Universidad de Alcalá

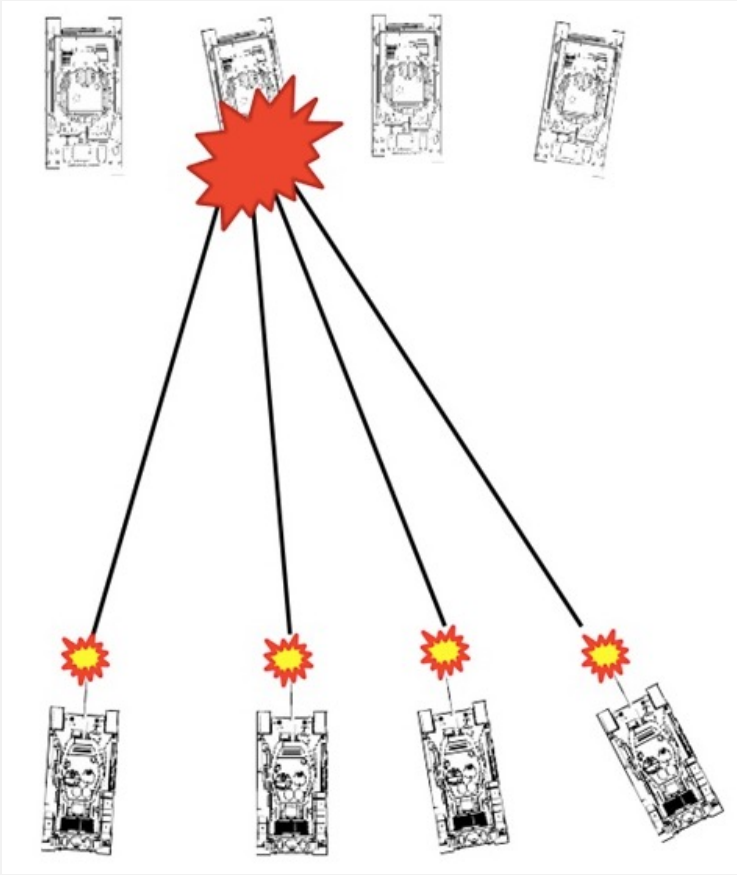
Analogical Learning: K-nn

Prof. Ignacio Olmeda
AI LAB

ALGORITHM INTUITION

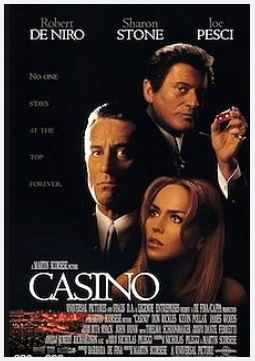
- Nearest neighbors algorithm exploits the idea of *analogical reasoning*.

- In some situations we may try to find the solution to a particular problem thinking on solutions to similar problems.

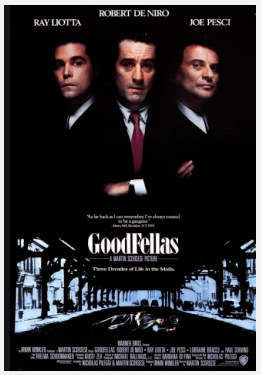


- Another example, when netflix recommends films it can e.g find the filsm that have similar characterististics to movies we have watched and that we liked.

gangsters al pacino drama … action ★★★★★

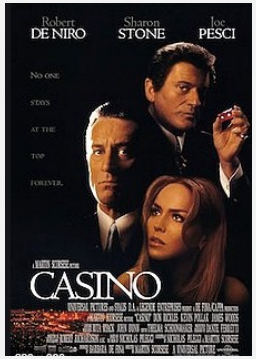


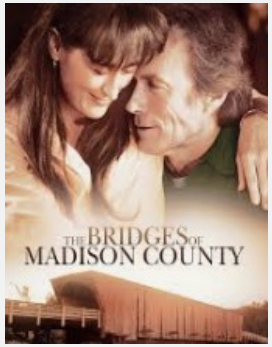
gangsters al pacino drama … action ★★★★★

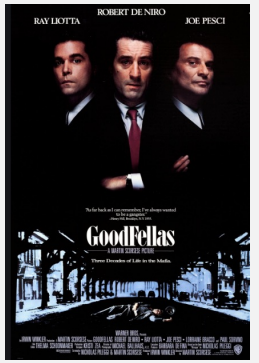


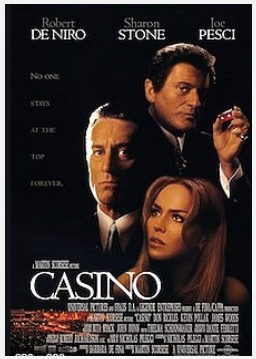
gangsters al pacino drama … action ? ? ? ? ?

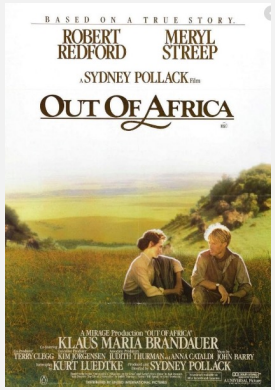
- Or, if there are few of our ratings:

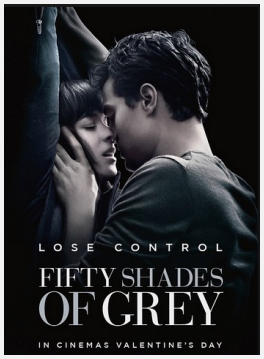












? ? ? ?

- Each of these approaches give rise to different kinds of recommender systems, i.e.:
  - *Collaborative Filtering*: assume that users whose tastes are similar to others in the past will continue to resemble each other in the future and that tastes' patterns will not change.
  - *Content based filtering*: based on the description of the item to be recommended and user preferences
  - *Hybrid systems*: combining both, e.g. Netflix.

- The nearest neighbors algortithm exploits the idea that patterns repeat.

- The idea is very simple: assume you have two patterns:

	X					Y
pattern A	1	4	3	0	2	9
pattern B	4	6	0	1	3	5

- And let us suppose that a new –partial- pattern is observed

	X					Y
pattern C	1	4	3	1	2	?

- Notice that pattern A and C share 4 out of 5 features while pattern B and C share only one.

- If we had to guess the value of Y then the intuitive answer would be 9 because 9 is the value of Y for the closest pattern.

- Obviously, the situation can be a bit more complex, for example we might have

	X					Y
pattern A	1	4	3	0	2	9
pattern B	4	4	0	1	3	5

	X					Y
pattern C	1	4	0	1	2	?

- Notice that in this case, pattern C shares 3 characteristics with pattern A and another 3 with pattern B so a "mixed" predition of Y could be Y=(9+5)/2=7.

- Also, not all the attributes need to be exactly the same

	X					Y
pattern C	0.9	3.5	0	1	2.2	?

So that some "distance" must be defined.

- Finally, note that alll the attributes need not be equally important, for example attribute 1 and 3 could be more relevant so that "proximity" should consider this.

- The nearest neighbor method considers these and other extensions to build nonparametric estimates of the underlying function.

- The nearest neighbor algorithm is also called *instance-based learning* or *lazy learning* because, in fact, no formal model is produced, the model is the data.

- This method is also called Memory-based, Instance-based , Exemplar-based , Case-based, or Experience-based.

ALGORITHM DESIGN

- An item with D characteristics can be seen as a point in a D-dimensional space:

$$(x^1, x^2, \ldots, x^D)$$

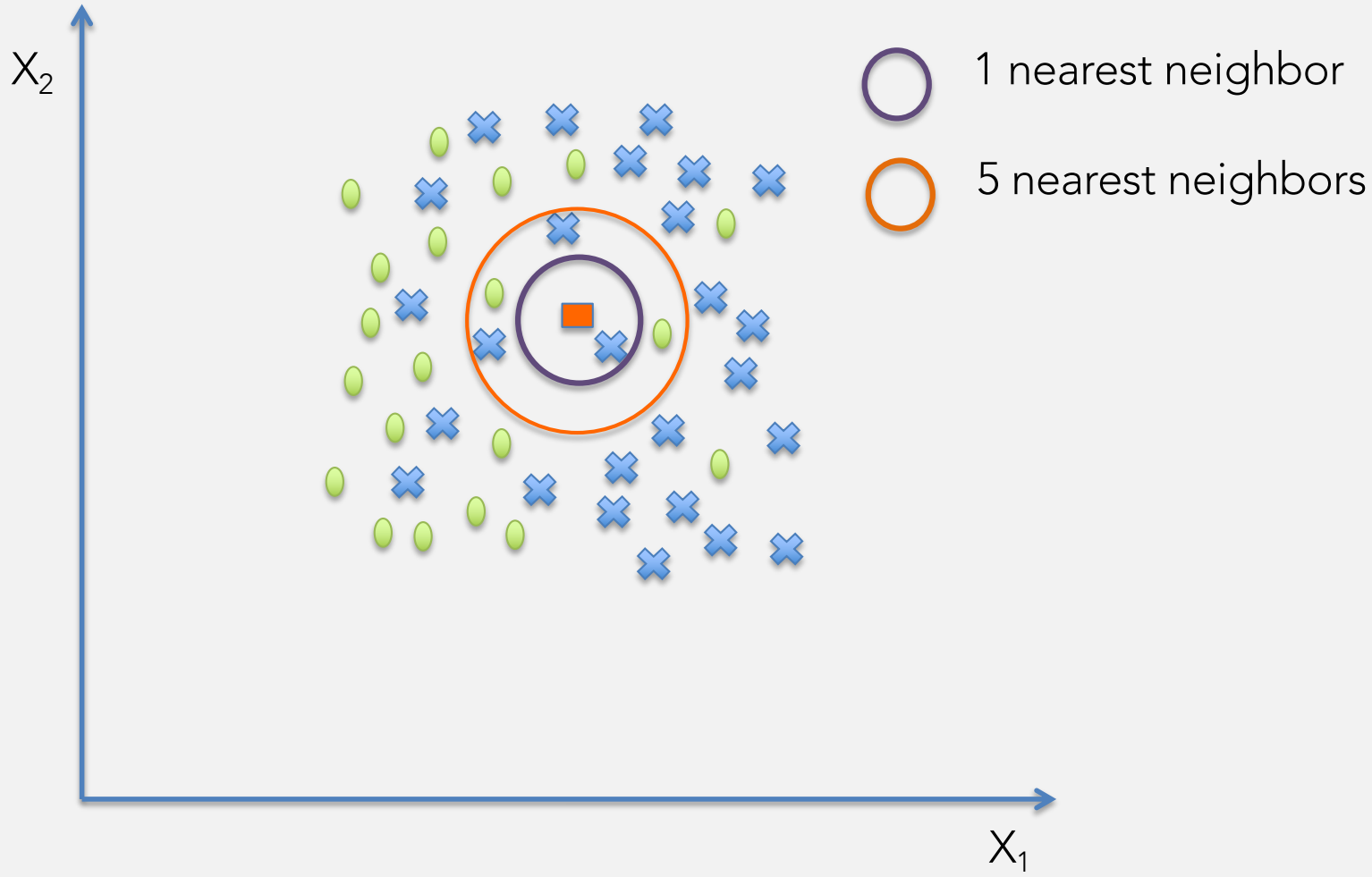
- And similarly, using this geometric intuition, we can measure the proximity of two examples using an appropriate distance function, for example the *Euclidean distance*:

$$d_{euclidian}(x_i, x_j) = \sqrt{\sum_{k=1}^{D} (x_i^k - x_j^k)^2}$$

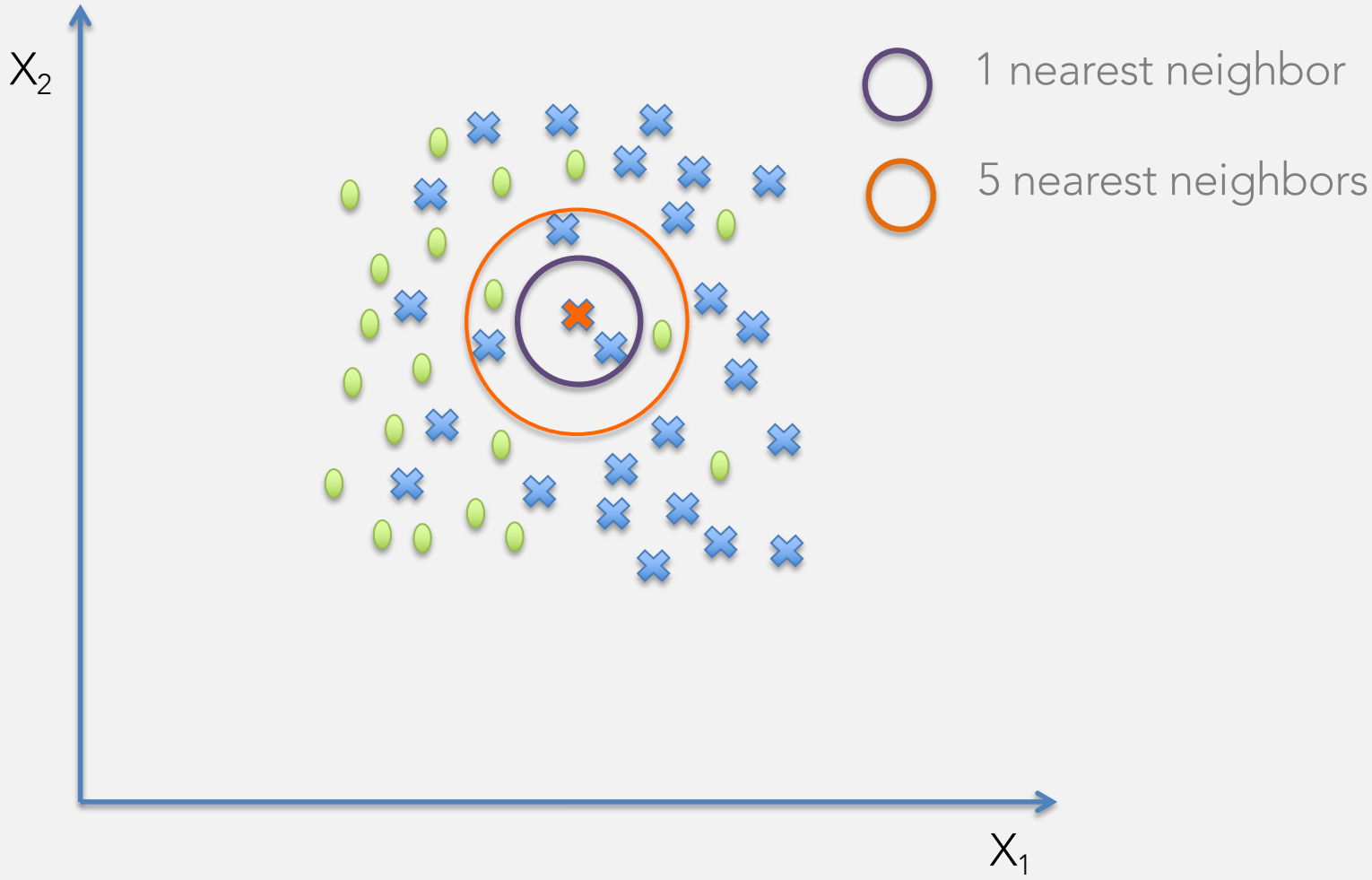
- Which using matrix notation is:

$$d_{euclidian}(x_i, x_j) = \sqrt{(x - x')^T (x - x')}$$

- After distances among all the datapoints are computed, it is possible to choose a "neighbor" of any datapoint by considering the k examples which are closer;



- Finally, we can assign e.g. the class that is more representative in the neighborhood



- "Representativeness" can be interpreted in several ways:
  - In the case of regression the mean of the values of the dependent variable is taken:

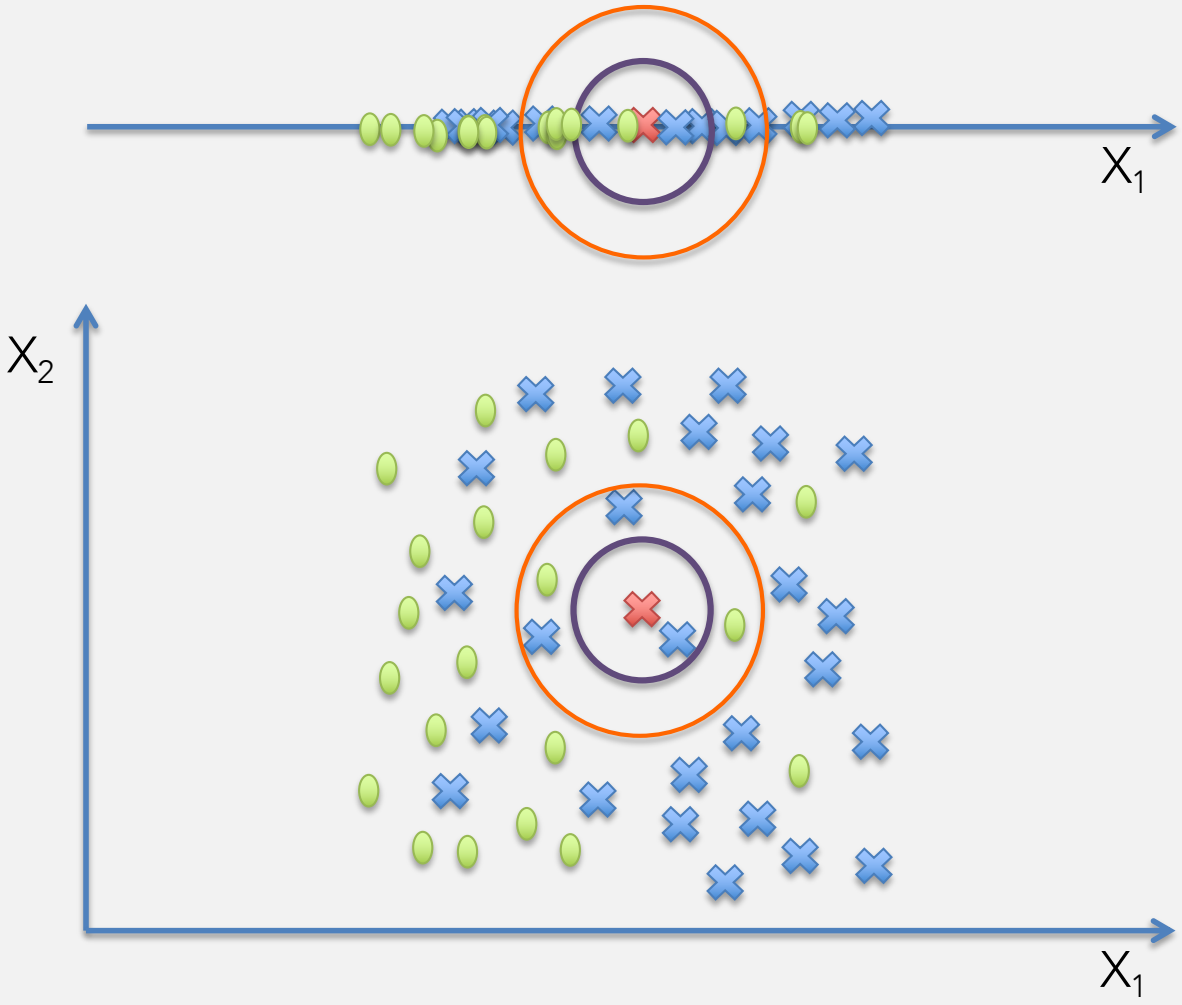
$$\hat{y}_i^{'} = \frac{1}{k}\sum_{j=1}^{k} y_i^j$$

  - In the case of classification the mode of the values of the dependent variable is taken:

$$\hat{y}_i^{'} = \mathrm{mode}(y_i^j)$$

PROBLEMS

1. *Dimensionality*: The first problem we encounter with using k-nn algorithm is that as dimensionality increases, it may be more difficult to find enough data points in a neighborhood



- This is called the *curse of dimensionality* which essentially says that as we increase the embedding space we need to sample more actively to obtain a sufficient number of data points.

- Note that when we increase e.g. the dimension form one to two we need to increase the sample not from n to $2n$ but to n^2.

2. *Calibration*: In k-nn we need to determine the number of neighbors to consider (k):

- Increasing k reduces variance
- Increasing k increases bias

- To select k we can use e.g. Cross-validation
- A popular heuristic rule is $k = \sqrt{n}$
- As we will see, compared to other ML algorithms the number of hyper-parameters in k-nn is the smallest (only ONE parameter).

3. *Data scaling*: Notice that if the features are expressed in very different scales then the distances might be biased, for this reason is common to re-scale the features using the methods seen before.

- For example, assume that we want to find the relationship between the price of a house and its size (measured in either squared meters or squared feet) and distance to a school (measured etither in meters or foot):

	measures 1		*measures 2*		*measures 3*	
	big	*school*	*big*	*school*	*big*	*school*
	ft	*m*	*m*	*m*	*ft*	*ft*
example 1	1500	1200	139	1200	1500	3937
example 2	2000	1600	186	1600	2000	5249
example 3	1750	1800	163	1800	1750	5906
example 4	1450	2000	135	2000	1450	6562
example 5	1650	1350	153	1350	1650	4429
mean	*1670*	*1590*	*155*	*1590*	*1670*	*5217*

- Notice that even though the relationship might be the same the unit measures bias the conclusions:

	measures 1			*measures 2*			*measures 3*		
	dif(big)2	*dif(school)2*	*dist1*	*dif(big)2*	*dif(school)2*	*dist2*	*dif(big)2*	*dif(school)2*	*dist3*
example1	250000	160000	**640**	2158	160000	**403**	250000	1722226	**1404**
example2	62500	360000	**650**	539	360000	**600**	62500	3875008	**1984**
example3	2500	640000	**802**	22	640000	**800**	2500	6888903	**2625**
example4	22500	22500	212	194	22500	**151**	22500	242188	**514**
			576			*488*			*1632*

- If we scale to [0,1] using the max-min trasformation:

	measures 1			*measures 2*			*measures 3*		
	dif(big)2	*dif(school)2*	*dist1*	*dif(big)2*	*dif(school)2*	*dist2*	*dif(big)2*	*dif(school)2*	*dist3*
example1	0.8264463	0.25	**1.0375**	0.8264	0.2500	**1.0375**	0.82645	0.2500	**1.0375**
example2	0.2066116	0.5625	**0.8770**	0.2066	0.5625	**0.8770**	0.20661	0.5625	**0.8770**
example3	0.0082645	1	**1.0041**	0.0083	1.0000	**1.0041**	0.00826	1.0000	**1.0041**
example4	0.0743802	0.0351563	0.3310	0.0744	0.0352	**0.3310**	0.07438	0.0352	**0.3310**
			0.8124			*0.8124*			*0.8124*

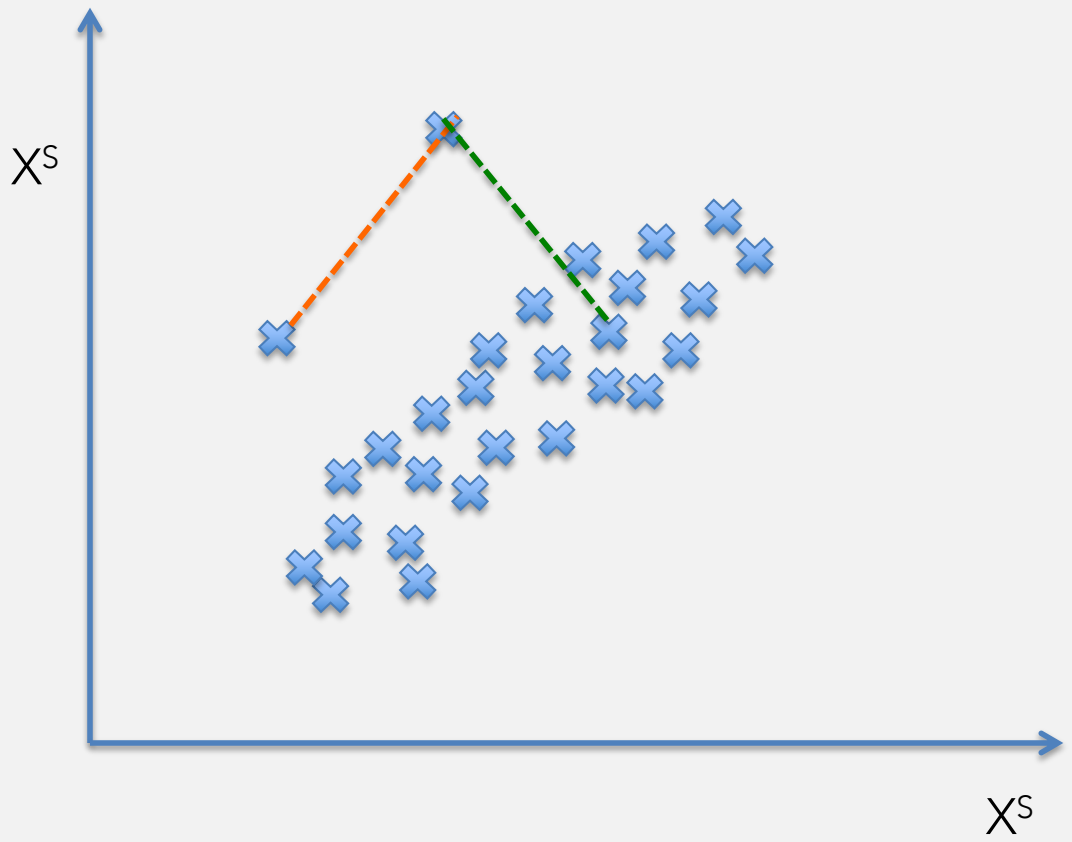
units.ipynb

4. *Feature independence*:

- Euclidean distance metric assumes that each feature is independent from others.
- But, in many cases, there exists some correlation between two variables or among several of them (for example weight and height, income and expense…).
- When correlation exists,the consecuence is that, in fact, we are "double counting" the correlated features

$$\rho(X^R, X^S) > 0$$

$$d(x_i, x_j) = (x_i^1 - x_j^1)^2 + \ldots + (x_i^R - x_j^R)^2 + \ldots + (x_i^S - x_j^S)^2 + \ldots + (x_i^D - x_j^D)^2$$



- One possibility is to employ some sort of feature selection, such as Principal Component Analysis

5. Feature importance:

- Euclidean distance metric assumes that each feature is equally important.
- This assumption may not be always satisfied in real applications, especially when dealing with high dimensional data where some features may not be tightly related to the topic of interest.
- A better distance metric should identify important features and discriminate relevant and irrelevant features.
- It is possible to derive also more sophisticated methods to weight or even select the features that enter into the computation of neighbors.
- Note that when the weight are either zero or one we are performing *feature selection*.

- When the importance of the features can be estimated, it is also common to calculate a weighted measure that takes into account the relative importance of each of the variables:

$$d_{euclidian}(x_i, x_j) = \sqrt{\sum_{k=1}^{D} w_k (x_i^k - x_j^k)^2}$$

- This procedure is called *feature weighting* (do not confuse with distance weighting, explained latter).

6. Outlier sensitivity

- Compared to other ML algorithms such as Classification adn Regression Trees, k-nn are highly sensitive to outlier, this is due to the fact that all the features are treated the same and the squared distance is higly distorted for big differences.

$$z - score(x_i^S) > 2$$

$$d(x_i, x_j) = (x_i^1 - x_j^1)^2 + \ldots + (x_i^S - x_j^S)^2 + \ldots + (x_i^D - x_j^D)^2$$

7. *System requirements*

- In contrast with lazy learning, *eager learning* uses the training data to construct a model which, in some sense, it comprises the examples used.

- After this model is found, he training data can be discarded since predictions are just made by recalling from the estimated model.

- Lazy algorithms such as have fewer computational costs than *eager algorithms* during training (in fact, there is NO training) but greater storage requirements and higher computational costs on recall.

- The computational cost of calculating the distance is O(nDk) wich can make it infeasible for very large datasets.

- Note also that knn is a high intensive memory method: we need to keep all the examples and distances in memory