Text Classification and Clustering

[DAT640] Information Retrieval and Text Mining

Ivica Kostric & Krisztian Balog
University of Stavanger

August 24, 2022

In this module

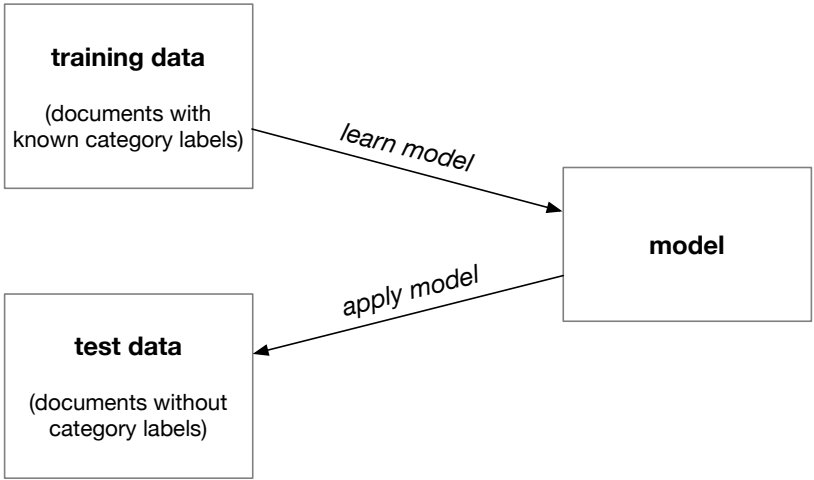
1. Text classification
2. Naive Bayes text classifier
3. Text classification evaluation
4. Text clustering

Text classification

Text classification

- **Classification** is the problem of assigning objects to one of several predefined categories
  - One of the fundamental problems in machine learning, where it is performed the basis of a training dataset (instances whose category membership is known)
- In **text classification** (or **text categorization**) the objects are text documents
- Binary classification (two classes, 0/1 or -/+)
  - E.g., deciding whether an email is spam or not
- Multiclass classification (n classes)
  - E.g., Categorizing news stories into topics (finance, weather, politics, sports, etc.)

General approach



Formally

- Given a training sample $(\boldsymbol{X}, y)$, where $\boldsymbol{X}$ is a set of documents with corresponding labels y, from a set $\boldsymbol{Y}$ of possible labels, the task is to learn a function $f(\cdot)$ that can predict the class $y' = f(x)$ for an unseen document x.

Families of approaches

- **Feature-based approaches** (“traditional” machine learning)
- Neural approaches (“deep learning”)

Question

What could be used as features in text classification?

Features for text classification

- Use words as features (**bag-of-words**)
  - Words will be referred to as **terms**
- Values can be, e.g., binary (term presence/absence) or integers (term counts)
- Documents are represented by their **term vector**
- **Document-term matrix** is huge, but most of the values are zeros; stored as a sparse matrix

	t_1	t_2	t_3	$\ldots$	t_m
d_1	1	0	2		0
d_2	0	1	0		2
d_3	0	0	1		0
$\ldots$					
d_n	0	1	0		0

Document-term matrix

Question

Is it possible to use other, non-term-frequency-based features?

Additional features for text classification

- Descriptive statistics (avg. sentence length, length of various document fields, like title, abstract, body,...)
- Document source
- Document quality indicators (e.g., readability level)
- Presence of images/attachments/JavaScript/...
- Publication date
- Language
- ...

Naive Bayes text classifier

Naive Bayes

- Example of a generative classifier
- Estimating the probability of document $\boldsymbol{x}$ belonging to class y

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

- $P(\boldsymbol{x}|y)$ is the class-conditional probability
- $P(y)$ is the prior probability
- $P(\boldsymbol{x})$ is the evidence (note: it's the same for all classes)

Naive Bayes classifier

- Estimating the class-conditional probability $P(y|\boldsymbol{x})$
  - $\boldsymbol{x}$ is a vector of term frequencies $\{x_1, \ldots, x_n\}$

$$P(\boldsymbol{x}|y) = P(x_1, \ldots, x_n|y)$$

- "Naive" assumption: features (terms) are independent:

$$P(\boldsymbol{x}|y) = \prod_{i=1}^{n} P(x_i|y)$$

- Putting our choices together, the probability that $\boldsymbol{x}$ belongs to class y is estimated using:

$$P(y|\boldsymbol{x}) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Estimating prior class probabilities

- $P(y)$ is the probability of each class label
- It is essential when class labels are imbalanced

Estimating feature distribution

- How to estimate $P(x_i|y)$?
- Maximum likelihood estimation: count the number of times a term occurs in a class divided by its total number of occurrences

$$P(x_i|y) = \frac{c_{i,y}}{c_i}$$

  - $c_{i,y}$ is the number of times term x_i appears in class y
  - c_i is the total number of times term x_i appears in the collection
- But what happens if $c_{i,y}$ is zero?!

Smoothing

- Ensure that $P(x_i|y)$ is never zero
- Simplest solution:[1] Laplace ("add one") smoothing

$$P(x_i|y) = \frac{c_{i,y} + 1}{c_i + m}$$

  - m is the number of classes

[1] More advanced smoothing methods will follow later for Language Modeling

Practical considerations

- In practice, probabilities are small, and multiplying them may result in numerical underflows
- Instead, we perform the computations in the log domain

$$\log P(y|\boldsymbol{x}) \propto \log P(y) + \sum_{i=1}^{n} \log P(x_i|y)$$

Exercise

E2-1 Naive Bayes

Text classification evaluation

Evaluation

- Measuring the performance of a classifier
  - Comparing the predicted label y' against the true label y for each document in some set dataset
- Based on the number of records (documents) correctly and incorrectly predicted by the model
- Counts are tabulated in a table called the **confusion matrix**
- Compute various **performance measures** based on this matrix

Text classification evaluation

- Evaluating binary classification
- Evaluating multiclass classification
- Model development

Confusion matrix

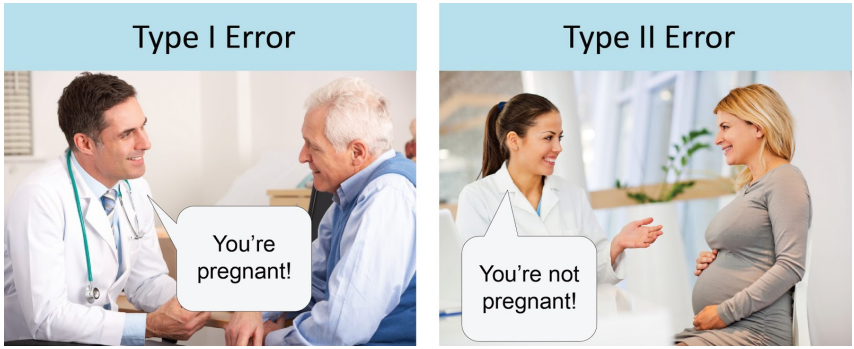
		Predicted class	
		negative	positive
Actual class	negative	true negatives (TN)	false positives (FP)
	positive	false negatives (FN)	true positives (TP)

- False positives = Type I error ("raising a false alarm")
- False negatives = Type II error ("failing to raise an alarm")

Question

Which of Type I and Type II error is worse?

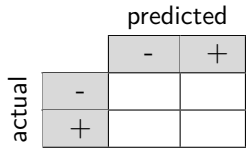
Type I vs. Type II errors[2]



[2]Source: https://www.analyticsindiamag.com/understanding-type-i-and-type-ii-errors/

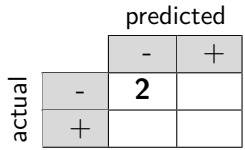
Example

Id	Actual	Predicted
1	+	-
2	+	+
3	-	-
4	+	+
5	+	-
6	+	+
7	-	-
8	-	+
9	+	-
10	+	-



Example

Id	Actual	Predicted
1	+	-
2	+	+
3	-	-
4	+	+
5	+	-
6	+	+
7	-	-
8	-	+
9	+	-
10	+	-



Example

Id	Actual	Predicted
1	+	-
2	+	+
3	-	-
4	+	+
5	+	-
6	+	+
7	-	-
8	-	+
9	+	-
10	+	-

		predicted	
		-	+
actual	-	2	**1**
	+		

Example

Id	Actual	Predicted
1	+	-
2	+	+
3	-	-
4	+	+
5	+	-
6	+	+
7	-	-
8	-	+
9	+	-
10	+	-

		predicted	
		-	+
actual	-	2	1
	+	4	3

Evaluation measures

- Summarizing performance in a single number
- **Accuracy**
  Fraction of correctly classified items out of all items

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Error rate**
  Fraction of incorrectly classified items out of all items

$$ERR = \frac{FP + FN}{FP + FN + TP + TN}$$

		predicted -	predicted +
actual	-	TN	FP
	+	FN	TP

Evaluation measures (2)

- **Precision**
  Fraction of items correctly identified as positive out of the total items identified as positive

$$P = \frac{TP}{TP + FP}$$

- **Recall** (also called Sensitivity or True Positive Rate)
  Fraction of items correctly identified as positive out of the total actual positives

$$R = \frac{TP}{TP + FN}$$

actual \ predicted	-	+
-	TN	FP
+	FN	TP

Evaluation measures (3)

- **F1-score**
  The harmonic mean of precision and recall

$$F1 = \frac{2 \cdot P \cdot R}{P + R}$$

		predicted -	predicted +
actual	-	TN	FP
	+	FN	TP

Evaluation measures (4)

- **False Positive Rate (Type I Error)**
  Fraction of items wrongly identified as positive out of the total actual negatives

$$FPR = \frac{FP}{FP + TN}$$

- **False Negative Rate (Type II Error)**
  Fraction of items wrongly identified as negative out of the total actual positives

$$FNR = \frac{FN}{FN + TP}$$

		predicted -	predicted +
actual	-	TN	FP
	+	FN	TP

Example

actual \ predicted	-	+
-	TN=2	FP=1
+	FN=4	TP=3

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = \frac{5}{10} = 0.5$$

$$P = \frac{TP}{TP + FP} = \frac{3}{4} = 0.75$$

$$R = \frac{TP}{TP + FN} = \frac{3}{7} = 0.429$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R} = \frac{2 \cdot 3/4 \cdot 3/7}{3/4 + 3/7} = 0.545$$

Text classification evaluation

- Evaluating binary classification
- Evaluating multiclass classification
- Model development

Multiclass classification

- Imagine that you need to automatically sort news stories according to their topical categories

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

Table: Categories in the 20-Newsgroups dataset

Multiclass classification

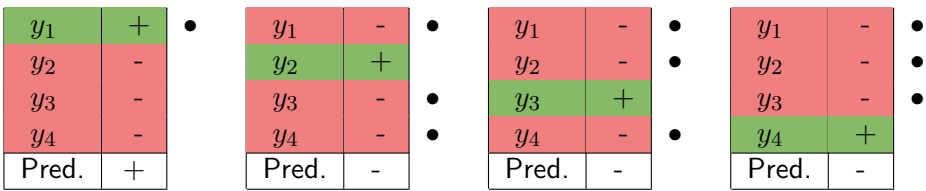
- Many classification algorithms are originally designed for binary classification
- Two main strategies for applying binary classification approaches to the multiclass case
  - One-against-rest
  - One-against-one
- Both apply a voting scheme to combine predictions
  - A tie-breaking procedure is needed (not detailed here)

One-against-rest

- Assume there are k possible target classes $(y_1, \ldots, y_k)$
- Train a classifier for each target class y_i $(i \in [1..k])$
  - Instances that belong to y_i are positive examples
  - All other instances y_j, $j \neq i$ are negative examples
- Combining predictions
  - If an instance is classified positive, the positive class gets a vote
  - If an instance is classified negative, all classes except for the positive class receive a vote

Example

- 4 classes (y_1, y_2, y_3, y_4)
- Classifying a given test instance (dots indicate the votes cast):



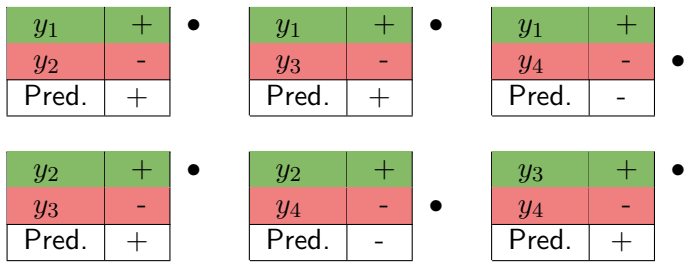
- Sum votes received: (y_1,●●●●), (y_2,●●), (y_3,●●), (y_4,●●)

One-against-one

- Assume there are k possible target classes $(y_1, \ldots, y_k)$
- Construct a binary classifier for each pair of classes (y_i, y_j)
  - $\frac{k \cdot (k-1)}{2}$ binary classifiers in total
- Combining predictions
  - The predicted class receives a vote in each pairwise comparison

Example

- 4 classes (y_1, y_2, y_3, y_4)
- Classifying a given test instance (dots indicate the votes cast):



- Sum votes received: $(y_1$,••), $(y_2$,•), $(y_3$,•), $(y_4$,••)

Question

How to evaluate multiclass classification?
Which of the evaluation measures from binary classification can be applied?

Evaluating multiclass classification

- Accuracy can still be computed as

$$ACC = \frac{\#\text{correctly classified instances}}{\#\text{total number of instances}}$$

- For other metrics
  - View it as a set of k binary classification problems (k is the number of classes)
  - Create confusion matrix for each class by evaluating "one against the rest"
  - Average over all classes

Confusion matrix

Actual \ Predicted	1	2	3	...	k
1	**24**	0	2		0
2	0	**10**	1		1
3	1	0	**9**		0
...					
k	2	0	1		**30**

Binary confusion matrices, one-against-rest

Actual \ Predicted	1	2	3	...	k
1	**24**	0	2		0
2	0	**10**	1		1
3	1	0	**9**		0
...					
k	2	0	1		**30**

$\Rightarrow$

Act. \ Predicted	1	$\neg 1$
1	TP=24	FN=3
$\neg 1$	FP=2	TN=52

Act. \ Predicted	2	$\neg 2$
2	TP=10	FN=2
$\neg 2$	FP=0	TN=69

...

For the sake of this illustration, we assume that the cells which are not shown are all zeros.

Averaging over classes

- Averaging can be performed on the instance level or on the class level
- **Micro-averaging** aggregates the results of individual instances across all classes
  - All instances are treated equal
- **Macro-averaging** computes the measure independently for each class and then take the average
  - All classes are treated equal

Micro-averaging

- **Precision**

$$P_\mu = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} (TP_i + FP_i)}$$

- **Recall**

$$R_\mu = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{k} (TP_i + FN_i)}$$

- **F1-score**

$$F1_\mu = \frac{2 \cdot P_\mu \cdot R_\mu}{P_\mu + R_\mu}$$

		predicted	
		i	$\neg i$
actual	i	TP_i	FN_i
	$\neg i$	FP_i	TN_i

Macro-averaging

- **Precision**

$$P_M = \frac{\sum_{i=1}^{k} \frac{TP_i}{TP_i + FP_i}}{k}$$

- **Recall**

$$R_M = \frac{\sum_{i=1}^{k} \frac{TP_i}{TP_i + FN_i}}{k}$$

- **F1-score**

$$F1_M = \frac{\sum_{i=1}^{k} \frac{2 \cdot P_i \cdot R_i}{P_i + R_i}}{k}$$

  - where P_i and R_i are Precision and Recall, respectively, for class i

		predicted	
		i	$\neg i$
actual	i	TP_i	FN_i
	$\neg i$	FP_i	TN_i

Text classification evaluation

- Evaluating binary classification
- Evaluating multiclass classification
- Model development

Question

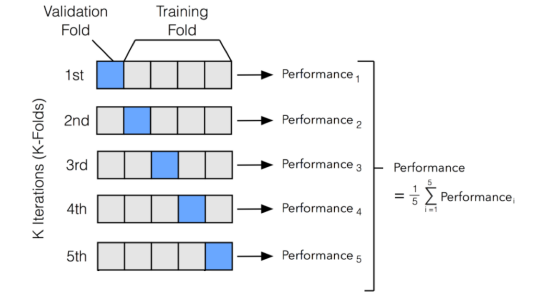
How can we evaluate the performance of the model during development?

Using a validation set

- Idea: hold out part of the training data for testing into a **validation set**
- **Single train/validation split**
  - Split the training data into $X\%$ training split and $100 - X\%$ validation split (an $80/20$ split is common)

Using a validation set[3]

- **k-fold cross-validation**
  - Partition the training data randomly into k folds
  - Use $k-1$ folds for training and test on the kth fold; repeat k times (each fold is used for testing exactly once)
  - k is typically 5 or 10
  - Extreme: k is the number of data points, to maximize the number of training material available (called "leave-one-out" evaluation)



[3] Image source:
http://ethen8181.github.io/machine-learning/model_selection/model_selection.html

Exercise

E2-2 Cross-validation

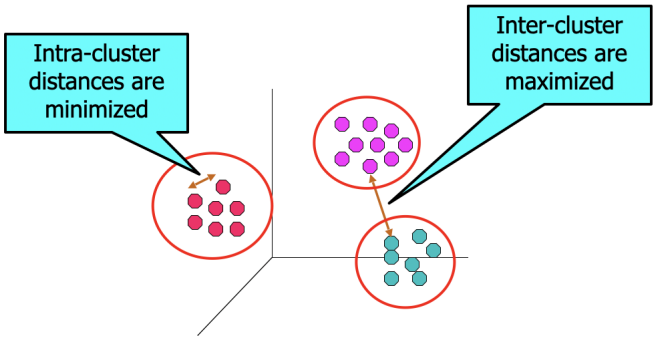
Text clustering

Clustering

- **Clustering** is concerned with the task of grouping similar objects together
  - Objects can be documents, sentences, words, users, etc.
- It is a general data mining technique for exploring large datasets
  - Clustering can reveal natural semantic structures
  - Can also help to navigate data, discover redundant content, etc.
- Clustering is regarded as an unsupervised learning problem

Clustering

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



Question

How many clusters should be formed?

Original data

The notion of a cluster can be ambiguous

Original data

Two Clusters

Four Clusters

Six Clusters

Types of clustering

- Partitional vs. hierarchical
  - Partitional: non-overlapping clusters such that each data object is in exactly one cluster
  - Hierarchical: a set of nested clusters organized as a hierarchical tree
- Exclusive vs. non-exclusive
  - Whether objects may belong to a single or multiple clusters
- Partial versus complete
  - In some cases, we only want to cluster some of the data
- Hard vs. soft
  - In hard clustering each object can only belong to a single cluster
  - In soft (or "fuzzy") clustering, an object belongs to every cluster with some probability

Text clustering

- Clustering algorithms
- Agglomerative Hierarchical Clustering
- K-means clustering

Clustering techniques

- **Similarity-based clustering**: require a *similarity function* to work. Each object can only belong to one cluster (*hard clustering*).
  - **Agglomerative clustering** (also called **hierarchical clustering**): gradually merge similar objects to generate clusters ("bottom-up")
  - **Divisive clustering**: gradually divide the whole set into smaller clusters ("top-down")
- **Model-based techniques**: rely on a probabilistic model to capture the latent structure of data
  - Typically, this is an example of *soft clustering*, since one object may be in multiple clusters (with some probability)

Similarity-based clustering

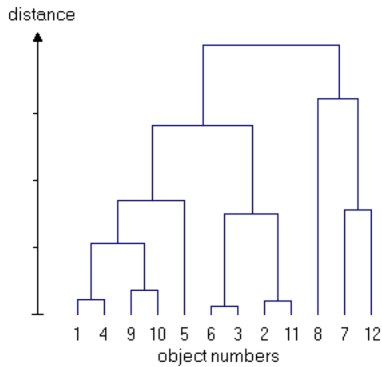
- Both agglomerative and divisive clustering methods require a document-document **similarity measure**, $sim(d_1, d_2)$
- In particular, the similarity measure needs to be
  - **symmetric**: $sim(d_1, d_2) = sim(d_2, d_1)$
  - **normalized**: $sim(d_1, d_2) \in [0, 1]$
- The choice of similarity measure is closely tied with how documents are represented

Text clustering

- Clustering algorithms
- Agglomerative Hierarchical Clustering
- K-means clustering

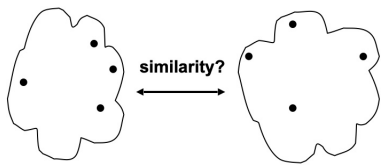
Agglomerative Hierarchical Clustering

- Progressively construct clusters to generate a hierarchy of merged groups ("bottom-up")
- Start with each document being a cluster on its own, and gradually merge clusters into larger and larger groups until there is only one cluster left
- This series of merges forms a **dendrogram**
- The tree may then be segmented based on how many clusters are needed
  - Alternatively, the merging may be stopped when the desired number of clusters is found



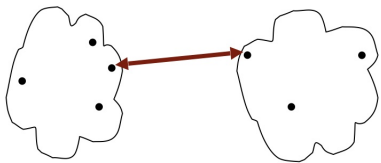
Measuring inter-cluster similarity

- Single-link
- Complete-link
- Average-link
- Prototype-based (centroid)



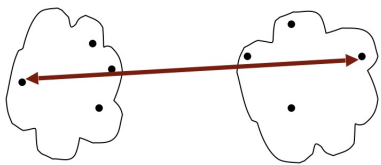
Single-link (“min”)

- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Results in “looser” clusters



Complete-link (“max”)

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Results in “tight” and “compact” clusters (tends to break large clusters)

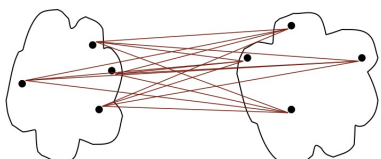


Average-link (“avg”)

- Similarity of two clusters is the average of pairwise similarity between points in the two clusters

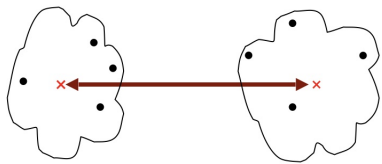
$$sim(C_i, C_j) = \frac{\sum_{x \in C_i, y \in C_j} sim(x, y)}{|C_i| \times |C_j|}$$

  - Less susceptible to noise and outliers than single- and complete-link



Prototype-based (centroid)

- Represent clusters by their centroids and base their similarity on the similarity of the centroids
  - To find the centroid, one computes the (arithmetic) mean of the points' positions separately for each dimension



Text clustering

- Clustering algorithms
- Agglomerative Hierarchical Clustering
- K-means clustering

K-means clustering

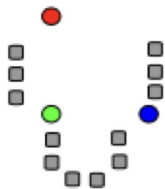
- Divisive clustering
- Start with an initial tentative clustering and iteratively improve it until we reach some stopping criterion
- It's a particular manifestation of the Expectation-Maximization algorithmic paradigm
- A cluster is represented with a **centroid**: representing all other objects in the cluster, usually as an average of all its members' values
- Finds a user-specified number of clusters (K)

Basic K-means algorithm

1. Select K points as initial centroids
2. **Repeat**
   - 2.1 Form K clusters by assigning each point to its closest centroid
   - 2.2 Recompute the centroid of each cluster
3. **Until** centroids do not change

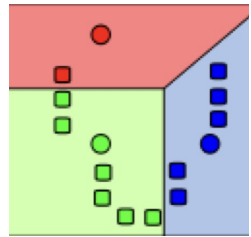
Basic K-means algorithm

1. Select K points as initial centroids $\Leftarrow$
2. **Repeat**
   - 2.1 Form K clusters by assigning each point to its closest centroid
   - 2.2 Recompute the centroid of each cluster
3. **Until** centroids do not change



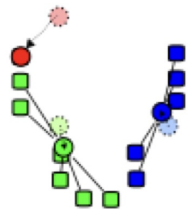
Basic K-means algorithm

1. Select K points as initial centroids
2. **Repeat**
   - 2.1 Form K clusters by assigning each point to its closest centroid ⇐
   - 2.2 Recompute the centroid of each cluster
3. **Until** centroids do not change



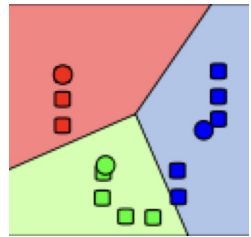
Basic K-means algorithm

1. Select K points as initial centroids
2. **Repeat**
   - 2.1 Form K clusters by assigning each point to its closest centroid
   - 2.2 Recompute the centroid of each cluster $\Leftarrow$
3. **Until** centroids do not change



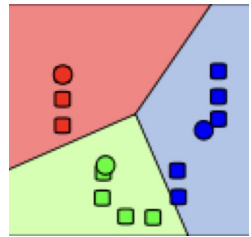
Basic K-means algorithm

1. Select K points as initial centroids
2. Repeat $\Leftarrow$
   - 2.1 Form K clusters by assigning each point to its closest centroid
   - 2.2 Recompute the centroid of each cluster
3. **Until** centroids do not change



Basic K-means algorithm

1. Select K points as initial centroids
2. **Repeat**
   - 2.1 Form K clusters by assigning each point to its closest centroid
   - 2.2 Recompute the centroid of each cluster
3. **Until centroids do not change** $\Leftarrow$



Exercise

E2-3 Text classification sklearn

Summary

- Problem of text classification (binary and multiclass variants)
- Feature-bases text classifiers (bag-of-words representation, document-term matrix)
- Naive Bayes classifier
- Evaluation (confusion matrix, binary/multiclass)
- Evaluation measures (accuracy, precision, recall, F1, micro- and macro-averaging)
- Training/test splits, cross-validation
- Problem of (text) clustering
- Similarity-based clustering algorithms (Agglomerative Hierarchical Clustering and K-means)

Reading

- Text Data Management and Analysis (Zhai&Massung)
  - Chapter 14 (Sections 14.1, 14.2)
  - Chapter 15 (Sections 15.1–15.4, 15.5.2, 15.6)