Clustering

Connectionist and Statistical Language Processing

Frank Keller
keller@coli.uni-sb.de

Computerlinguistik
Universität des Saarlandes

Overview

- clustering vs. classification
- supervised vs. unsupervised learning
- types of clustering algorithms
- the *k*-means algorithm
- applying clustering to word similarity
- evaluating clustering models

Literature: Witten and Frank (2000: ch. 6), Manning and Schütze (1999: ch. 14)

Clustering vs. Classification

Classification: the task is to learn to assign instances to predefined classes.

Clustering: no predefined classification is required. The task is to learn a classification from the data.

Clustering algorithms divide a data set into *natural groups* (clusters). Instances in the same cluster are *similar* to each other, they share certain properties.

Supervised vs. Unsupervised Learning

Supervised learning: classification requires supervised learning, i.e., the training data has to specify what we are trying to learn (the classes).

Unsupervised learning: clustering is an unsupervised task, i.e., the training data doesn't specify what we are trying to learn (the clusters).

Example: to learn the parts of speech in a supervised fashion we need a predefined inventory of POS (a tagset) and an annotated corpus.

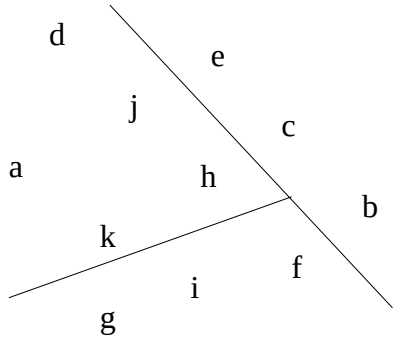
To learn POS in an unsupervised fashion, we only need an unannotated corpus. The task is to discover a set of POS.

Clustering Algorithms

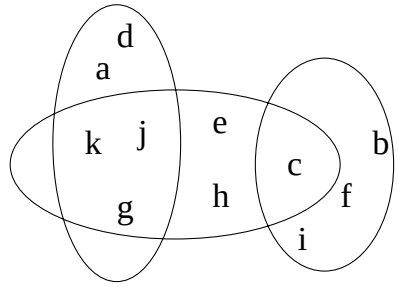
Clustering algorithms can have different properties:

- *Hierarchical or flat:* hierarchical algorithms induce a hierarchy of clusters of decreasing generality, for flat algorithms, all clusters are the same.
- *Iterative:* the algorithm starts with initial set of clusters and improves them by reassigning instances to clusters.
- *Hard and soft:* hard clustering assigns each instance to exactly one cluster. Soft clustering assigns each instance a probability of belonging to a cluster.
- *Disjunctive:* instances can be part of more than one cluster.

Examples

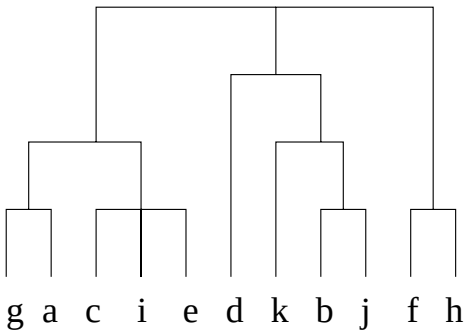


Hard, non-hierarchical, non-disjunctive



Hard, non-hierarchical, disjunctive

Examples



Hard, hierarchical, non-disjunctive

	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1

Soft, non-hierarchical, disjunctive

Using Clustering

Clustering can be used for:

- *Exploratory data analysis:* visualize the data at hand, get a feeling for what the data look like, what its properties are. First step in building a model.
- *Generalization:* discover instances that are similar to each other, and hence can be handled in the same way.

Example for generalization: *Thursday* and *Friday* co-occur with the preposition *on* in our data, but *Monday* doesn't.

If we know that *Thursday*, *Friday*, and *Monday* are syntactically similar, then we can infer that they all co-occur with *on*.

Properties of Clustering Algorithms

Hierarchical clustering:

- Preferable for detailed data analysis.
- Provides more information than flat clustering.
- No single best algorithm exists (different algorithms are optimal for different applications).
- Less efficient than flat clustering.

Properties of Clustering Algorithms

Flat clustering:

- Preferable if efficiency is important or for large data sets.
- k-means is conceptually the most simple method, should be used first on new data, results are often sufficient.
- k-means assumes a Euclidian representation space; inappropriate for nominal data.
- Alternative: EM algorithm, uses definition of clusters based on probabilistic models.

The k-means Algorithm

Iterative, hard, flat clustering algorithm based on Euclidian distance. Intuitive formulation:

- Specify k, the number of clusters to be generated.
- Chose k points at random as cluster centers.
- Assign each instance to its closest cluster center using Euclidian distance.
- Calculate the centroid (mean) for each cluster, use it as new cluster center.
- Reassign all instances to the closest cluster center.
- Iterate until the cluster centers don't change any more.

The k-means Algorithm

Each instance $\vec{x}$ in the training set can be represented as a vector of n values, one for each attribute:

$$\vec{x} = (x_1, x_2, \ldots, x_n) \quad (1)$$

The Euclidian distance of two vectors $\vec{x}$ and $\vec{y}$ is defined as:

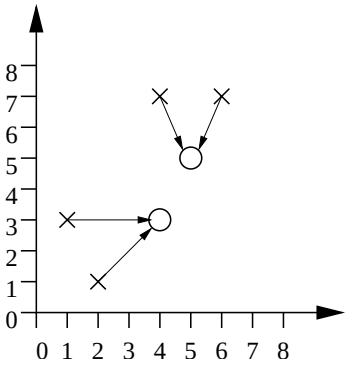
$$|\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (2)$$

The mean $\vec{\mu}$ of a set of vectors c_j is defined as:

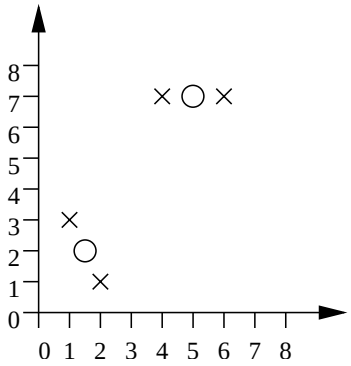
$$\vec{\mu} = \frac{1}{|c_j|} \sum_{\vec{x} \in c_j} \vec{x} \quad (3)$$

The k-means Algorithm

Example:



Assign instances (crosses) to closest cluster centers (circles) according to Euclidian distance



Recompute cluster centers as means of the instances in each cluster

The k-means Algorithm

Properties of the algorithm:

- It only finds a *local maximum*, not a global one.
- The clusters it comes up with depend a lot on which *random cluster centers* are chose initially.
- Can be used for *hierarchical clustering*: first apply k-means with $k = 2$, yielding two clusters. Then apply it again on each of the two clusters, etc.
- *Distance metrics* other than Euclidian distance can be used, e.g., the cosine.

Applications of Clustering

Task: discover contextually similar words

First determine a set of *context words*, e.g., the 500 most frequent words in the corpus.

Then determine a *window size*, e.g., 10 words (5 to the left and 5 to the right).

For each word in the corpus, compute a *co-occurrence vector* that specifies how often the word co-occurs with the context words in the given window.

Use clustering to find words with similar context vectors. This can find words that are *syntactically or semantically* similar, depending on parameters (context words, window size).

Applications of Clustering

Example (Manning and Schütze 1999: ch. 8):

Use adjectives (rows) as context words to cluster nouns (columns) according to semantic similarity.

	cosmonaut	astronaut	moon	car	truck
Soviet	1	0	0	1	1
American	0	1	0	1	1
spacewalking	1	1	0	0	0
red	0	0	0	1	1
full	0	0	1	0	0
old	0	0	0	1	1

Cosmonaut and *astronaut* are similar, as are *car* and *truck*.

Evaluating Clustering Models

As with all machine learning algorithms, set the model parameters on the training set. Then test its performance on the test set, keeping the parameters constant.

Training set: set of instances used to generate the clusters (compute the cluster centers in the case of k-means).

Test set: a set of unseen instances classified using to the clusters generated during training.

Problem: How do we determine k, i.e., the number of clusters? (Many algorithms require a fixed k, not only k-means.)

Solution: treat k as a parameter, i.e., vary k and find the best performance on a given data set.

Evaluating Clustering Models

Problem: How do we evaluate the performance on the test set? How do we know if the clusters are correct? Possible solutions:

- Test the resulting clusters *intuitively*, i.e., inspect them and see if they make sense. Not advisable.
- Have an *expert* generate clusters manually, and test the automatically generated ones against them.
- Test the clusters against a predefined *classification* if there is one.
- Perform *task-based evaluation*, i.e., test if the performance of some other algorithm can be improved by using the output of the clusterer.

Evaluating Clustering Models

Example: Use a clustering algorithm to discover parts of speech in a set of word. The algorithm should group together words with the same syntactic category.

Intuitive: check if the words in the same cluster seem to have the same part of speech.

Expert: ask a linguist to group the words in the data set according to POS. Compare the result to the automatically generated clusters.

Classification: look up the words in a dictionary. Test if words in the same cluster have the same POS in the dictionary.

Task: use the clusters for parsing (for example) and test if the performance of the parser improves.

Summary

Clustering algorithms *discover* groups of similar instances, instead of requiring a predefined classification. They are *unsupervised*.

Depending on the application, *hierarchical* or *flat*, and *hard* or *soft* clustering is appropriate.

The *k-means algorithm* assigns instances to clusters according to Euclidian distance to the cluster centers. Then it recomputes cluster centers as the means of the instances in the cluster.

Clusters can be evaluated against an external *classification* (expert-generated or predefined) or *task-based*.

References

Manning, Christopher D., and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing.* Cambridge, MA: MIT Press.

Witten, Ian H., and Eibe Frank. 2000. *Data Mining: Practical Machine Learing Tools and Techniques with Java Implementations.* San Diego, CA: Morgan Kaufmann.