Machine learning with Python

K-Nearest Neighbours

Outline

1. Instance based classification
2. The k-NN algorithm
3. Measuring similarity
4. Choosing k
5. Data preparation

Nearest neigbours

- Instance-based methods are the simplest form of learning
- Training instances are searched for instance that most closely resembles new instance
- The instances themselves represent the knowledge
- Similarity (distance) function defines what's "learned"
- "lazy evaluation", until a new instance must be classified
- Instance-based learning is lazy learning

Nearest neigbours

- Instance based learning - Store training examples and delay the processing until a new instance must be classified
- That is way it is often described as lazy learner
- Nearest neighbours are defined by their characteristic of classifying unlabeled examples by assigning them the class of similar labeled examples.
- Well-suited for classification tasks, where relationships among the features and the target classes are numerous, complicated, or extremely difficult to understand, yet the items of similar class type tend to be fairly homogeneous.
- If the data is noisy and thus no clear distinction exists among the groups, the nearest neighbor algorithms may struggle to identify the class boundaries.

Succesful applications

- Computer vision applications, optical character recognition and facial recognition
- Recommendation predictions
- Identifying patterns in genetic data

The k-nn algorithm

- The simplest example of nearest neighbours approach to classification
- The k-NN algorithm gets its name from the fact that it uses information about an example's k-nearest neighbors to classify unlabeled examples
- Algorithm steps
  1. Provide a training dataset made up of examples that have been classified into categories as labeled by a nominal variable
  2. For each unlabeled record in the test dataset do the following:
     1. Compute the distance between test example and every example from training set
     2. Select the k items from training set that are closest to the unlabeled record
     3. Assign unlabeled record to the class of the majority of the k nearest neighbors

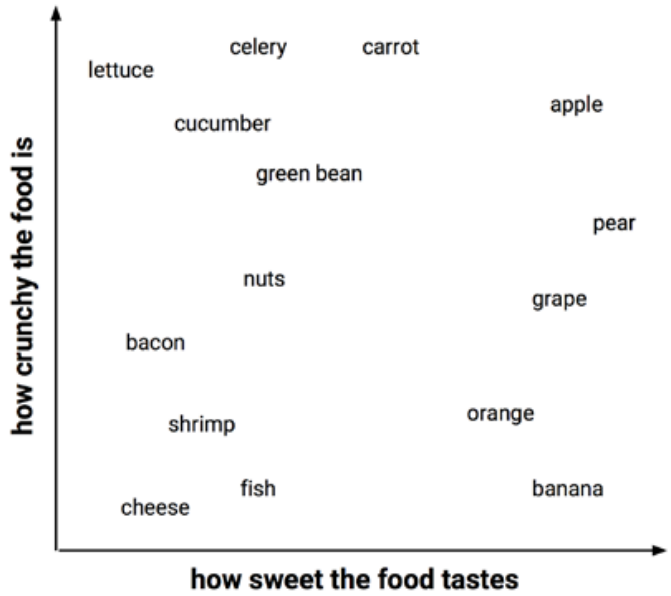
k-NN illustration

- We had created a dataset in which we recorded our impressions of a number of ingredients we tasted previously.
- Crunchiness - measure from 1 to 10 of how crunchy the ingredient is
- Sweetness - measure from 1 to 10 how sweet the ingredient tastes.
- each ingredient is labeled as one of the three types of food: fruits, vegetables, or proteins.

Ingredient	Sweetness	Crunchiness	Food type
apple	10	9	fruit
bacon	1	4	protein
banana	10	1	fruit
carrot	7	10	vegetable
celery	3	10	vegetable
cheese	1	1	protein

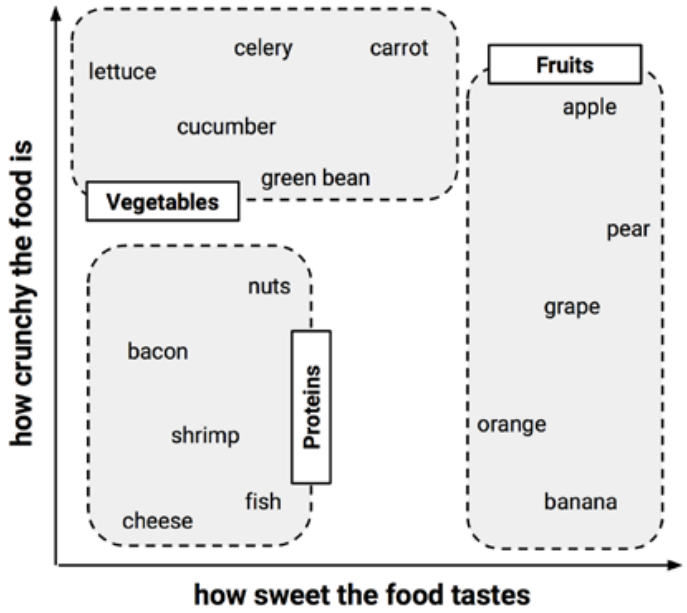
k-NN illustration

- The k-NN algorithm treats the features as coordinates in a multidimensional feature space.
- Data presented on a scatter plot, with the x dimension indicating the ingredient's sweetness and the y dimension, the crunchiness:



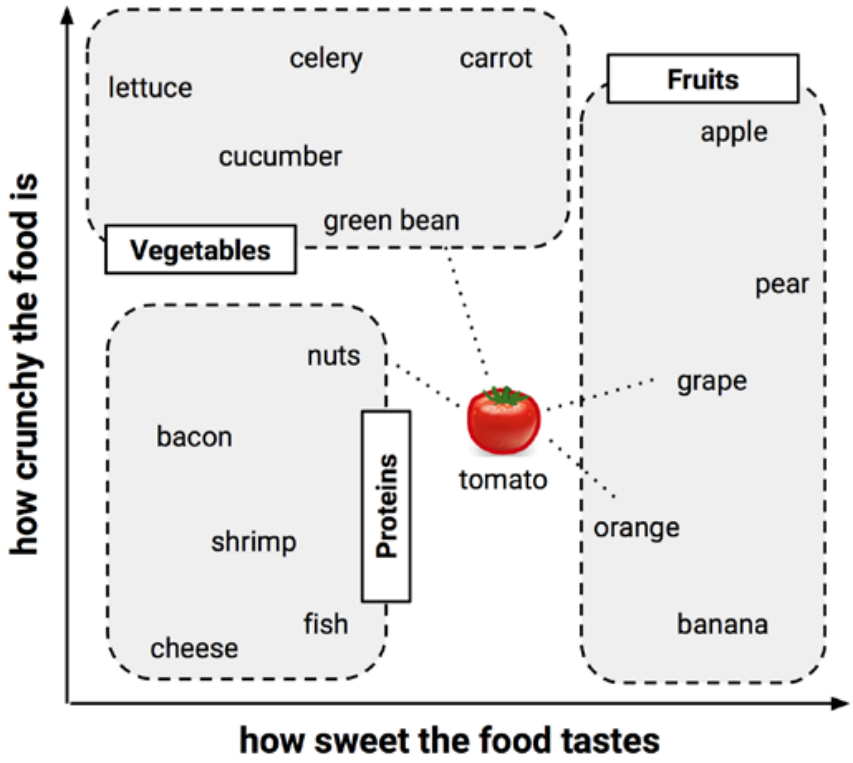
k-NN illustration

- Similar types of food tend to be grouped closely together.
- Vegetables tend to be crunchy but not sweet, fruits tend to be sweet and either crunchy or not crunchy, while proteins tend to be neither crunchy nor sweet:



k-NN illustration

- Is tomato a fruit or vegetable?
- Nearest neighbor approach can be used to determine which class is a better fit



To answer this question we need to locate the tomato's nearest neigbours

Measuring similarity

- Traditionally the k-NN algorithm uses Euclidean metric
- Euclidean metric can be described as the shortest direct route if it would be possible to connect two points with line.
- But there are many different distance metrics like Manhattan, Mahalanobis etc.
- Euclidean metric can be formulated as:

$$dist(X, Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_d - y_d)^2} \quad (1)$$

where X and Y are the instances to be compared based on d features

k-NN illustration

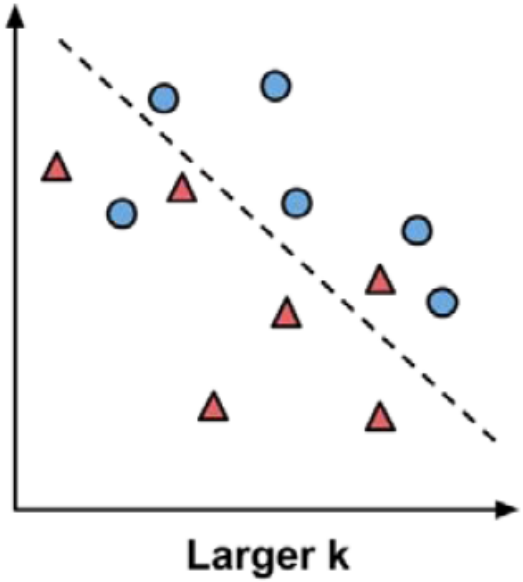
Ingredient	Sweetness	Crunchiness	Food type	Distance to the tomato
grape	8	5	fruit	*sqrt((6 - 8)^2 + (4 - 5)^2) = 2.2*
green bean	3	7	vegetable	*sqrt((6 - 3)^2 + (4 - 7)^2) = 4.2*
nuts	3	6	protein	*sqrt((6 - 3)^2 + (4 - 6)^2) = 3.6*
orange	7	3	fruit	*sqrt((6 - 7)^2 + (4 - 3)^2) = 1.4*

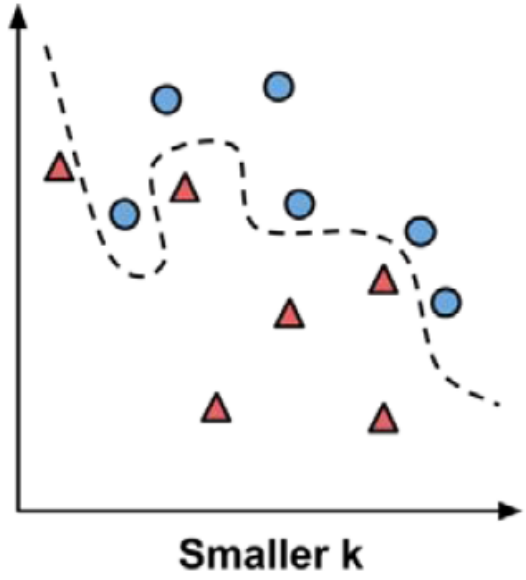
- Setting $k = 1$ classification is based on the single nearest neighbour and k-NN would classify tomato as a fruit.
- Setting $k = 3$ k-NN would classify tomato performing a vote among the three nearest neighbours.

Setting k

- How many neighbors to use for k-NN?
- We need to set k before we can run the algorithm
- k determines how well the model will generalize to future data.
- Decision about k is subject to bias-variance tradeoff
- Choosing a small k leads to more flexible classification but classifier is more sensitive to noise points
- Choosing a large k reduces the impact or variance caused by noisy data, but can bias the learner so that it runs the risk of ignoring small, but important patterns.

Decision boundary relation to k





- Smaller values allow more complex decision boundaries that more carefully fit the training data.
- We do not know whether the straight boundary or the curved boundary better represents the true underlying concept

Setting k

- In practice k depends on the difficulty of the concept to be learned, and the number of records in the training data.
- You can begin with k equal to the square root of the number of training examples but it may not provide the single best k
- The best way to set k would be to test several k values on a variety of test datasets and choose the one that delivers the best classification performance.
- In fact the best way to set number of k is to use k-fold Cross Validation method

Scaling issues

- Features are typically transformed to a standard range prior to applying the k-NN algorithm.
- The distance formula is highly dependent on how features are measured.
- If certain features have a much larger range of values than the others, the distance measurements will be strongly dominated by the features with larger ranges.
- The solution is to rescale the features by shrinking or expanding their range such that each one contributes relatively equally to the distance formula.

Scaling issues

- min-max normalization
- $$X_{nor} = \frac{X - min(X)}{max(X) - min(X)} \quad (2)$$
- The formula subtracts the minimum of feature X from each value and divides by the range of X.
- Normalized feature values indicate how far, from 0 percent to 100 percent, the original value fell along the range between the original minimum and maximum.

Scaling issues

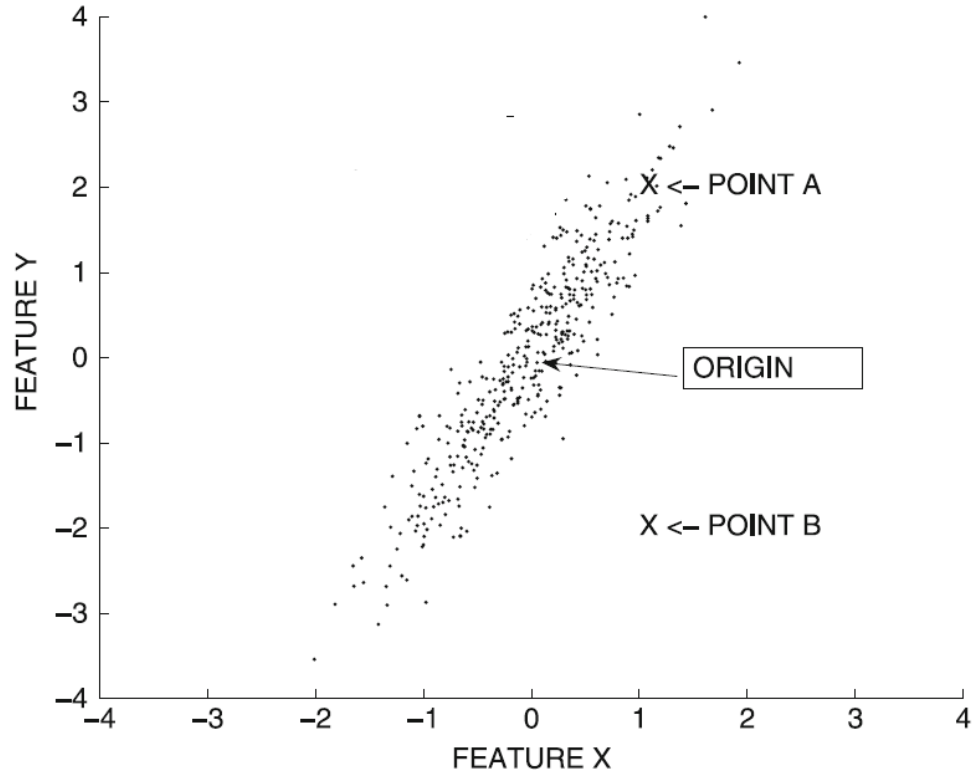
- z-score standardization
- $$X_{st} = \frac{X - mean(X)}{stdev(X)} \tag{3}$$
- Above formula subtracts the mean value of feature X, and divides the outcome by the standard deviation of X.
- z-score standarization is based on the properties of the normal distribution rescales each of the feature's values in terms of how many standard deviations they fall above or below the mean value.
- Unlike the normalized values, they have no predefined minimum and maximum.
-

k-NN with Euclidean distance

- For high dimensional data Euclidean measure can produce counter-intuitive resulst - curse of dimensionality
- This is because the Euclidean function is usually not the most effective distance metric in terms of its sensitivity to feature and class distribution
- With standarization all attributes are equally important
- Instead of using the Euclidean distance metric, the distance between two d-dimensional points Y and X is defined with respect to a $d \times d$ matrix A
- $$Dist((X,Y) = \sqrt{(X-Y)A(X-Y)^T} \tag{4}$$
- When A is the identity matrix distance function is the same as the Euclidean metric
- Different choices of A can lead to better sensitivity of the distance function to the local and global data distributions

Mahlanobis distance

- Should distances depend on the underlying data distribution of the remaining points in the data set?
- A and B are equidistant from the origin according to Euclidean Metric
- Statistically, it is much less likely for B to be so far away from O along this direction.



Mahlanobis distance

- Let Σ be $d \times d$ covariance matrix of the data set. In this case, the $(i, j)th$ entry of the covariance matrix is equal to the covariance between the dimensions i and j.
- The Mahlanobis distance is as follows:

$$Maha(X, Y) = \sqrt{(X - Y)\Sigma^{-1}(X - Y)^T} \tag{5}$$

- The Mahalanobis distance is similar to the Euclidean distance, except that it normalizes the data on the basis of the interattribute correlations.
- Adjusts well to the different scaling of the dimensions and the redundancies across different features.
- In case of noncorrelated features it auto scales data measured in different quantities
- In cases where the attributes are correlated, the Mahalanobis metric accounts well for the varying redundancies in different features

k-NN discussion

- Simple, often very accurate but slow
- Makes no assumption about underlying data distribution
- Does not produce a model, limiting the ability to understand how the features are related to the class
- Requires selection of an appropriate k
- Computationally heavy but some improvements are possible
- Nominal features and missing data require additional processing
- Need large number of samples for accuracy