Data Analysis and Knowledge Discovery

K-Nearest Neighbour (K-NN) method

prof. Jukka Heikkonen

University of Turku
Department of Information Technology

Outline

Outline

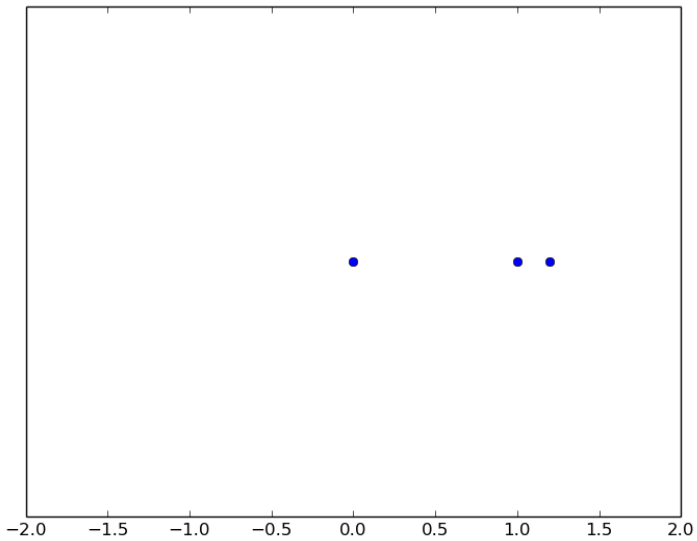
Feature space

- Let us assume, that each instance in our data is represented by a d-dimensional feature vector
- $\mathbf{x} = [x_1, x_2, ..., x_d]$
- each of the d numerical values thus corresponds to one recorded property of the instance
- e.g. if instances persons, then x_1 age, x_2 height etc.
- assumption: data pre-processed, categorical values converted to numerical ones, missing values imputed (simplest method: insert the mean value for that feature in the data matrix)
- usually also z-score standardization done, so that all the values roughly in the same range

Feature space

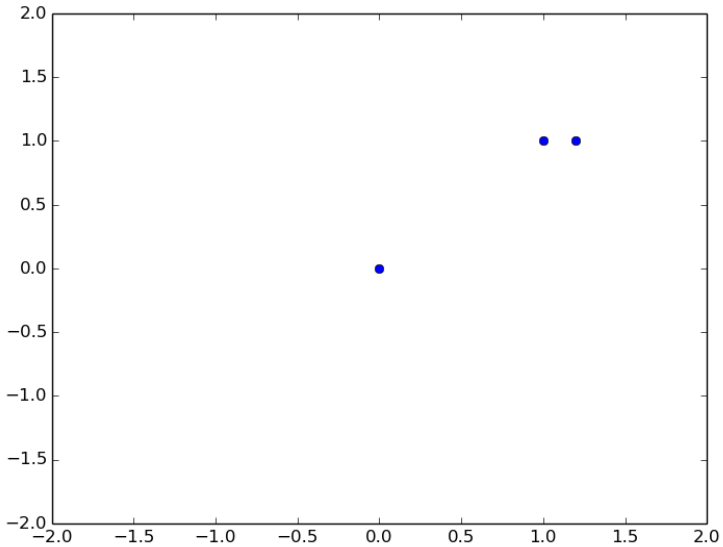
- Example: let our data set consist of 3 instances, represented by the following 4-dimensional feature vectors
- $\mathbf{x} = [1, 1, 0, 0.5]$
- $\mathbf{y} = [0, 0, 0, 1]$
- $\mathbf{z} = [1.2, 1, 0, -1]$
- The instances can be thought of as points in a 4-dimensional space, where each axis corresponds to one of the features
- Learning algorithms can be defined in terms of geometry in this space
- What is the distance between any given two instances?

Feature space



- Let us consider a 1D-projection of our data to the first coordinate
- $\mathbf{x} = [1]$, $\mathbf{y} = [0]$, $\mathbf{z} = [1.2]$
- Euclidean distance: $d(\mathbf{x}, \mathbf{y}) = |1 - 0| = 1$
- Similarly $d(\mathbf{x}, \mathbf{z}) = |1 - 1.2| = 0.2$
- Since $d(\mathbf{x}, \mathbf{z}) < d(\mathbf{x}, \mathbf{y})$ we may say that $\mathbf{z}$ is closer to $\mathbf{x}$ than $\mathbf{y}$

Feature space



- A 2D-projection of our data to the first two coordinates
- $\mathbf{x} = [1, 1]$, $\mathbf{y} = [0, 0]$, $\mathbf{z} = [1.2, 1]$
- Euclidean distance: $d(\mathbf{x}, \mathbf{y}) = \sqrt{(1-0)^2 + (1-0)^2} = \sqrt{2}$
- Similarly $d(\mathbf{x}, \mathbf{z}) = \sqrt{(1-1.2)^2 + (1-1)^2} = 0.2$

Feature space

- Basic geometric concepts generalize to higher dimensions
- (Euclidean) distance: $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{d} (\mathbf{x}_i - \mathbf{y}_i)^2}$
- Inner (dot) product: $\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{d} \mathbf{x}_i \mathbf{y}_i$
- When $\mathbf{x}$ and $\mathbf{y}$ considered column vectors, may equivalently be written as the matrix product $\mathbf{x}^T \mathbf{y}$
- In addition to standard Euclidean distance measure, also many others can be defined

Nearest neighbour predictors

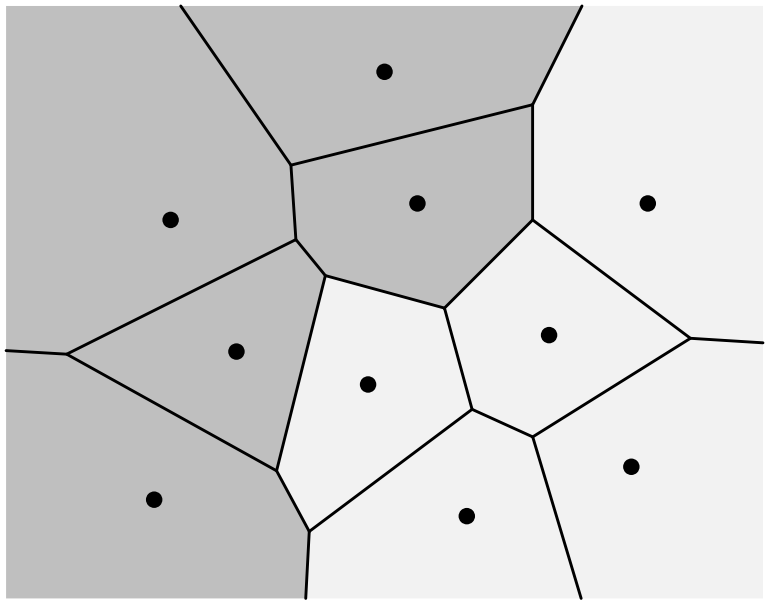
- Nearest neighbour predictors are special case of instance-based learning.
- Basic idea: similar instances should have similar class-labels or real-valued labels (classification / regression)
- Similarity defined in terms of distance: the smaller the distance between two feature vectors the more similar the corresponding instances are considered

Nearest neighbour predictors

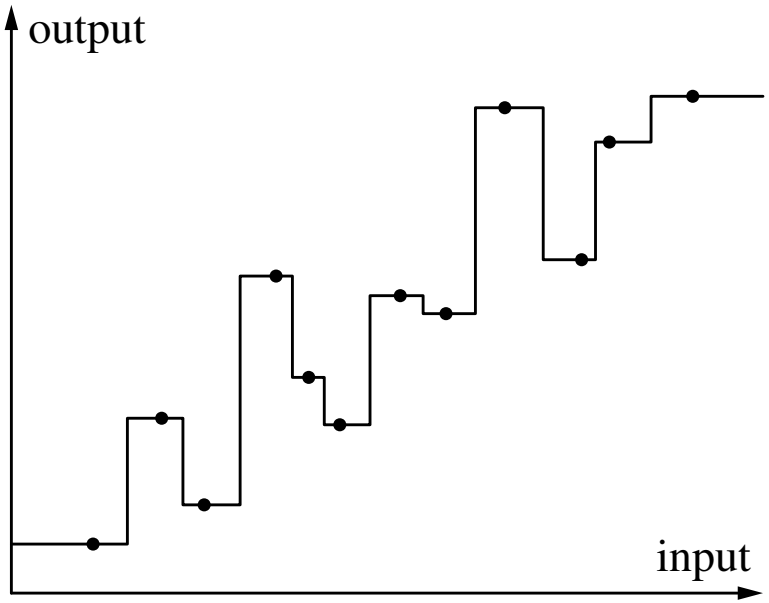
- As simple as it gets: memorize your training data, for new instances predict the majority class (or average value in regression) of its k-nearest neighbours
- Neghbours: k training instances having the smallest distance from the new instance
- No learning algorithm needed. We do not explicitly learn a model out of the training data, the data is the model.
- Surprisingly powerful, though has its limitations
- Choices needed: value of k, distance measure

Simple nearest neighbour predictor

For a new instance, use the target value of the closest neighbour in the training set.



Classification



Regression

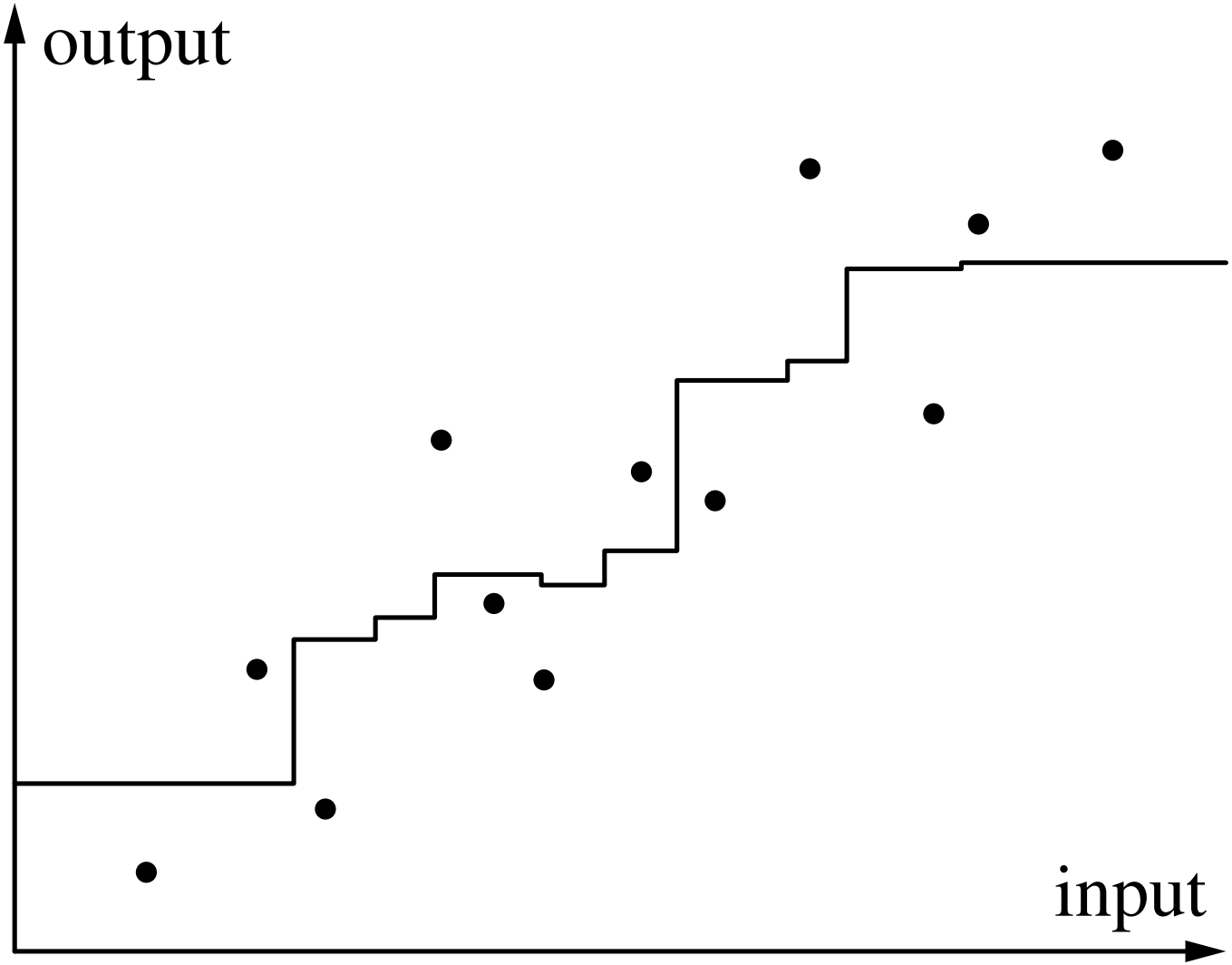
k-nearest neighbour predictor

- Instead of relying for the prediction on only one instance, the (single) nearest neighbour, usually the k ($k > 1$) are taken into account, leading to the k-nearest neighbour predictor.
- Classification: Choose the majority class among the k nearest neighbours for prediction.
- Regression: Take the mean value of the k nearest neighbours for prediction.

Disadvantage:

- All k nearest neighbours have the same influence on the prediction.
- Closer nearest neighbours should (perhaps) have a higher influence on the prediction.

Nearest neighbour predictor



Average (3 nearest neighbours)

Ingredients for the k-nearest neighbour predictor

- **distance metric**
  The distance metric, together with a possible task-specific scaling or weighting of the attributes, determines which of the training examples are nearest to a query data point and thus selects the training example(s) used to compute a prediction.
- **number of neighbours**
  The number of neighbours of the query point that are considered can range from only one (the basic nearest neighbour approach) through a few (like k-nearest neighbour approaches) to, in principle, all data points as an extreme case.

Ingredients for the k-nearest neighbour predictor

- **weighting function for the neighbours**
  Weighting function defined on the distance of a neighbour from the query point, which yields higher values for smaller distances.
- **prediction function**
  For multiple neighbours, one needs a procedure to compute the prediction from the (generally differing) classes or target values of these neighbours, since they may differ and thus may not yield a unique prediction directly.

Distance

Choosing the "ingredients"

- **distance metric**
  Problem dependent. Often Euclidean distance (after normalisation).
- **number of neighbours**
  Very often chosen on the basis of cross-validation (cross-validation will be introduced later).
- **weighting function for the neighbours**
  Simplest choice uniform weighting. May also use (a function of) inverse of the distance for weighting.
- **prediction function**

Nearest neighbour predictor

Choosing the "ingredients"

- **prediction function**

  Regression. Compute the weighted average of the target values of the nearest neighbours.

  Classification. Sum up the weights for each class among the nearest neighbours. Choose the class with the highest value. If tie among several classes, choose one for example randomly.

Adjusting the distance function

- The choice of the distance function is crucial for the success of a nearest neighbour approach.
- In addition to standard Euclidean distance a large number of other possible distance measures (Manhattan, Mahalanobis, Chebyshev, Hamming...)
- If your data is not numerical, also distances defined between strings, graphs, sets exist
- One can also try do adapt the distance function.
- One way to adapt the distance function is feature weights to put a stronger emphasis on those features that are considered more important, either based on prior knowledge or experiments.

Advantages of k-nearest neighbors

- Easy to implement, no time for training needed (though in practice you need to choose features, select k, distance measure...)
- Can easily be adapted to different types of data by choosing suitable distance measure
- Can model very complex non-linear problems in areas where basic linear models fail (e.g. optical character recognition...)
- Assuming large and representative enough training data set, will often yield good predictive accuracy

Disadvantages of k-nearest neighbors

- Prediction can be very slow for large data sets, since need to find the k-nearest neighbors. Also memory usage can be an issue, since whole training set needs to be loaded.
  - Search can be accelerated using advanced data structures, as long as the dimensionality not too large
- Having many irrelevant features problematic, since these may dominate the distances drowning out the relevant ones
- The method tends to give poor results on very high-dimensional data
  - Feature selection, dimensionality reduction may be helpful
- Black box model, offers no insight about patterns underlying the data

Applying the k-nearest neighbor method

- Few lines of code to implement, if you use standard choices like Euclidean distance, uniform weighting
- Implement function predict(x, k), that finds k-nearest neighbors of x, then returns the majority class (or mean value for regression) among them
- More complicated to implement, if you want to optimize the search, allow different distance measures etc.
- sklearn (machine learning in python), see: `http://scikit-learn.org/stable/modules/neighbors.html`

Conclusions

- K-nearest neighbors, your first classification/regression method
- Simple yet powerful method, freely available implementations
- Consider data normalization, feature selection, dimensionality reduction as pre-processing
- Open questions:
  - How to choose parameters such as the value of k, or the distance measure?
  - How to reliably evaluate, how well the model predicts?
  - Next time: answer to these questions (and not only for k-nn)