Machine learning 1

KNN - k-nearest neighbours

Piotr Wójcik
pwojcik@wne.uw.edu.pl

academic year 2017/2018

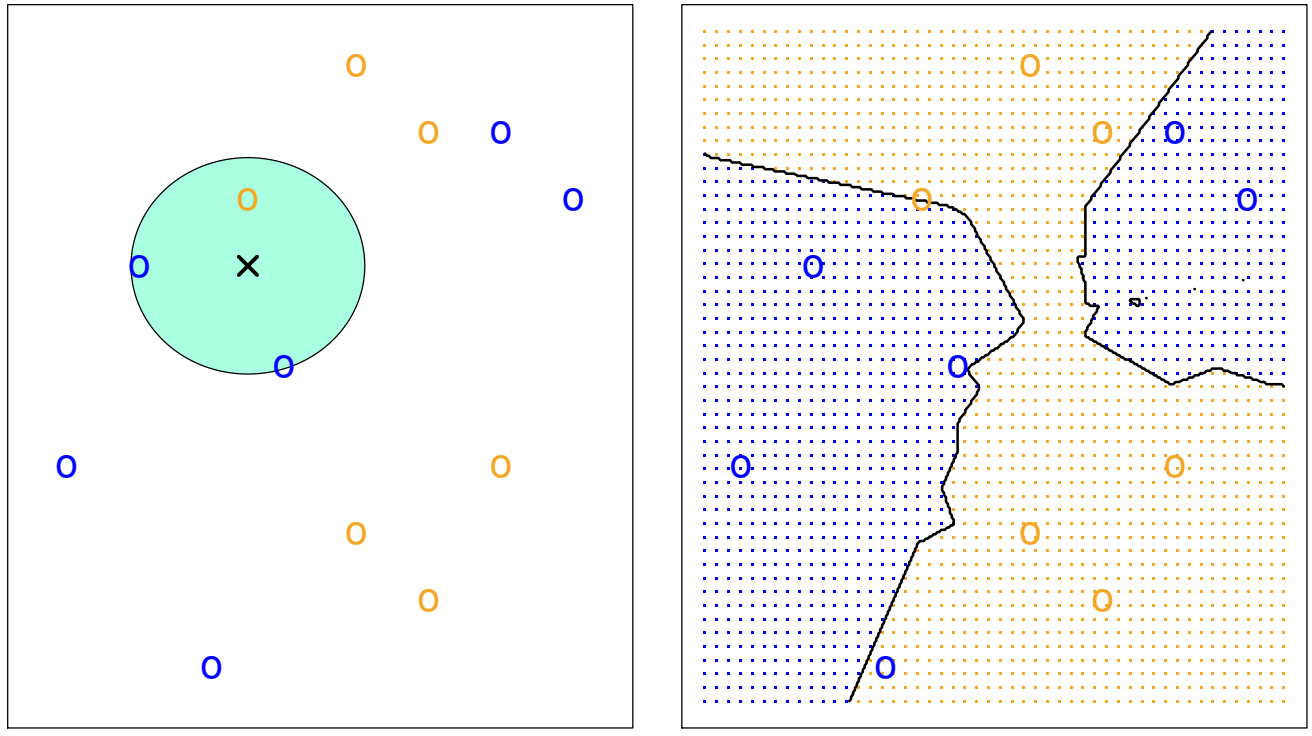
UNIWERSYTET
WARSZAWSKI

Uniwersytet Warszawski
Wydział Nauk Ekonomicznych

Classification with K-nearest neighbors (**KNN**)

- KNN was initially designed as a tool for **classification**
- for the natural number K and the observation from the test sample x_0, the KNN classifier identifies K points from the learning sample, which are located **closest to** x_0 – its **nearest neighbors**
- then checks which groups the selected observations from the learning sample belong to
- the new observation is classified into the group **most represented** among K neighbors
- **learning** of this model is therefore very fast – there is no estimation, optimization, etc. in this case.
- **predicting** based on the model on the test sample can be quite time-consuming in large samples

K-nearest neighbors – example



Source: James et al (2017), p. 40

K-nearest neighbors – example

- in the left picture we have a small training set consisting of six blue and six orange observations
- in turn, the black cross indicates a new observation, which should be classified
- suppose $K = 3$
- the KNN algorithm will first find three observations from the learning sample that are closest to the black cross
- the neighborhood is marked with a green circle: there are two blue points and one orange in it
- as the result a new point will be assigned to the blue class
- the right figure shows the use of $K = 3$ on a dense grid of points and the **decision area** of the classification for both groups is marked

KNN method – distance

- the KNN algorithm treats each variable as a **separate dimension of space** – taking into account p variables, we operate in the p-dimensional space
- there are many ways to measure the distance (similarity) of objects
- the most common method in the KNN method is the **Euclidean distance** – the length of the shortest segment connecting two points
- it is calculated as the square root of the sum of squares of differences corresponding to the coordinates of individual points
- for points i and j and p variables (dimensions) the Euclidean distance can be calculated as

$$d_e(i,j) = \sqrt{(x_{1i} - x_{1j})^2 + (x_{2i} - x_{2j})^2 + \ldots + (x_{pi} - x_{pj})^2}$$

KNN method – distance, cont'd

- alternatively one can use **city distance** (or Manhattan distance), which assumes that moving between points is possible only along the coordinate axes:

$$d_c(i,j) = |x_{1i} - x_{1j}| + |x_{2i} - x_{2j}| + \ldots + |x_{pi} - x_{pj}|$$

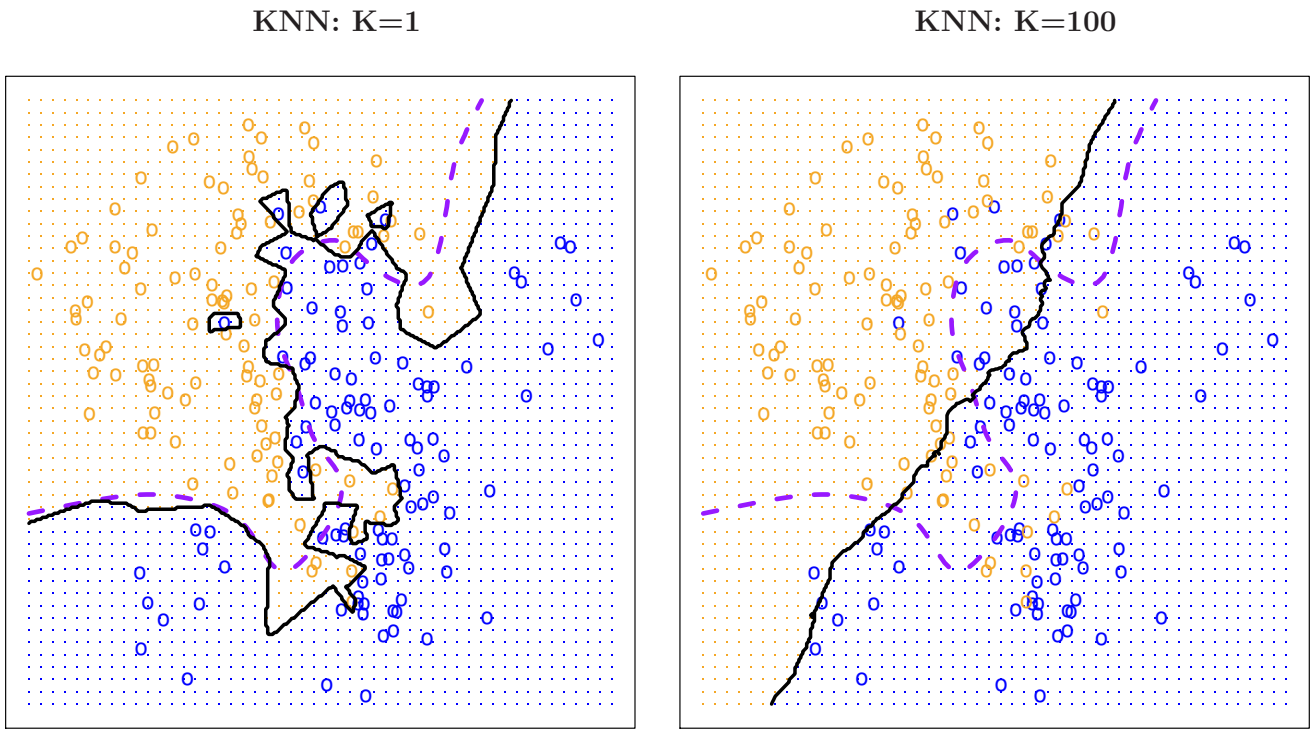
- both of the above mentioned measures can be treated as special cases of the **Mahalanobis distance**:

$$d_M(i,j,\lambda) = \left(|x_{1i} - x_{1j}|^\lambda + |x_{2i} - x_{2j}|^\lambda + \ldots + |x_{pi} - x_{pj}|^\lambda\right)^{1/\lambda}$$

Choosing the right value of K

- the choice of the value of the K parameter has a key impact on the classification results
- for $K = 1$ the method is **most flexible**, it fits very closely to the data – in such case it has **large variance** and **low bias**
- with the increase of K the flexibility of the method decreases, and the boundaries between groups become more and more linear
- the choice of a larger K **decreases the variance**, but raises the risk of obtaining biased model – ignoring small but distinct groups of observations in the data
- in extreme case, when K is equal to the number of observations in the data set, the model will **always predict the same class** – the one with the highest frequency

Choosing the right value of K – example



Source: James et al (2017), p. 41

Choosing the right value of K – cont'd

- one should always check **different values** of K and choose the one that gives the best model (e.g. **using cross-validation**)
- a good reference point is the value of K equal to the square root of the number of observations (if the dataset is not too big)
- in general, in large data sets the impact of K on the obtained result will be smaller, because even "small" distinct groups of observations will be quite numerous to find a sufficient number of neighbors

Preparing data for KNN

- features that have a much wider range of values than the others will **strongly dominate** in the calculation of distance (e.g. age in years, annual income in PLN, EUR or USD)
- therefore, the use of KNN classification usually requires **preparation of data**
- variables should be **standardized** to similar range or variability to have a **similar effect on distance measures** when choosing neighbors

Standardization of variables

- there are many methods of **standardization** of data
- generally the standardization of a variable refers to subtracting from the variable value the **measure of location** (L), and then dividing by the selected **measure of scale** (S):

$$X\_new = \frac{X - L}{S}$$

Standardization of variables – *z-score*

- the most popular method of standardization (called **z-score standardization**) is to bring the variable to a distribution with an average of 0 and a variance equal to 1
- this effect will be obtained by using the **sample mean** as a measure of location ($\bar{X}$), and **standard deviation** as a measure of scale (σ_X):

$$X\_new = \frac{X - \bar{X}}{\sigma_X}$$

- the value transformed in this way is often referred to as **z-score**

Standardization of variables – interval [0,1]

- Alternatively, one can scale the variable to take values from 0 to 1
- in this case one should use **minimum** of the variable value as the measure of location: $min(X)$, and **range** as a measure of scale: $max(X) - min(X)$:

$$X\_new = \frac{X - min(X)}{max(X) - min(X)}$$

- however, even after standardization features will not have equal impact on results, as their distributions will still differ (e.g. standard deviation, kurtosis)

Nominal features in the KNN method

- above mentioned distance measures are not defined for nominal variables
- before using them in the analysis one should convert them into numeric features – appropriate **dummy variables**
- for a nominal variable with two levels, one dummy variable is enough, while for a feature with m levels we will create $m - 1$ corresponding variables
- **NOTE!** if one considers nominal features recoded to dummies in KNN, **range standardization** is a better method of standardization of continuous variables
- then **all variables** used in the analysis have the same range [0,1]

Ordinal features in the KNN method

- in the case of **ordinal features** alternatively, they can be coded as consecutive numerical values and standardized, similarly as quantitative features
- such encoding assumes, however, **equal distances** between individual levels of the ordinal variable, which is usually not appropriate
- therefore a safer approach in case of ordinal features is to use a **similar procedure as for nominal variables** – recoding to dummies

The pros and cons of the KNN algorithm

Advantages:

- simple and efficient
- does not require assumptions regarding distributions of the analyzed variables
- fast on the model training stage

Disadvantages:

- does not result in a model, which limits the understanding of how individual features affect the allocation of observations to groups
- requires choosing the appropriate value of the K parameter
- time-consuming at the stage of classification (prediction)
- nominal features and missing data require additional steps
- it is very difficult to compare observations (correctly identify neighbours) in multidimensional space – **curse of dimendionality**

Summary

- Despite its radical simplicity, the KNN algorithm works well in many applications
- it is successfully used for:
  - recognition of text or faces – both on static photos and in video films
  - building recommendation systems recommending books, films, music
  - identifying patterns in genetic data associated with various diseases
- KNN method gives good results in classification problems, where the function f is very complicated, based on many features, and at the same time units from individual classes are quite homogeneous
- however, if individual groups are not well separated, the KNN algorithm may not give satisfactory results

Extensions

- KNN algorithm mighy also be used in **regression**. In this case value of the dependent variable might be predicted for example as:
  - average of the numerical target of the K nearest neighbors
  - inverse distance weighted average of the K nearest neighbors
- **random KNN** – combination of base k-nearest neighbor models, each constructed from a random subset of input variables – can be applied both to classification and regression problems