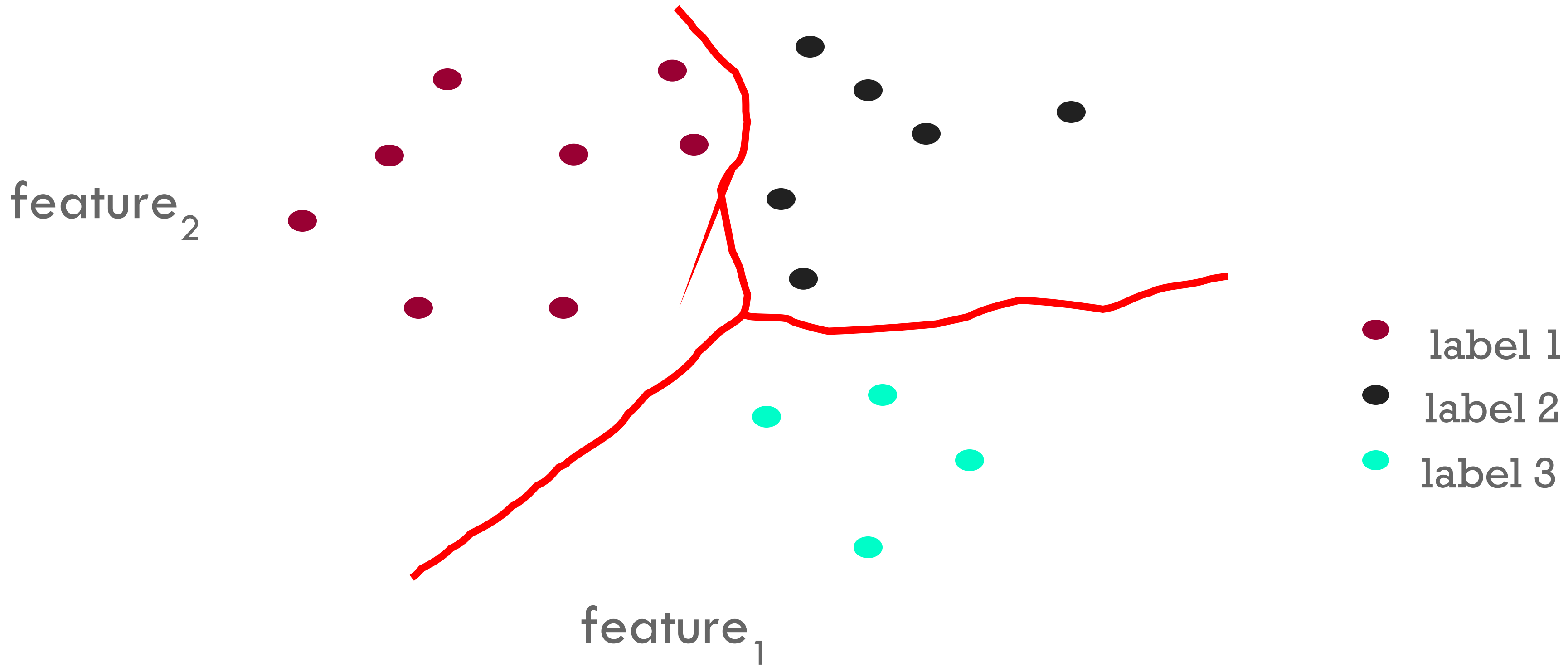
The **decision boundaries** are places in the features space where the classification of a point/example changes



Decision boundaries

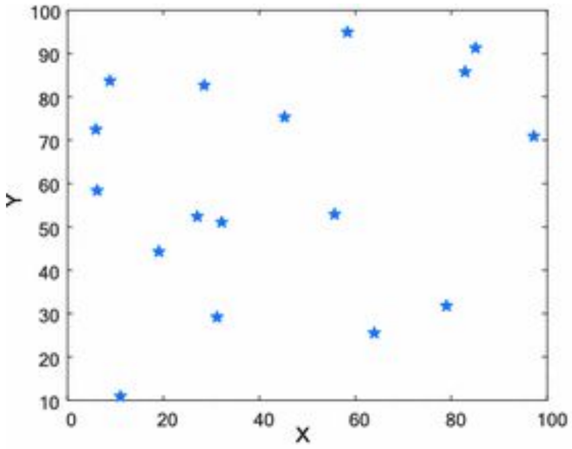
Voronoi

k-NN gives **locally** defined decision boundaries between classes

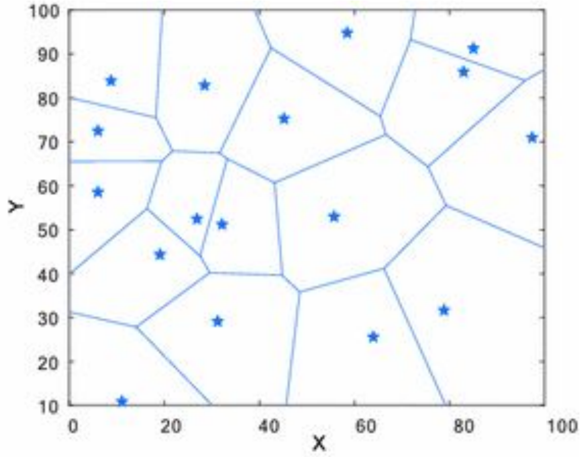


Voronoi Diagram

- Describes the areas that are nearest to any given point, given a set of data.
- Each line segment is equidistant between two points



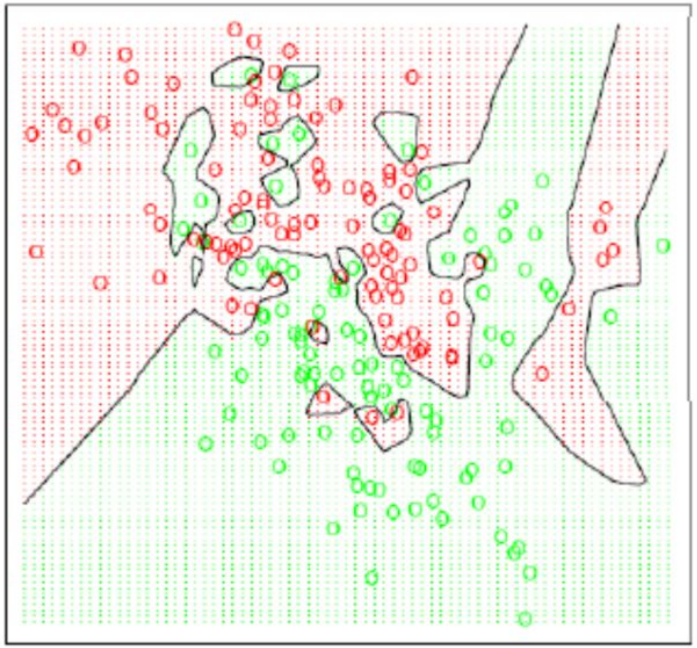
(a) Initial points distribution



(b) Voronoi diagram by initial points

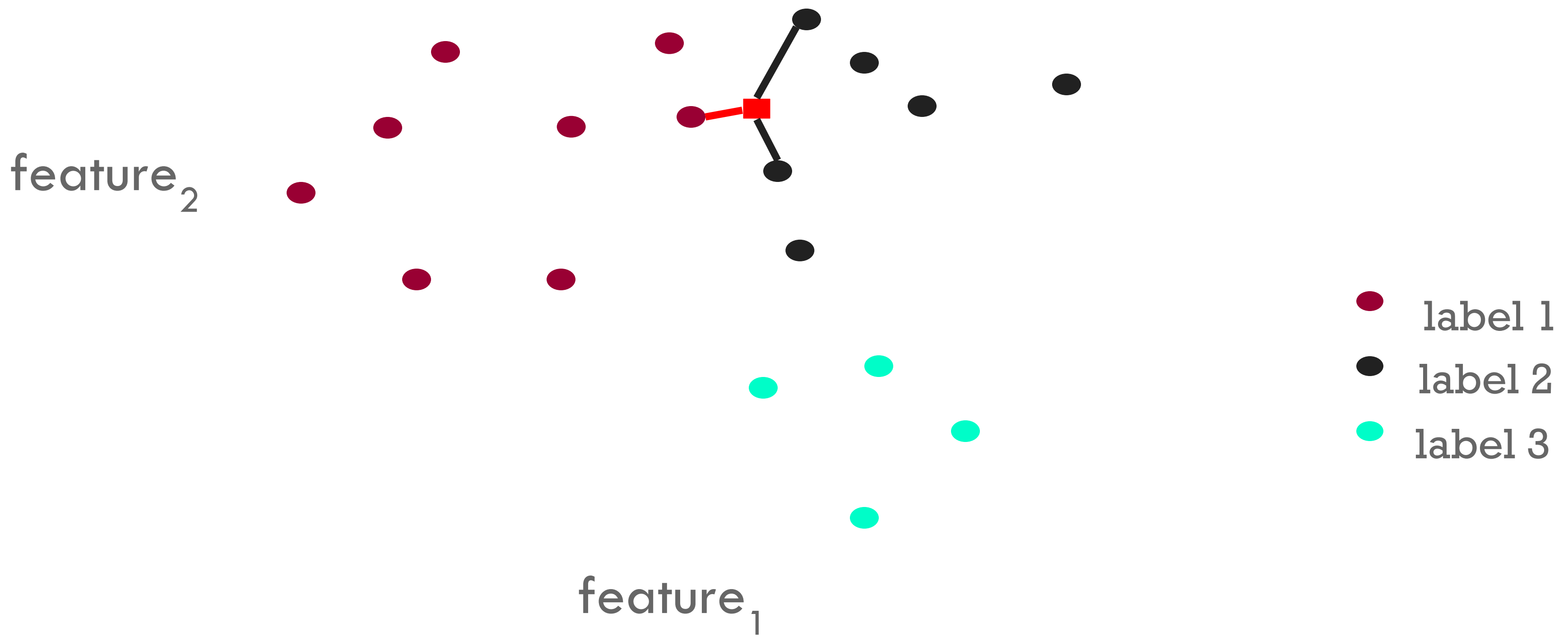
Decision boundaries

- k-NN algorithm does not explicitly compute decision boundaries.
- The decision boundaries form a subset of the Voronoi diagram for the training data.
- The more examples that are stored, the more complex the decision boundaries can become



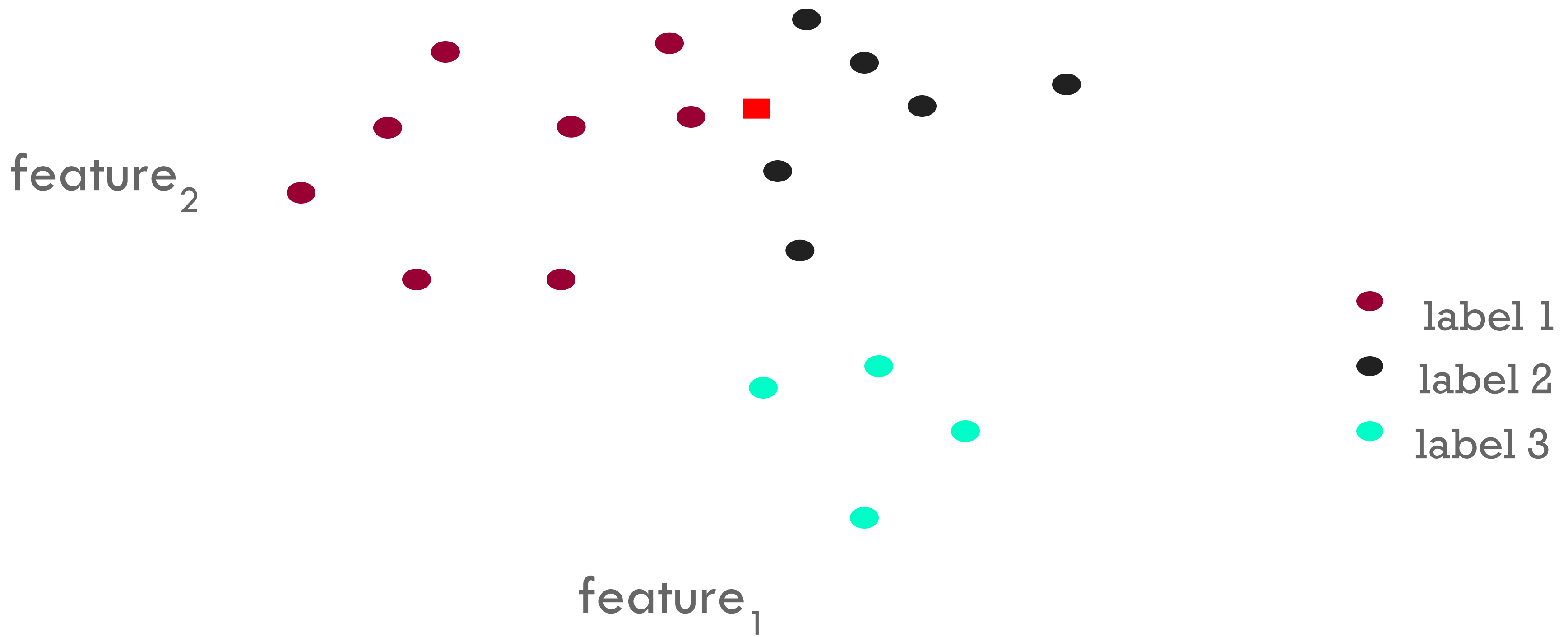
Choosing k

What is the label with k = 3?



Choosing k

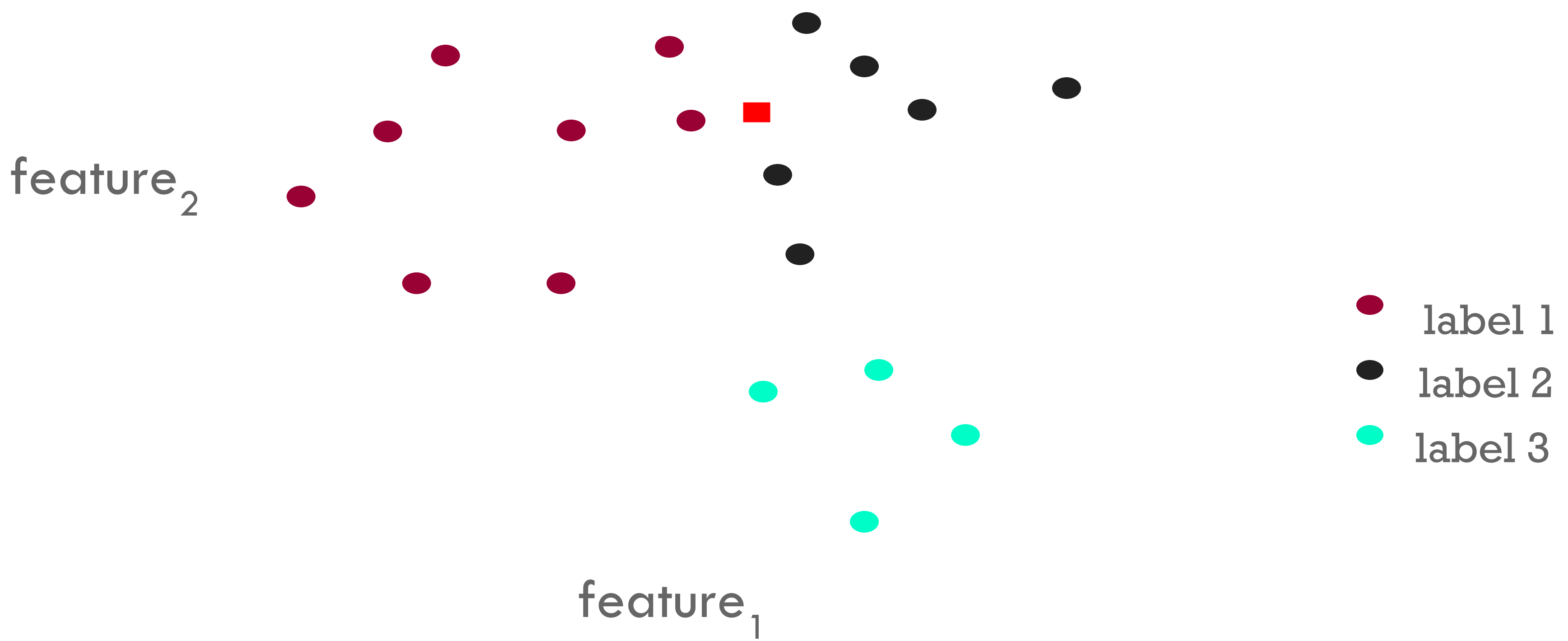
What is the label with k = 20?



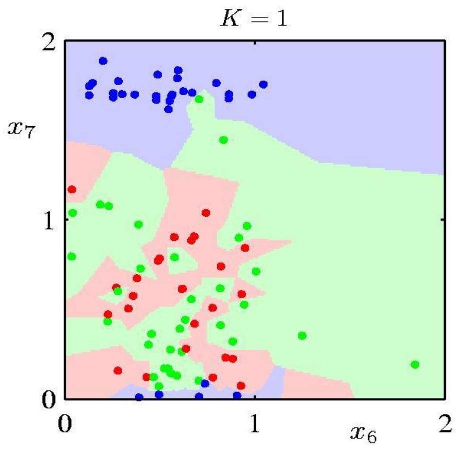
Choosing k

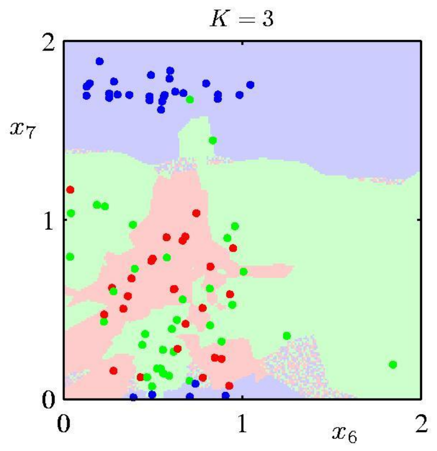
k deve essere sempre minore di n(numero di punti)

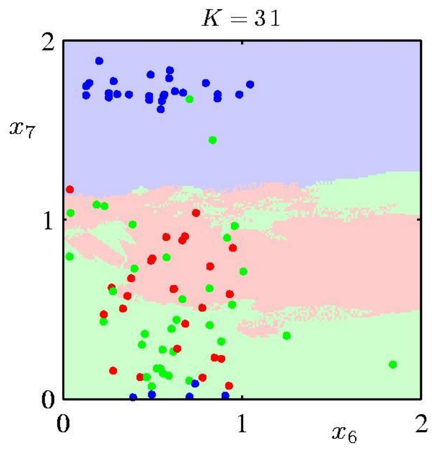
What is the label with k = 20? Red! (class with max number of datapoints, weird)



The impact of k







What is the role of k?

How does it relate to overfitting and underfitting?

RECAP: UNDERFITTING OVERFITTING

- The factors determining how well a machine learning algorithm will perform are its ability to:
    - Make the **training error small.**
    - Make the **gap between training and test error small.**
- These two factors correspond to the cases of **underfitting** and **overfitting.**
- **Underfitting** occurs when the model is not able to obtain a sufficiently low error value on the training set.
- **Overfitting** occurs when the gap between the training error and test error is too large.

How to pick k

- Common heuristics:
  - often 3, 5, 7
  - choose an odd number to avoid ties
- Use validation set
- Use cross-validation
- Rule of thumb is k < sqrt(n), where n is the number of training examples

k-NN variations

Instead of *k* nearest neighbors, count majority from all examples within a fixed distance

Weighted *k*-NN: weighted points

- Right now, all examples are treated equally
- Weight the “vote” of the examples, so that closer examples have more vote/weight
- Often use some sort of exponential decay

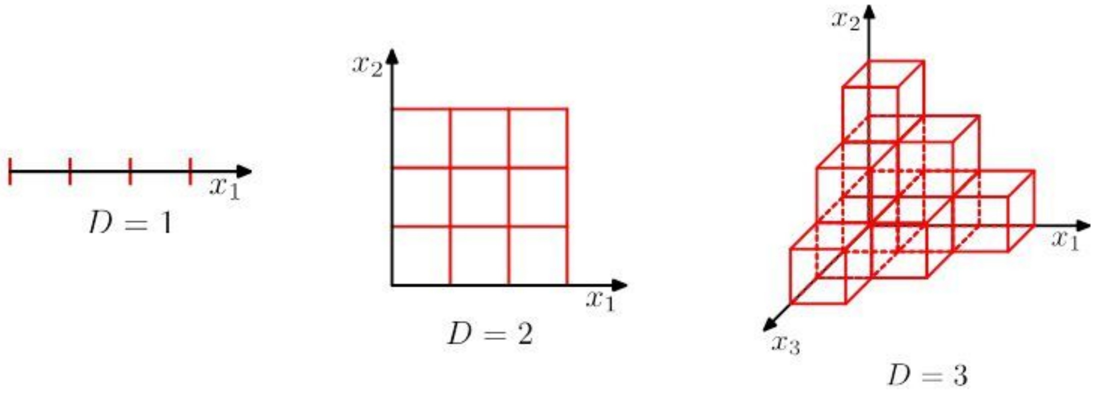
Lazy Learner VS Eager Learner

no training
processing only in the
inference phase

- k-NN is belongs to the class of **lazy learning** algorithms
- **Lazy learning:** Simply stores training data (or perform limited processing) and operates when it is given a test example
- **Eager learning** (e.g. Decision Trees, SVMs): Given a training set, constructs a classification model before receiving new test data to classify
- **Lazy learners**: less time in training but more time in predicting

Curse of Dimensionality

- **Curse of dimensionality:** In high dimensions almost all points are far away from each other
- The size of the data space grows exponentially with the number of dimensions.
- This means that the size of the data set must also grow exponentially in order to keep the same density.



Curse of Dimensionality

- The success of k-NN is very dependent on having a **dense data set.**
- Every machine learning algorithm needs a dense data set for accurate prediction
- What makes k-nearest neighbors special? tutte le features sono importanti perchè calcolo la distanza in tutte le dimensioni
- k-NN requires a point to be close in **every single dimension.**
- Some algorithms can create models based on single dimensions, and only need points to be close together along that axis.
- k-NN doesn't work that way. It needs all points to be close along every axis in the data space.

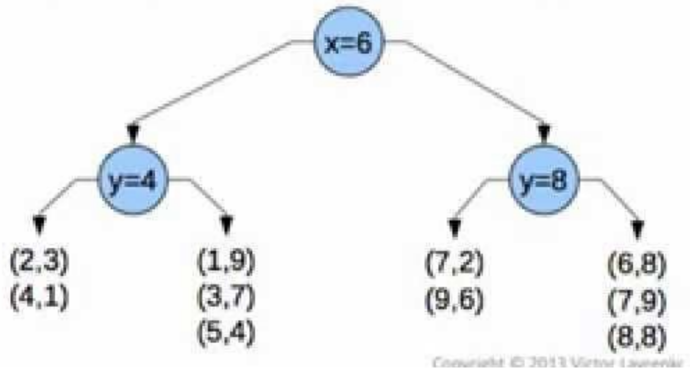
Computational cost

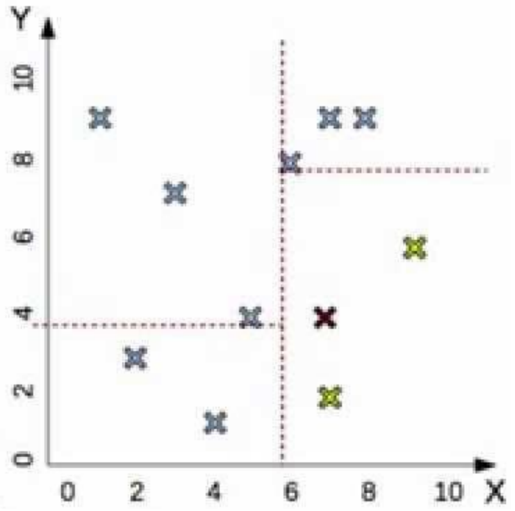
- Linear algorithm (no pre-processing):
  - Compute distance for all N datapoints
  - Complexity of distance computation: $O(kN)$
  - No additional space needed
- Tree-based **data structures**: pre-processing
  - Often used in applications: k-d trees (k-dimensional trees).

K-D Tree

l'idea è quella di muoversi un sottogruppo preciso prima di calcolare la "vicinanza"
computo questa struttura dati prima per velocizzare i calcoli durante la inference phase

- Building a K-D tree from training data:
  - {(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)}
  - pick random dimension, find median, split data, repeat
- Find NNs for new point (7,4)
  - find region containing (7,4)
  - compare to all points in region





An Aside: Parametric vs non-parametric models

- Parametric models we have a finite number of parameters (curve fitting)
  ex: parametri del polinomio
  - Linear regression, logistic regression, and linear Support Vector Machines
- Nonparametric models: the number of parameters is (potentially) infinite. The complexity of the model grows with the number of training data. (k-nn)
  - K-nearest neighbor, decision trees, or RBF kernel SVMs are considered as non-parametric learning algorithms since the number of parameters grows with the size of the training set.

k-Nearest Neighbor: Summary

- Non parametric model
- When to consider
  - Instance map to points in R^N
  - Few (e.g. less than 20) features per instance
  - Lots of training data
- Advantages
  - Training is very fast (no training)
  - Learn complex target functions
- Disadvantages
  - Slow at query time
  - Easily fooled by irrelevant attributes

k-Nearest Neighbor: Advantages

- Easy to program
- No optimization or training required
- Classification accuracy can be very good, it can outperform more complex models

resisted the test of time

Example

Decent performance when large training set

- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

specialized distance

k-Nearest Neighbor: Issues

- Choose distance measure
    - Most common: Euclidean
- Choosing k
- Curse of Dimensionality
- Memory based technique.
    - Must make a pass through the data for each classification. This can be prohibitive for large data sets.

QUESTIONS?