K-Nearest Neighbour Classifier

Izabela Moise, Evangelos Pournaras, Dirk Helbing
Reminder

Supervised data mining
✓ Classification
→ Decision Trees
K-Nearest Neighbour (kNN) Classifier
Classification steps

1. **Training** phase: a model is constructed from the training instances.
   → classification algorithm finds relationships between predictors and targets
   → relationships are summarised in a *model*

2. **Testing** phase: test the model on a test sample whose class labels are known but not used for training the model

3. **Usage** phase: use the model for classification on new data whose class labels are unknown
Instance-based Classification

Main idea:

Similar instances have similar classification

- no clear separation between the three phases of classification
- also called lazy classification, as opposed to eager classification
Eager vs Lazy Classification

Eager

- Model is computed **before** classification
- Model is **independent** of the test instance
- Test instance **is not** included in the training data
- Avoids too much work at classification time
- Model is not accurate for each instance

Lazy

- Model is computed **during** classification
- Model is **dependent** on the test instance
- Test instance **is** included in the training data
- High accuracy for models at each instance level
k-Nearest Neighbor (kNN)

Learning by analogy:

Tell me who your friends are and I’ll tell you who you are

→ an instance is assigned to the **most common** class among the instances **similar** to it

1. how to measure similarity between instances
2. how to choose the most common class
How does it work?

1. Initialization, define k
2. Compute distance (test instance, each training instance)
3. Sort the distances
4. Take k nearest neighbors
5. Apply simple majority

Class
How does it work?

1. Initialization, define $k$
2. Compute distance (test instance, each training instance)
3. Sort the distances
4. Take $k$ nearest neighbors
5. Apply simple majority
6. Class
Comparing Objects

→ **Problem**: measure similarity between instances
- different types of data: numbers, colours, geolocation, booleans etc.

✓ **Solution**: convert all features of the instances into numerical values
- represent instances as vectors of features in an n-dimensional space
Comparing Objects

→ **Problem**: measure similarity between instances

vs. text similarity

- different types of data: numbers, colours, geolocation, booleans etc.

✓ **Solution**: convert all features of the instances into numerical values

- represent instances as vectors of features in an n-dimensional space
Comparing Objects

→ Problem: measure similarity between instances vs. text similarity

• different types of data: numbers, colours, geolocation, booleans etc.

✓ Solution: convert all features of the instances into numerical values

• represent instances as vectors of features in an n-dimensional space
Comparing Objects

→ **Problem**: measure similarity between instances vs. text similarity

- different types of data: numbers, colours, geolocation, booleans etc.

✓ **Solution**: convert all features of the instances into numerical values

- represent instances as vectors of features in an n-dimensional space
Comparing Objects

→ **Problem**: measure similarity between instances vs. text similarity

- different types of data: numbers, colours, geolocation, booleans etc.

 ✓ **Solution**: convert all features of the instances into numerical values

- represent instances as vectors of features in an n-dimensional space
An Example:

- **Closeness** is defined in terms of the *Euclidean* distance between two examples.
  - The Euclidean distance between $X = (x_1, x_2, x_3, \ldots, x_n)$ and $Y = (y_1, y_2, y_3, \ldots, y_n)$ is defined as:
    \[ D(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \]

- Distance (John, Rachel) = \[\sqrt{((35-41)^2 + (95-215K)^2 + (3-2)^2)}\]
Distance Metrics

1. Euclidean Distance

\[ D(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \]

2. Manhattan Distance

\[ D = \sum_{i=1}^{n} |x_i - y_i| \]

3. Minkowski Distance

\[ D = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \]
Choosing $k$

- Classification is sensitive to the correct selection of $k$
- If $k$ is too small $\Rightarrow$ overfitting
  - Algorithm performs too good on the training set, compared to its true performance on unseen test data

small $k$?
larger $k$?
Choosing k

- Classification is sensitive to the correct selection of k
- if k is too small $\Rightarrow$ overfitting
  $\Rightarrow$ algorithm performs too good on the training set, compared to its true performance on unseen test data

small k? $\Rightarrow$ less stable, influenced by noise
larger k? $\Rightarrow$ less precise, higher bias
Choosing k

- Classification is sensitive to the correct selection of k
- if k is too small $\Rightarrow$ overfitting
  - algorithm performs too good on the training set, compared to its true performance on unseen test data

small k? $\rightarrow$ less stable, influenced by noise
larger k? $\rightarrow$ less precise, higher bias

$k = \sqrt[n]{n}$
Pros and Cons

Pros:

✓ simple to implement and use
✓ robust to noisy data by averaging k-nearest neighbours
✓ kNN classification is based solely on local information
✓ the decision boundaries can be of arbitrary shapes
Pros and Cons

Cons:

× curse of dimensionality: distance can be dominated by irrelevant attributes
× $O(n)$ for each instance to be classified
× more expensive to classify a new instance than with a model