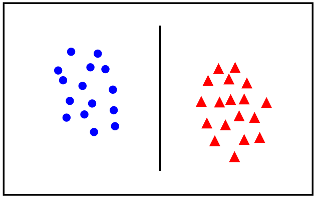
Learning from Data
Lecture 3: Support Vector Machines
Unsupervised learning: Clustering

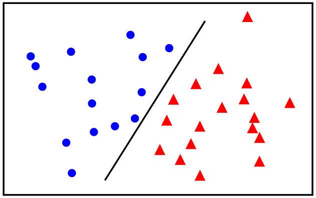
Malvina Nissim
m.nissim@rug.nl
room 1311.421

28 November 2016

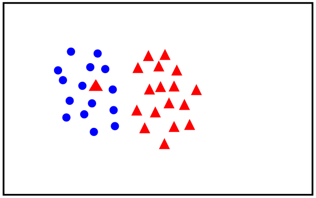
Linear separability

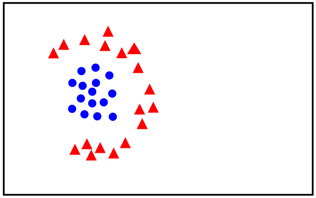
linearly
separable





not
linearly
separable

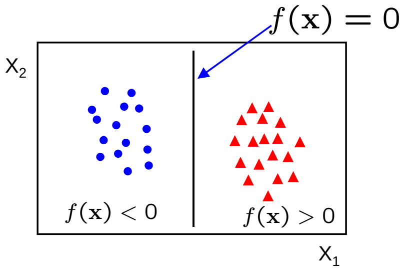




Linear classifiers

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$

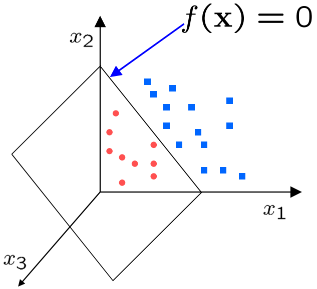


- in 2D the discriminant is a line
- $\mathbf{w}$ is the normal to the line, and b the bias
- $\mathbf{w}$ is known as the weight vector

Linear classifiers

A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$$



- in 3D the discriminant is a plane, and in nD it is a hyperplane

For a K-NN classifier it was necessary to `carry' the training data

For a linear classifier, the training data is used to learn **w** and then discarded

Only **w** is needed for classifying new data

Learning process

- the goal of the learning process is to come up with a “good” weight vector w
- the learning process will use examples to guide the search of a “good” w
- different notions of “goodness” exist, which yield different learning algorithms

Support Vector Machines

(mainly based on slides by Glenn Fung, O.L.Mangasarian, Rob Lothian, Kristin Bennett)

Support Vector Machines

vector space based machine-learning method aiming to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)

Support Vector Machines

vector space based machine-learning method aiming to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)

Support Vector Machines

vector space based machine-learning method aiming to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)

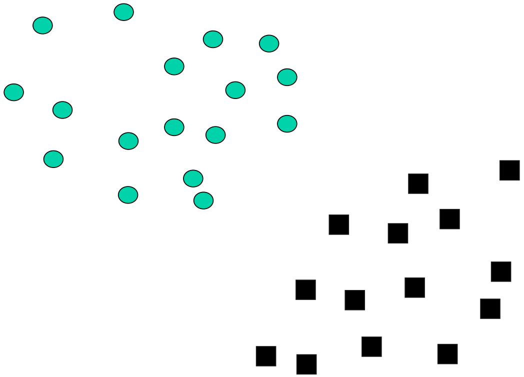
Support Vector Machines

vector space based machine-learning method aiming to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)

Support Vector Machines

vector space based machine-learning method aiming to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)

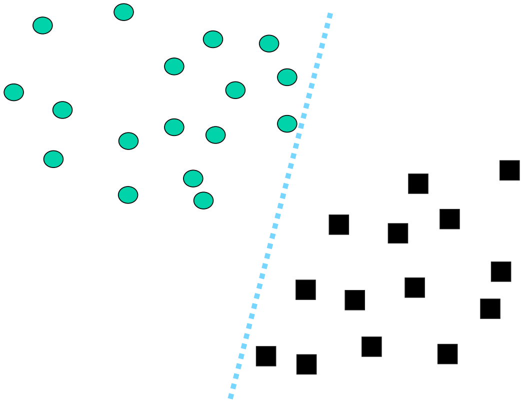
Which of the linear separators is optimal?



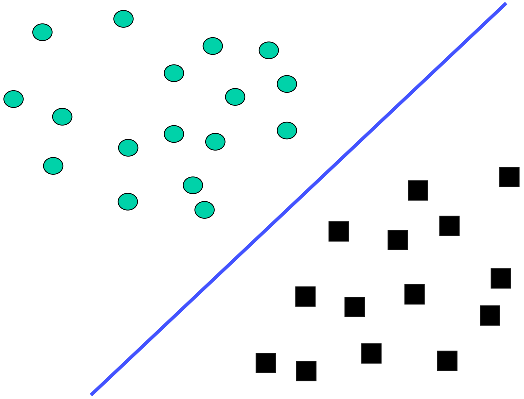
One intuition:

the fewer points near the decision surface, the fewer very uncertain classification decisions

Best Linear Separator?

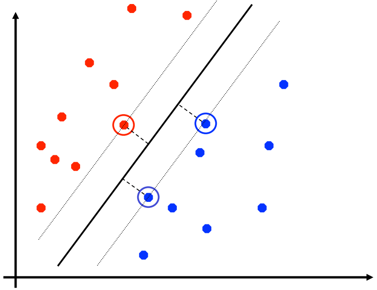


Best Linear Separator?



Maximum Margin Classification

- Maximizing the margin is good according to intuition and theory.
- Implies that only support vectors are important; other training examples are ignorable.



About the Name...

A Support Vector

A training sample used to define classification boundaries in SVMs

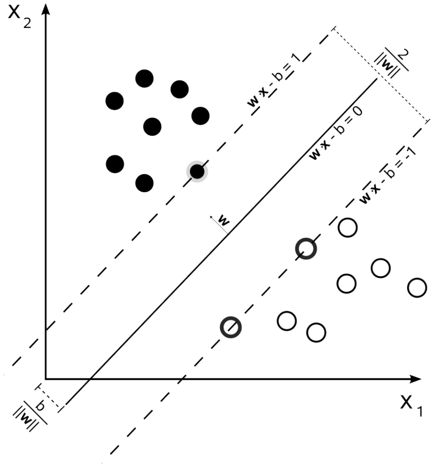
- located near class boundaries

Support Vector Machines

Binary classifiers whose decision boundaries are defined by support vectors

We are given a training dataset of n points:
$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$

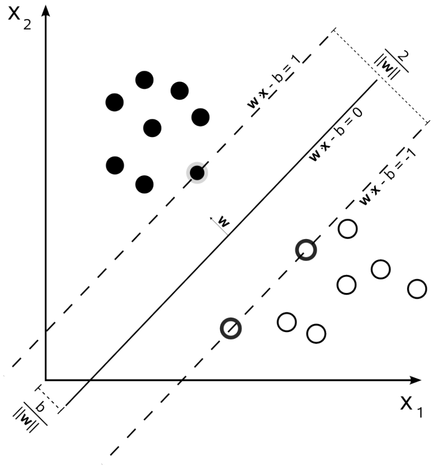
y_i is either 1 or –1, the class to which the point $\vec{x}_i$ belongs.



Each $\vec{x}_i$ is a p-dimensional real vector.

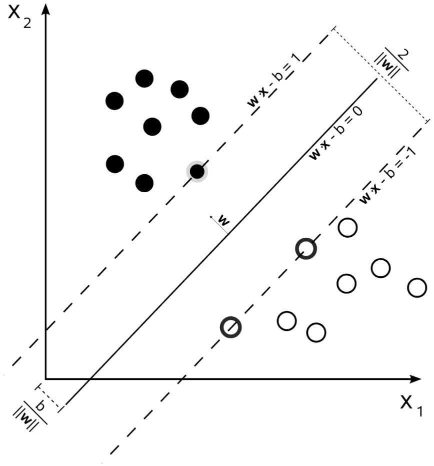
We want to find the "maximum-margin hyperplane" that divides the group of points $\vec{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$

The *maximum-margin hyperplane* is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized.



Any hyperplane can be written as the set of points $\vec{x}$ satisfying $\vec{w} \cdot \vec{x} - b = 0$, where $\vec{w}$ is the (not necessarily normalized) normal vector to the hyperplane.

The parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\vec{w}$.



More on ‘The Margin’

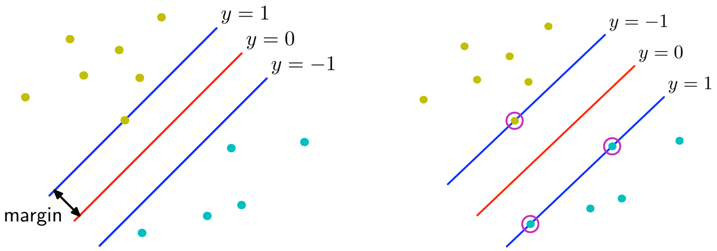
The Margin, *b*

Is the minimum distance of *any* sample to the decision boundary.

Training SVMs

= maximizing the margin, moving the decision boundary as far away from all training samples as possible.

Maximizing the Margin



- At left: a sub-optimal margin
- At right: optimal margin
- y values: for linear function defined by the SVM. For linearly separable data, all training instances correctly classified as -1 or 1 (locations in the margins have y values in (-1,1))

R·I·T *Bishop, "Pattern Recognition and Machine Learning," p. 327 6

Support Vector Machines

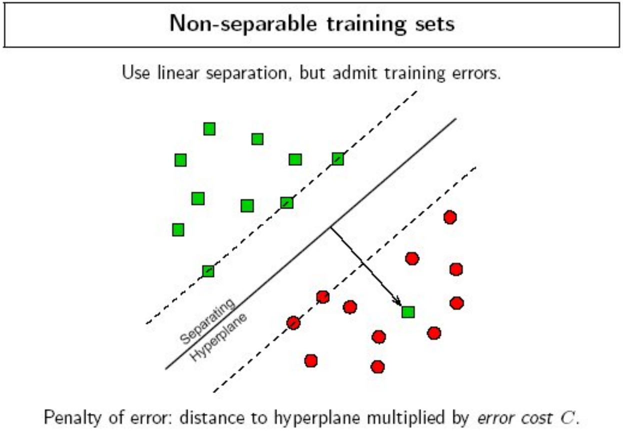
- SVMs pick best separating hyperplane according to some criterion
  - e.g. maximum margin
- Training process is an optimisation
- Training set is effectively reduced to a relatively small number of support vectors

Soft margin

what if there isn't a margin that can split the data linearly?

Soft margin

what if there isn't a margin that can split the data linearly?



Non-separable training sets

Use linear separation, but admit training errors.

Separating Hyperplane

Penalty of error: distance to hyperplane multiplied by *error cost C*.

Soft margin

what if there isn't a margin that can split the data linearly?

- introduce a parameter to allow for (a certain amount of) mislabelled examples
- the C parameter tells the SVM optimization how much you want to avoid misclassifying each training example
  - large values of C: large penalty to errors; the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly
  - a very small value of C: the optimizer will look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.
  - very very tiny values of C: you are likely to get misclassified examples, often even if your training data is linearly separable.

Soft margin

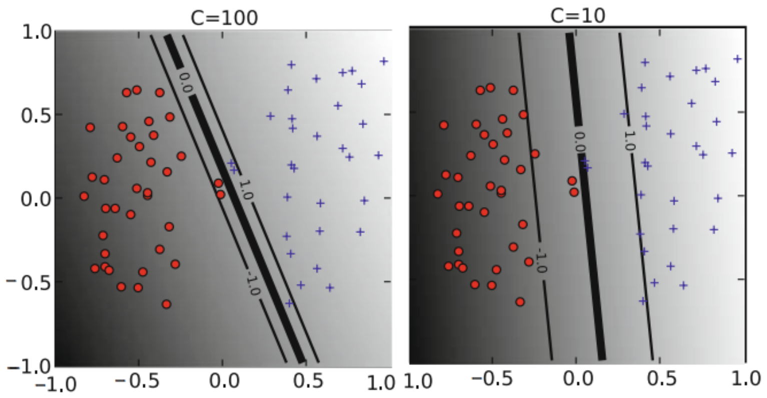
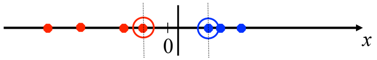


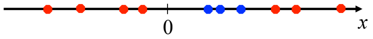
Fig. 13.3. The effect of the soft-margin constant, *C*, on the decision boundary. A smaller value of *C* (*right*) allows to ignore points close to the boundary and increases the margin. The decision boundary between negative examples (*circles*) and positive examples (*crosses*) is shown as a thick line. The lighter lines are on the margin (discriminant value equal to −1 or +1). The grayscale level represents the value of the discriminant function, dark for low values and a light shade for high values.

Non-linear SVMs

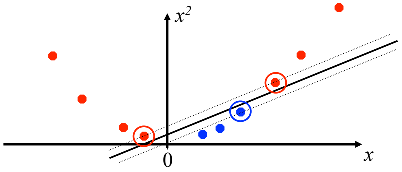
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

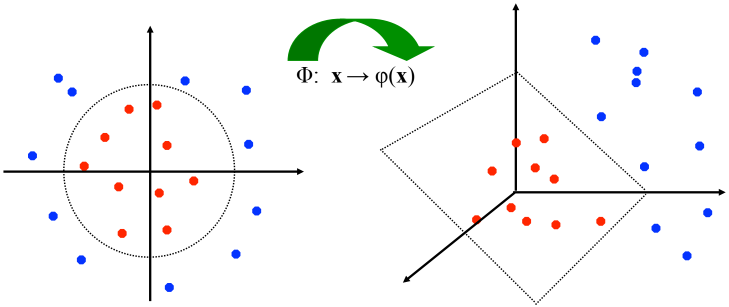


- How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



Kernels

- We may use Kernel functions to implicitly map to a new feature space
- Kernel fn:

$$K(\mathbf{x}_1, \mathbf{x}_2) \in \mathbf{R}$$

- Kernel must be equivalent to an inner product in some feature space

Example Kernels

Linear:	$\langle \mathbf{x} \cdot \mathbf{z} \rangle$
Polynomial:	$P(\langle \mathbf{x} \cdot \mathbf{z} \rangle)$
Gaussian:	$\exp\left(-\lVert \mathbf{x} - \mathbf{z} \rVert^2 / \sigma^2\right)$

SVM in Scikit-learn

```
from sklearn import svm

C=1.0 # remember what this is?
cls = svm.SVC(kernel='linear', C=C)
classifier = Pipeline( [('vec', vec),
                        ('cls', cls)] )
classifier.fit(X, y)
```

Check: `http://scikit-learn.org/stable/modules/svm.html#tips-on-practical-use` supervised classification algorithms

supervised classification algorithms

supervised classification algorithms

Classification vs Regression

create models of prediction from gathered data

- classification
  the dependent variables are categorical
  - input x: feature vector
  - output: **discrete class label**

- regression
  the dependent variables are numerical
  - input x: feature vector
  - output y: **continuous value**

classification and regression are the most standard ways of doing **supervised learning**

Supervised and Unsupervised Learning

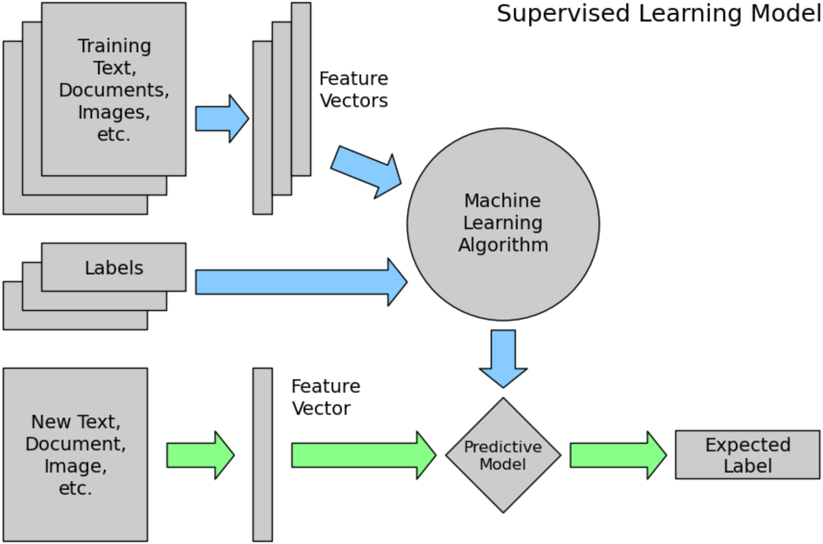
information about the correct distribution/label of the training examples

- in supervised learning it is known

  $\rightarrow$ fitting a model to labelled data which has the correct answer
- in unsupervised learning it is not known

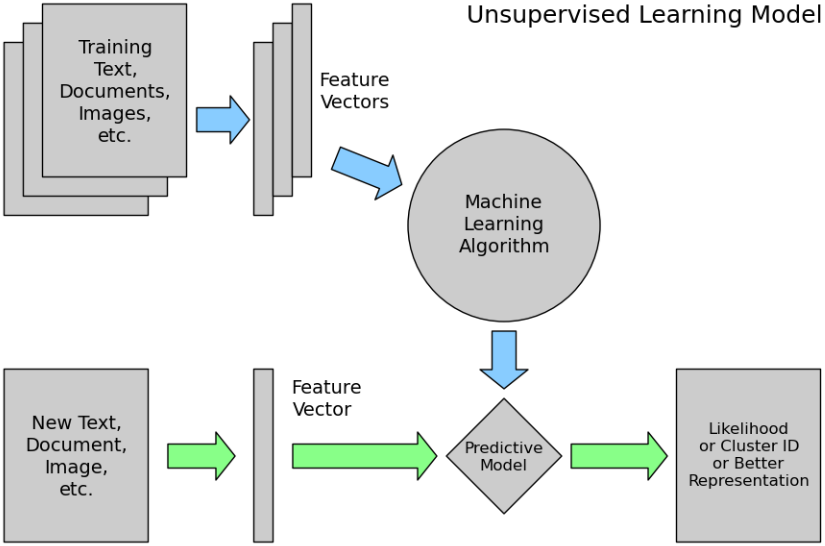
  $\rightarrow$ finding structure in unlabelled data

Supervised Learning Model


Training Text, Documents, Images, etc.
Feature Vectors
Labels
Machine Learning Algorithm
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Expected Label

source: `http://www.astroml.org/`

Unsupervised Learning Model


Training Text, Documents, Images, etc.
Feature Vectors
Machine Learning Algorithm
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Likelihood or Cluster ID or Better Representation

source: `http://www.astroml.org/`

Supervised learning

supervised learning – classification or regression

- in training, instances are associated with their class label
- based on features, the system must search for patterns and build a model
- the model must be able to *predict* the class of previously unseen instances

Unsupervised learning

unsupervised learning – clustering

- partitioning instances into subsets (clusters) that share similar characteristics
- subsets are not predefined
- a system can be told *how many* clusters it should form (K-means algorithm)

clustering

Classification vs. Clustering

- Classification: supervised learning
- Clustering: unsupervised learning
- Classification: Classes are human-defined and part of the input to the learning algorithm.
- Clustering: Clusters are inferred from the data without human input.
    - However, there are many ways of influencing the outcome of clustering: number of clusters, similarity measure, representation of documents, . . .

Soft vs Hard Clustering

- hard: partition the objects so that each object is in **exactly one** partition
- soft: assign a **degree** to which each object is in one cluster. It's a bit like fractional membership in several clusters: a document's assignment is a distribution over all clusters

Soft vs Hard Clustering

- hard: partition the objects so that each object is in **exactly one** partition
- soft: assign a **degree** to which each object is in one cluster. It's a bit like fractional membership in several clusters: a document's assignment is a distribution over all clusters (cf. probability scores)

Flat vs Hierarchical Clustering

- flat: a set of clusters without any explicit structure that would relate clusters to each other.
- hierarchical: clusters within clusters (a hierarchy of clusters)

Clustering: overview

take a collection of objects, like our documents

- assume that each object can be represented as a point in a geometric space (VSM!)
- typically the points are constructed by using word frequencies (bag of words)

goal: create clusters that are *coherent internally, but clearly different from each other*

- documents within a cluster should be as similar as possible
- documents in one cluster should be as dissimilar as possible from documents in other clusters

K-means

K-means

- flat clustering
- number of clusters picked ahead of time
- iterative improvement
- notion of centroid
- typically uses Euclidean distance

K-means

- flat clustering
- number of clusters picked ahead of time
- iterative improvement
- notion of centroid
- typically uses Euclidean distance

objective: to minimize the average squared Euclidean distance of documents from their cluster centers, where a cluster center is defined as the mean or *centroid* of the documents in a cluster ω

K-means

Assume that there are *k* classes

- for every class, create a centroid: a point that is in the center of the class
- find centroids so that all the points in each class are as near as possible

K-means — iteration

- choose K (number of clusters)
- initialize K centroids

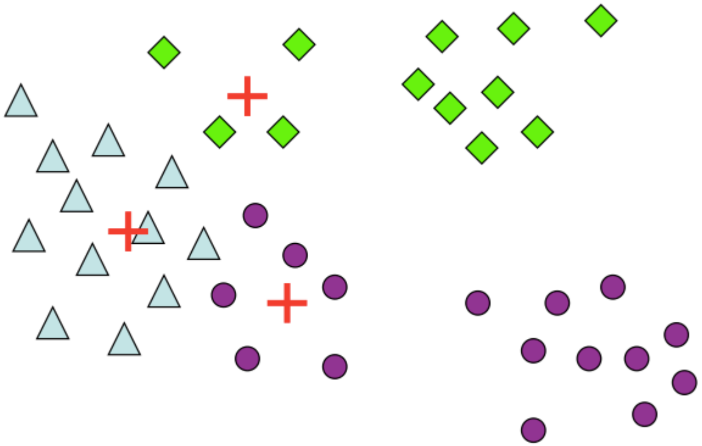
Then, repeatedly perform the following two steps:

1. cluster assignment:
   - points are assigned the class of the closest centroid
2. move centroid
   - Move centroid to the average of the points of the same cluster

After a number of iterations, centroid/assignments dont change anymore, then stop.

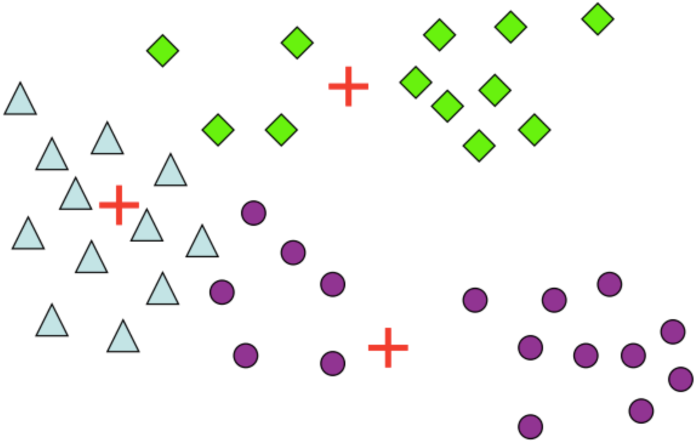
An Example

start: choose centroids and cluster



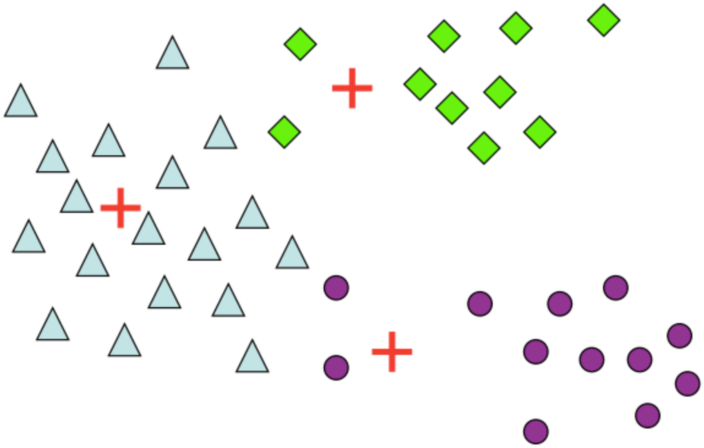
An Example

recompute centroids



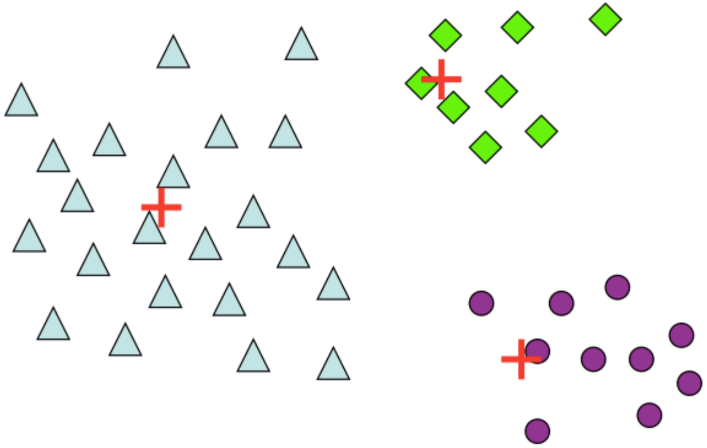
An Example

re-cluster around new centroids



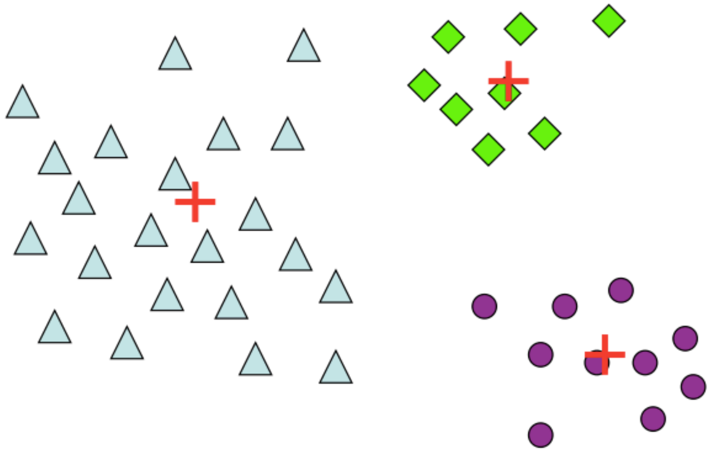
An Example

2nd recompute centroids and re-cluster

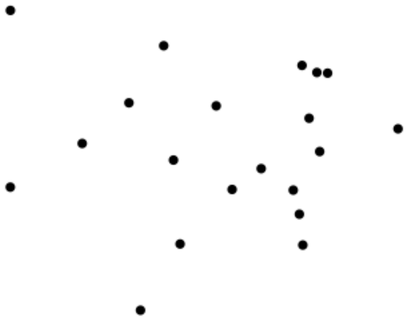


An Example

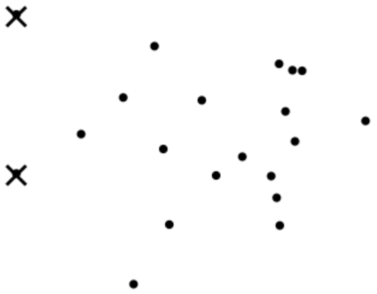
3rd (final) recompute and re-cluster



Worked Example: Set of to be clustered

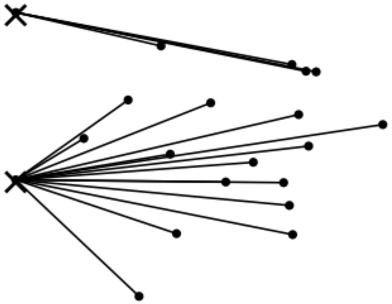


Worked Example: Random selection of initial centroids

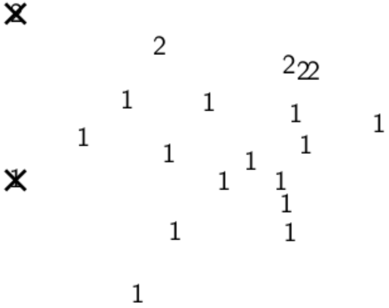


Exercise: (i) Guess what the optimal clustering into two clusters is in this case; (ii) compute the centroids of the clusters

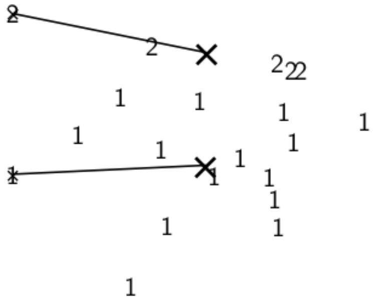
Worked Example: Assign points to closest center



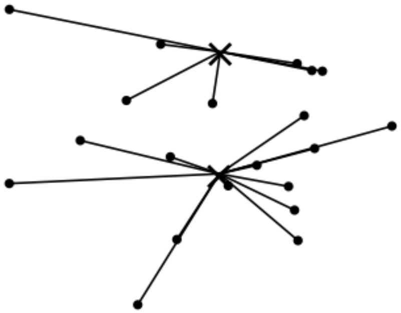
Worked Example: Assignment



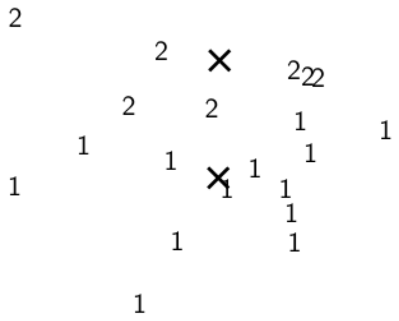
Worked Example: Recompute cluster centroids



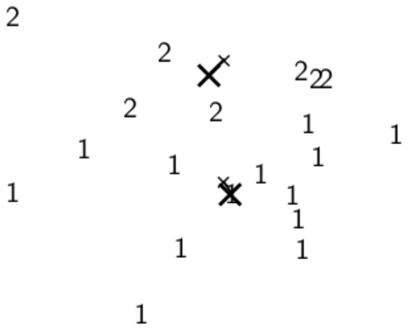
Worked Example: Assign points to closest centroid



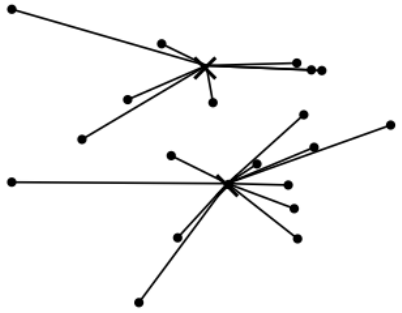
Worked Example: Assignment



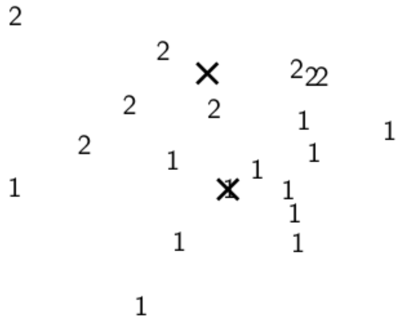
Worked Example: Recompute cluster centroids



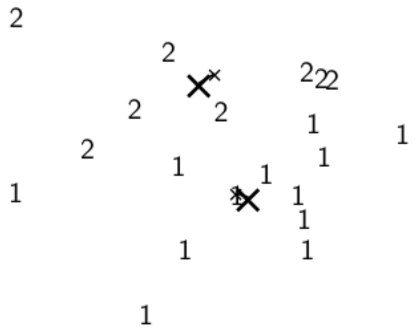
Worked Example: Assign points to closest centroid



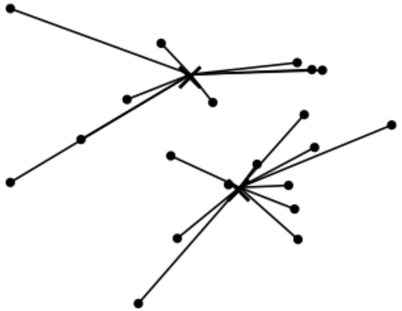
Worked Example: Assignment



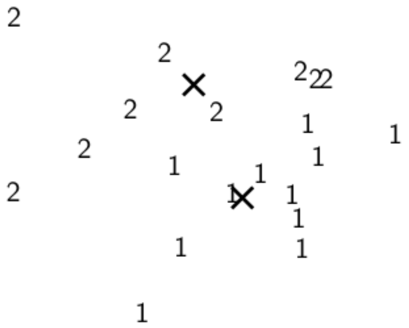
Worked Example: Recompute cluster centroids



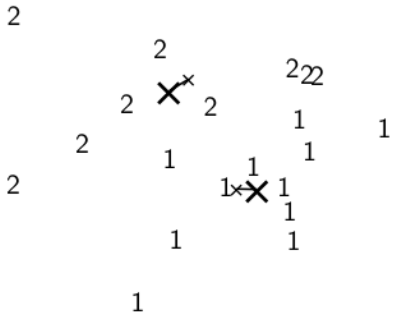
Worked Example: Assign points to closest centroid



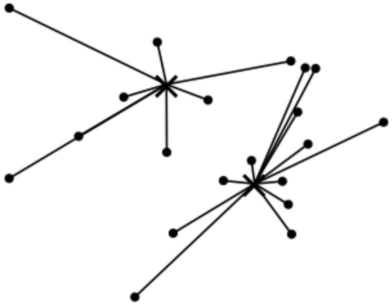
Worked Example: Assignment



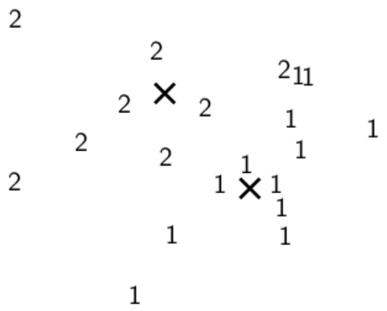
Worked Example: Recompute cluster centroids



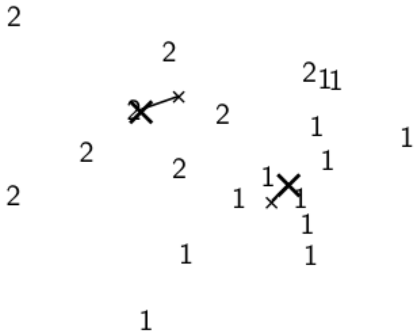
Worked Example: Assign points to closest centroid



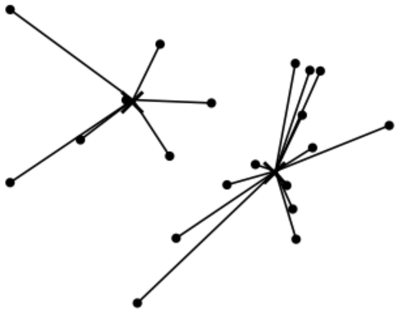
Worked Example: Assignment



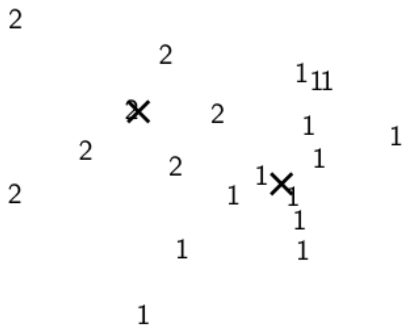
Worked Example: Recompute cluster centroids



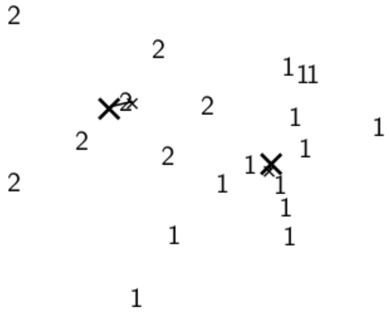
Worked Example: Assign points to closest centroid



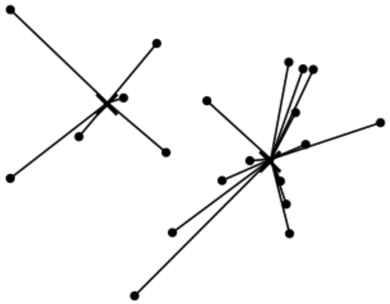
Worked Example: Assignment



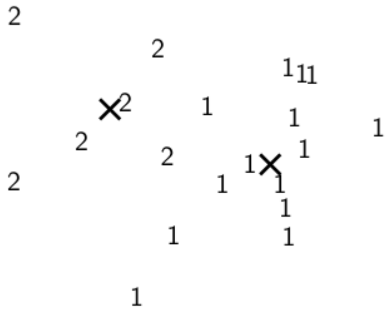
Worked Example: Recompute cluster centroids



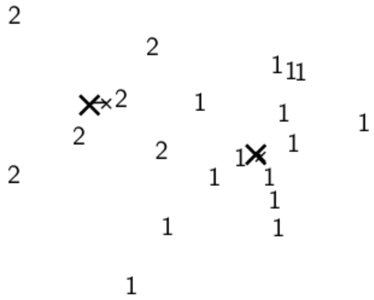
Worked Example: Assign points to closest centroid



Worked Example: Assignment



Worked Example: Recompute cluster caentroids



Demo

demo

Crucial K-means decisions

- K = ?
- initialisation (seeds)

Choosing K

- K can be given/known
  - experience for a given task
  - application/task requirements
- it might be not known, and finding the “right” number of clusters is part of the problem:
  - given docs, partition into an “appropriate” number of subsets
- tradeoff between having more clusters (more focus in each cluster) and having too many clusters

Initialising centroids

- random
  - fast and easy, but often poor results
- run random multiple times, take best
  - slower, and still no guarantee of results
- pre-conditioning
  - remove outliers
- choose seeds algorithmically
  - run hierarchical clustering on sample points and use resulting centroids
  - works well on small samples and for few initial centroids

Evaluation

How do we evaluate clusters?

Evaluation

How do we evaluate clusters?
If we have labels, we can tell how well the clustering matches the gold standard classes (external evaluation).
Note: we only use *the partition* provided by the gold standard, *not the class labels*.

- purity is a simple and transparent evaluation measure.
- rand index (RI) penalizes both false positive and false negative decisions during clustering
- F-measure supports differential weighting of FPs and FNs
- normalized mutual information (NMI) can be information-theoretically interpreted.

Purity

- N = total number of documents
- each cluster is assigned to the class which is most frequent in the cluster
- accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N.

Purity

- N = total number of documents
- each cluster is assigned to the class which is most frequent in the cluster
- accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N.

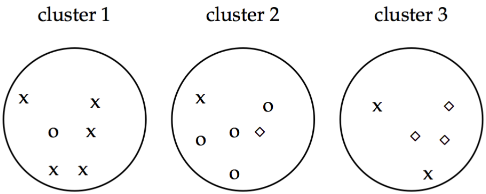
$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{K} \max_{j} |\omega_k \bigcap c_j|$$

- $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ = set of clusters
- $\mathbb{C} = \{c_1, c_2, ..., c_j\}$ = set of classes

Purity

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{K} \max_{j} |\omega_k \bigcap c_j|$$

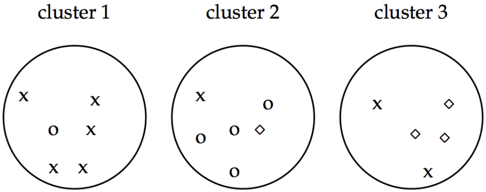
- $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ = set of clusters
- $\mathbb{C} = \{c_1, c_2, ..., c_j\}$ = set of classes



Purity

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{K} \max_{j} |\omega_k \bigcap c_j|$$

- $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ = set of clusters
- $\mathbb{C} = \{c_1, c_2, ..., c_j\}$ = set of classes

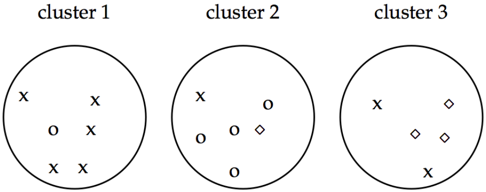


5 +

Purity

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_K \max_j |\omega_k \bigcap c_j|$$

- $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ = set of clusters
- $\mathbb{C} = \{c_1, c_2, ..., c_j\}$ = set of classes

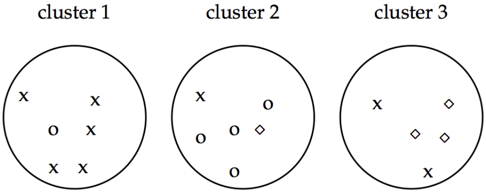


5 + 4 +

Purity

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{K} \max_{j} |\omega_k \bigcap c_j|$$

- $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ = set of clusters
- $\mathbb{C} = \{c_1, c_2, ..., c_j\}$ = set of classes

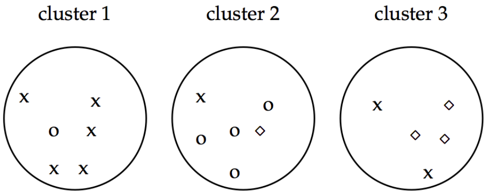


5 + 4 + 3

Purity

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{K} \max_{j} |\omega_k \bigcap c_j|$$

- $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ = set of clusters
- $\mathbb{C} = \{c_1, c_2, ..., c_j\}$ = set of classes



$$\frac{1}{17} \text{ x } (5 + 4 + 3) = 0.71$$

Purity

- N = total number of documents
- each cluster is assigned to the class which is most frequent in the cluster
- accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N.

what is a sure case where $\mathit{purity} = 1$?

Purity

- N = total number of documents
- each cluster is assigned to the class which is most frequent in the cluster
- accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N.

what is a sure case where $\textit{purity} = 1$?

the larger the number of clusters...

Rand Index

RI measures the percentage of decisions that are correct

Rand Index

RI measures the percentage of decisions that are correct

Consider the errors:

- FP
- FN

Rand Index

RI measures the percentage of decisions that are correct

Consider the errors:

- FP – a false positive decision assigns two dissimilar documents to the same cluster.
- FN – a false negative decision assigns two similar documents to different clusters.

Rand Index

RI measures the percentage of decisions that are correct

Consider the errors:

- FP – a false positive decision assigns two dissimilar documents to the same cluster.
- FN – a false negative decision assigns two similar documents to different clusters.

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

Rand Index

RI measures the percentage of decisions that are correct

Consider the errors:

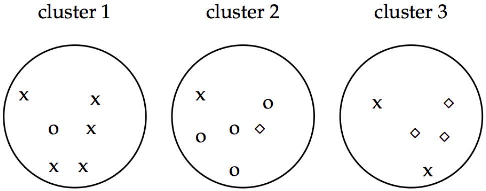
- FP – a false positive decision assigns two dissimilar documents to the same cluster.
- FN – a false negative decision assigns two similar documents to different clusters.

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

(accuracy)

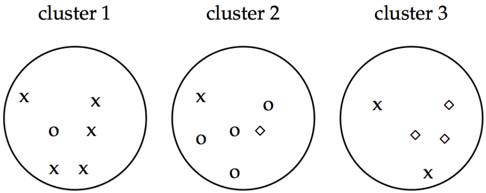
Rand Index

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



Rand Index

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



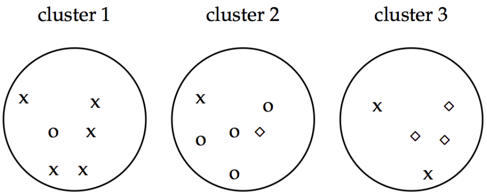
- positives = number of pairs that are in the same cluster

  TP + FP = $\binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40$

  TP = $\binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20$

Rand Index

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



	same cluster	different cluster
same class	TP=20	FN=24
different class	FP=20	TN=74

$$RI = \frac{20 + 72}{20 + 20 + 24 + 72} = 0.68$$

F-measure

$$RI = \frac{TP + TN}{TP + FP + TN + FN}$$ = equal weight to all errors (FPs and FNs)

BUT: separating similar documents is sometimes worse than putting pairs of dissimilar documents in the same cluster
(especially true in IR: favour recall over precision)

F-measure

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

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

F-measure

$$\frac{TP}{TP + FP} \quad \text{(precision)}$$

$$\frac{TP}{TP + FN} \quad \text{(recall)}$$

$$F = \frac{2PR}{P + R} \quad \text{(F-measure, also called F}_1\text{)}$$

F-measure

$$F = \frac{2PR}{P + R}$$ (F-measure, also called F$_1$)

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- if $\beta = 1$, then standard F-measure (F$_1$)
- if $\beta < 1$, then precision emphasised
- if $\beta > 1$, then recall emphasised

F-measure

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- if $\beta = 1$, then standard F-measure (F_1)
- if $\beta < 1$, then precision emphasised
- if $\beta > 1$, then recall emphasised

	same cluster	different cluster
same class	TP=20	FN=24
different class	FP=20	TN=74

F_1 = 0.48
F_5 = 0.46

Normalised Mutual Information

- how much information does the clustering contain about the classification?
- singleton clusters (number of clusters = number of docs) have maximum MI
- therefore: normalise by entropy of clusters and classes

What do clusters mean?

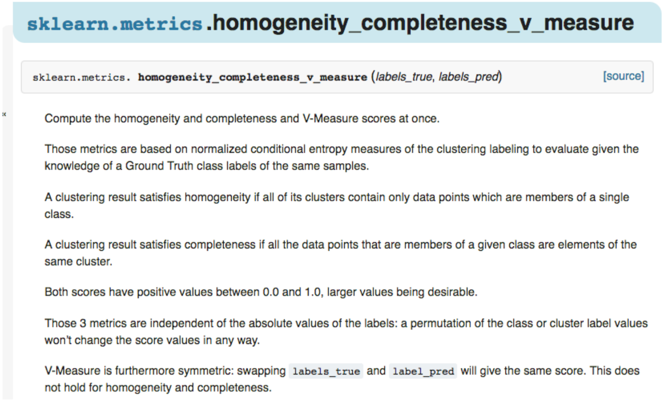
if we cluster our reviews, will it be on topic or sentiment?
or possibly something else? How can we know they're not clustered on say the author's gender? or Age?

Clustering with scikit-learn

`http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

Clustering with scikit-learn

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html



sklearn.metrics.homogeneity_completeness_v_measure

`sklearn.metrics.` **`homogeneity_completeness_v_measure`** (*labels_true*, *labels_pred*) [source]

Compute the homogeneity and completeness and V-Measure scores at once.

Those metrics are based on normalized conditional entropy measures of the clustering labeling to evaluate given the knowledge of a Ground Truth class labels of the same samples.

A clustering result satisfies homogeneity if all of its clusters contain only data points which are members of a single class.

A clustering result satisfies completeness if all the data points that are members of a given class are elements of the same cluster.

Both scores have positive values between 0.0 and 1.0, larger values being desirable.

Those 3 metrics are independent of the absolute values of the labels: a permutation of the class or cluster label values won't change the score values in any way.

V-Measure is furthermore symmetric: swapping `labels_true` and `label_pred` will give the same score. This does not hold for homogeneity and completeness.