IN2110 SPRING 2022

SPRÅKTEKNOLOGISKE METODER

Erik Velldal & Jan Tore Lønning

Plan – week 4

Lectures 2-5

- How to represent (language) data in a mathematical model.
- Vector space models.
- Representing
  - Documents (today)
  - Words (week 5)
- Vector-based machine learning
  - Classification (week 3)
  - Clustering (week 4)

Today

- Recap
- Evaluating classifiers
- Clustering

Disclaimer

- I am only a substitute teacher for Erik Velldal
- The slides will be a mixture
  - Erik's slides from last year
  - My slides from IN3050 and IN4080
  - Some new slides (like this one)

Recap

Three main types of ML

Supervised learning

- Learn from labeled data

-

 ?

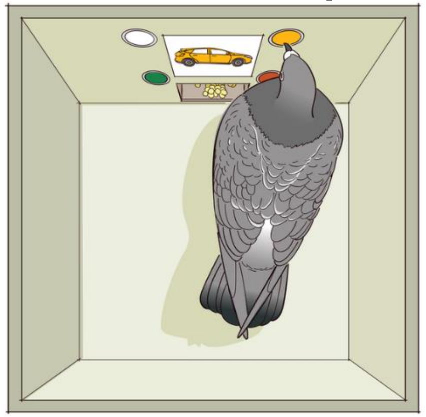
Unsupervised learning

- No labeled data

- Task: identify similarities and categorize together

Reinforcement learning

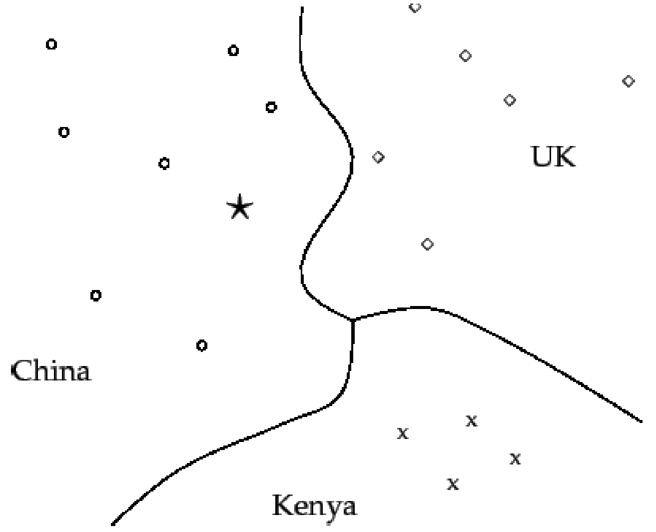
- Training with rewards (and punishments)



Source: Wikipedia

Classification based on vector spaces

- In our vector space model, objects are represented as points, so classes will correspond to collections of points; regions.
- Vector space classification is based on the contiguity hypothesis:



- Objects in the same class form a contiguous region, and regions of different classes do not overlap.
- Classification amounts to computing the boundaries in the space that separate the classes; the decision boundaries.

Two algorithms

Rocchio

- Training: Calculate the centroid to each class in the training set.
- Application: assign an object to the class with the nearest centroid
- A linear classifier
- Strong assumptions (bias) regarding the classes

***K* nearest neighbors (*k* NN)**

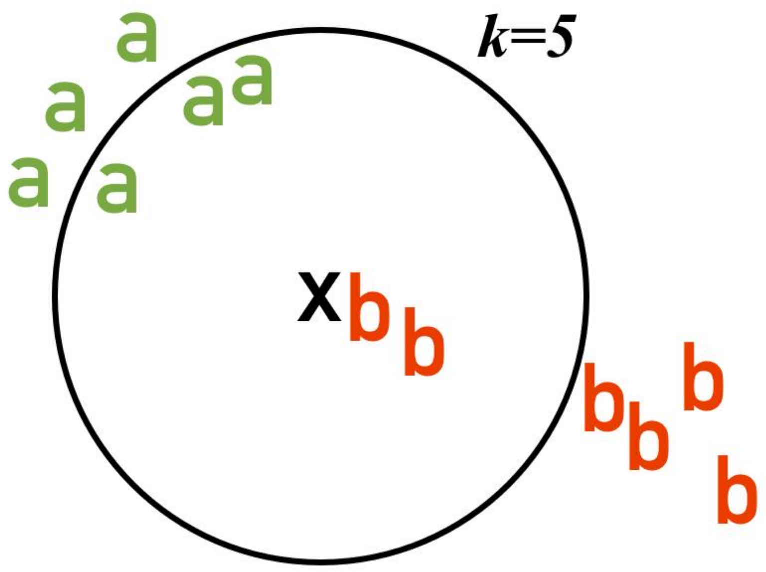
- No real training
- Application:
  - Find the *k* nearest neighbors
  - Pick the majority class of the neighbors
- Non-linear

Properties of *k*NN

- Instance-based, no real training
  - it simply memorizes all training examples
  - Fast to "train"
- Inefficient in predicting the label of new instances
  - Since it must consider all the training data each time (= linear in the size of the training set)
- Notice the similarity to retrieving relevant documents for a given query: Both are instances of finding nearest neighbors.
- One parameter: *k*
- The distance measure may influence the result
- The scaling of the axes might influence the result

Probabilistic *k*NN

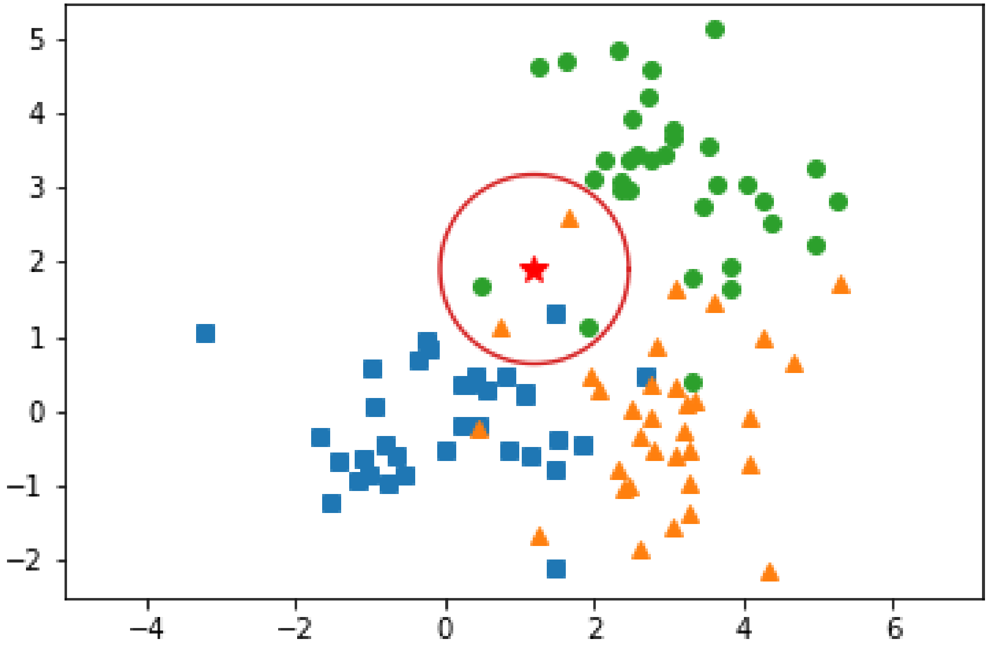
- Sometimes, we are not interested in a hard decision,
  but rather the probability of an item belonging to a class
  - In particular if we are to combine this with other information
- *k*NN can be made probabilistic:
  - The probability of class c is the proportion of the *k* nearest neighbors in c.
- We may here also apply the weighting from next slide



- $P(a|x) = \frac{3}{5}$
- $P(b|x) = \frac{2}{5}$

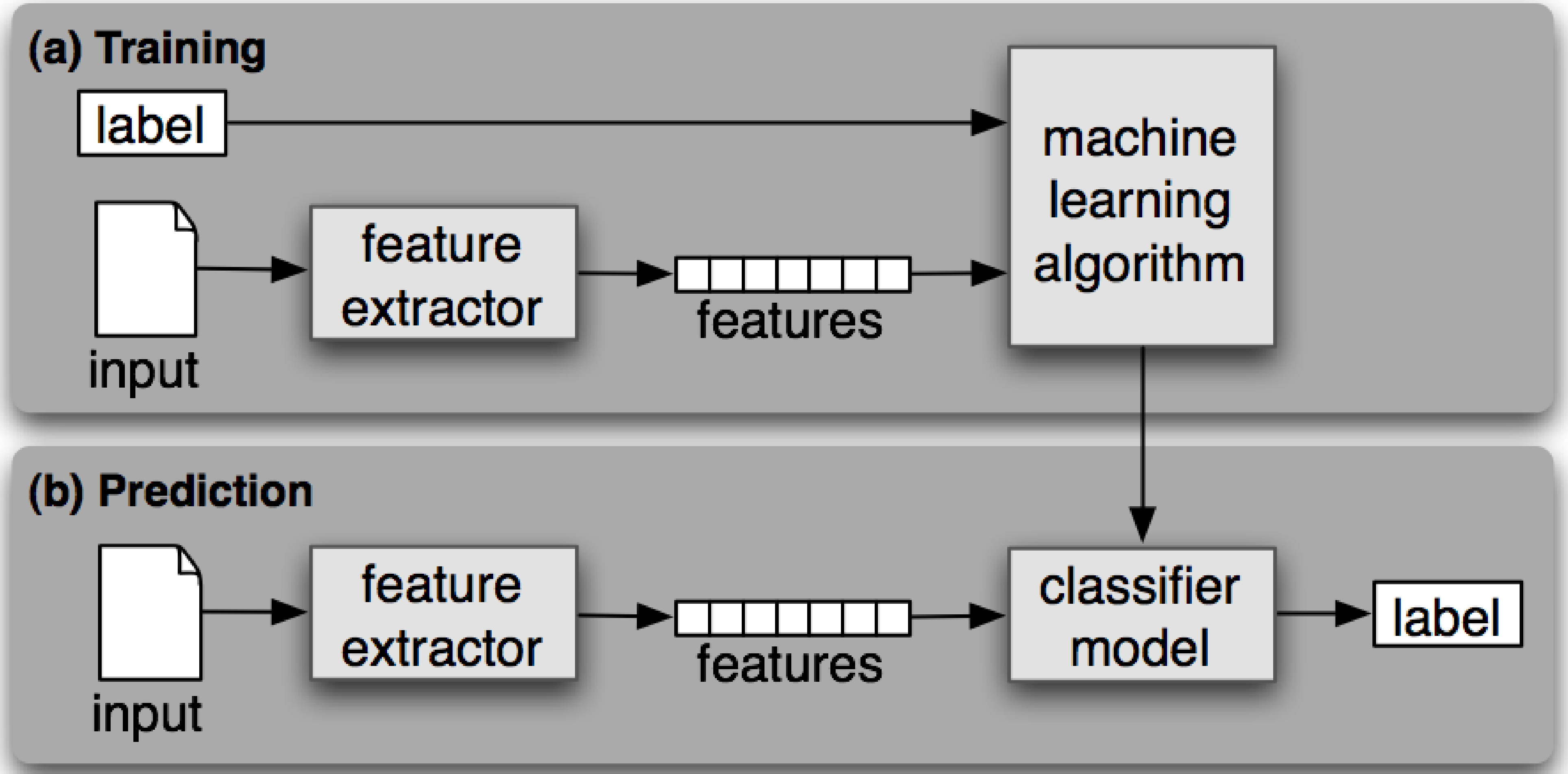
Footnote: More than two classes

- A binary classifier with odd *k* always reaches a decision
- With more than 2 classes, there might be a draw
- One possible way out
  1. Weight points by inverse distance from target,
  2. Sum weighted distances for each class
  3. Choose the class with largest weighted max.



Evaluation of classifiers

Classification

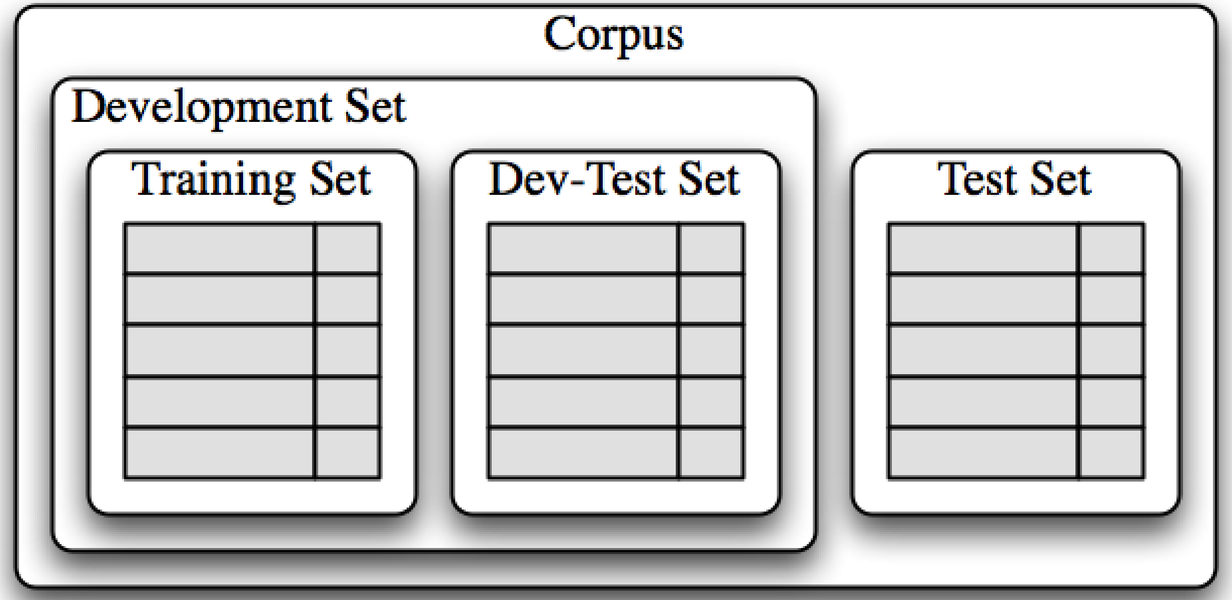


https://www.nltk.org/book/ch06.html

A practical note on feature extraction

- Before training and applying a classifier we first have to create the feature vectors to represent our data.
- Sometimes referred to as vectorization.
- The feature types (e.g. the BoW vocabulary) needs to be defined relative to the training set (including parameters like frequency cut-offs, idf-weights, etc).
- When vectorizing test data we must use the same features as in training.
- Vectorization therefore often done in two passes: first defining the feature set based on the training data, then creating the feature vectors. (*Fit* and *transform* in scikit-learn terminology)

Procedure



https://www.nltk.org/book/ch06.html

1. Train classifier on training set
2. Test it on dev-test set
3. Compare to earlier runs,
   - is this better?
4. Error analysis: What are the mistakes (on dev-test set)
5. Make changes to the classifier
6. Repeat from 1

==================

- When you have run empty on ideas, test on test set. Stop!

Cross-validation

- Small test sets ➔ Large variation in results
- N-fold cross-validation:
  - Split the development set into n equally sized bins
    - (e.g. n = 10)
  - Conduct n many experiments:
    - In experiment m, use part m as test set and the n-1 other parts as training set.
  - This yields n many results:
    - We can consider the mean of the results
    - We can consider the variation between the results.
      - Statistics!

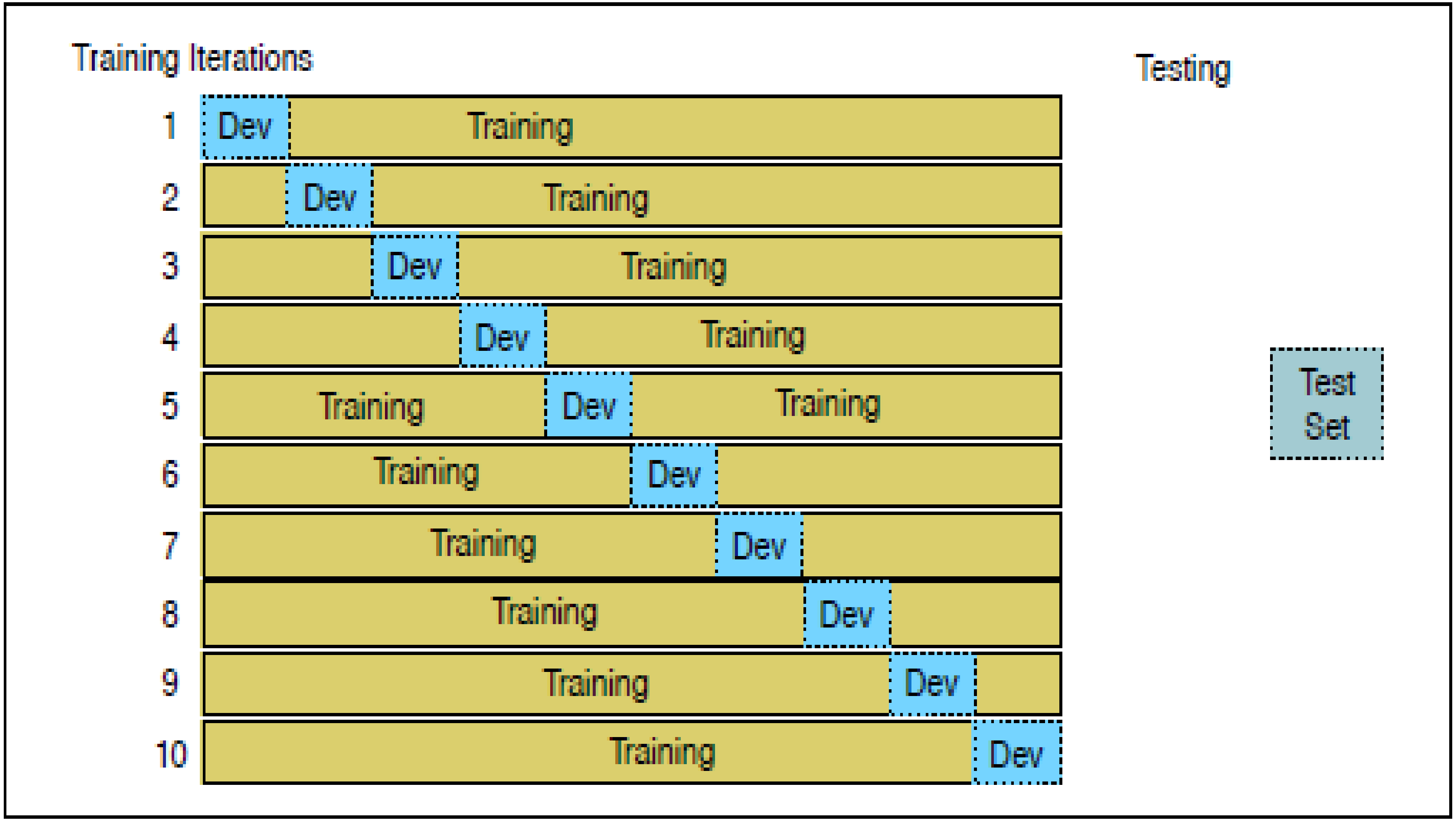
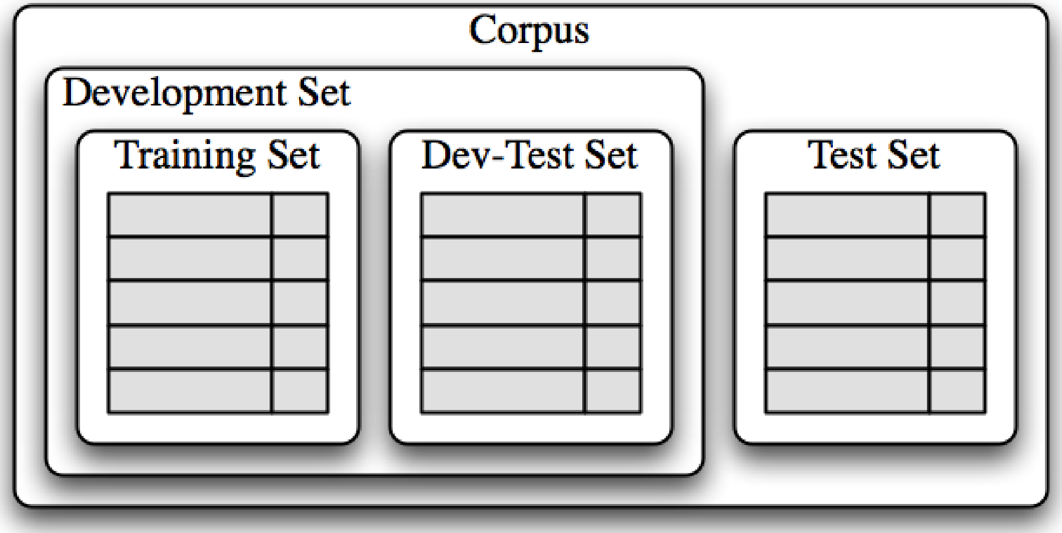


Figure 6.7 10-fold crossvalidation

Testing a classifier

- Train on the training set.
- Predict labels on the test set (after removing the labels)
- Compare the prediction to the given labels (called gold labels)



https://www.nltk.org/book/ch06.html

Confusion matrix and accuracy

		True label	
		Yes	NO
Predicted label	Yes	tp=150	fp=50
	No	fn=100	tn=200

Goal: Evaluate our spam classifier

- We run the classifier on the labeled test set (without the labels)
- Compare the predicted labels to the example labels and count
- We can present the numbers in a confusion table

- True positives, tp=150
- False positives, fp=50
- False negatives, fn=100
- True negatives, tn=200
- Accuracy: (tp+tn)/N = 350/500 = 0.7

More than two classes

		True label		
		spam	normal	urgent
Predicted label	spam	150	49	1
	normal	31	250	19
	urgent	19	31	50

Accuracy:

- (sum of the diagonal)/N
- $= {}^{\#\{y_i|y_i=t_i\}}\!/_{\#\{y_i\}} = {}^{450}\!/_{600} = 0.75$

Observe

- There is no consensus regarding what should be the columns and what should be the rows

Evaluation measure: Accuracy

- What does accuracy 0.81 tell us?
- Given a test set of 500 documents:
  - The classifier will classify 405 correctly
  - And 95 incorrectly
- A good measure given:
  - The 2 classes are equally important
  - The 2 classes are roughly equally sized
  - Example:
    - Woman/man
    - Movie reviews: pos/neg

But

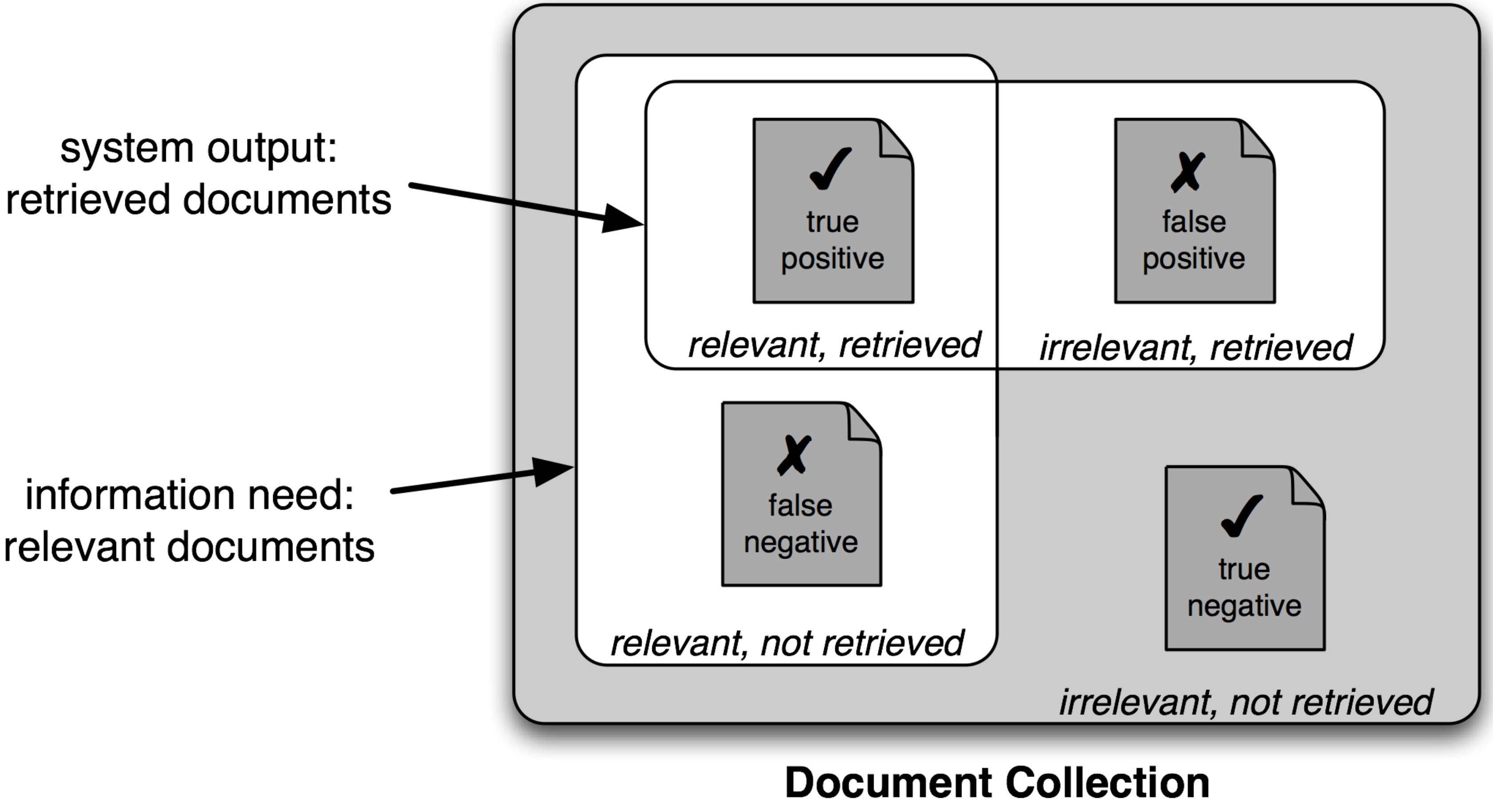
- For some tasks, the classes aren't equally important
  - Worse to lose an important mail than to receive yet another spam mail
- For some tasks, the different classes have different sizes.

Information retrieval (IR)

- Traditional IR, e.g., a library
  - Goal: Find all the documents on a particular topic out of 100 000 documents,
    - Say there are 5
  - The system delivers 10 documents: all irrelevant
    - What is the accuracy?

- For these tasks, focus on
  - The relevant documents
  - The documents returned by the system
- Forget the
  - Irrelevant documents which are not returned

IR - evaluation



Confusion matrix

		gold standard labels		
		gold positive	gold negative	
system output labels	system positive	**true positive**	**false positive**	precision = $\frac{tp}{tp+fp}$
	system negative	**false negative**	**true negative**	
		recall = $\frac{tp}{tp+fn}$		accuracy = $\frac{tp+tn}{tp+fp+tn+fn}$

Figure 6.4 Contingency table

- Beware what the rows and columns are:
  - NLTKs ConfusionMatrix swaps them compared to this table

Evaluation measures

		Is in C	
		Yes	NO
Classifier	Yes	tp	fp
	No	fn	tn

- Accuracy: (tp+tn)/N
- Precision:tp/(tp+fp)
- Recall: tp/(tp+fn)

- F-score combines P and R
- $F_1 = \frac{2PR}{P+R} \left(= \frac{1}{\frac{\frac{1}{R}+\frac{1}{P}}{2}} \right)$
- F_1 called "harmonic mean"
- General form
  - $F = \frac{1}{\alpha\frac{1}{P}+(1-\alpha)\frac{1}{R}}$
  - for some $0 < \alpha < 1$

Confusion matrix

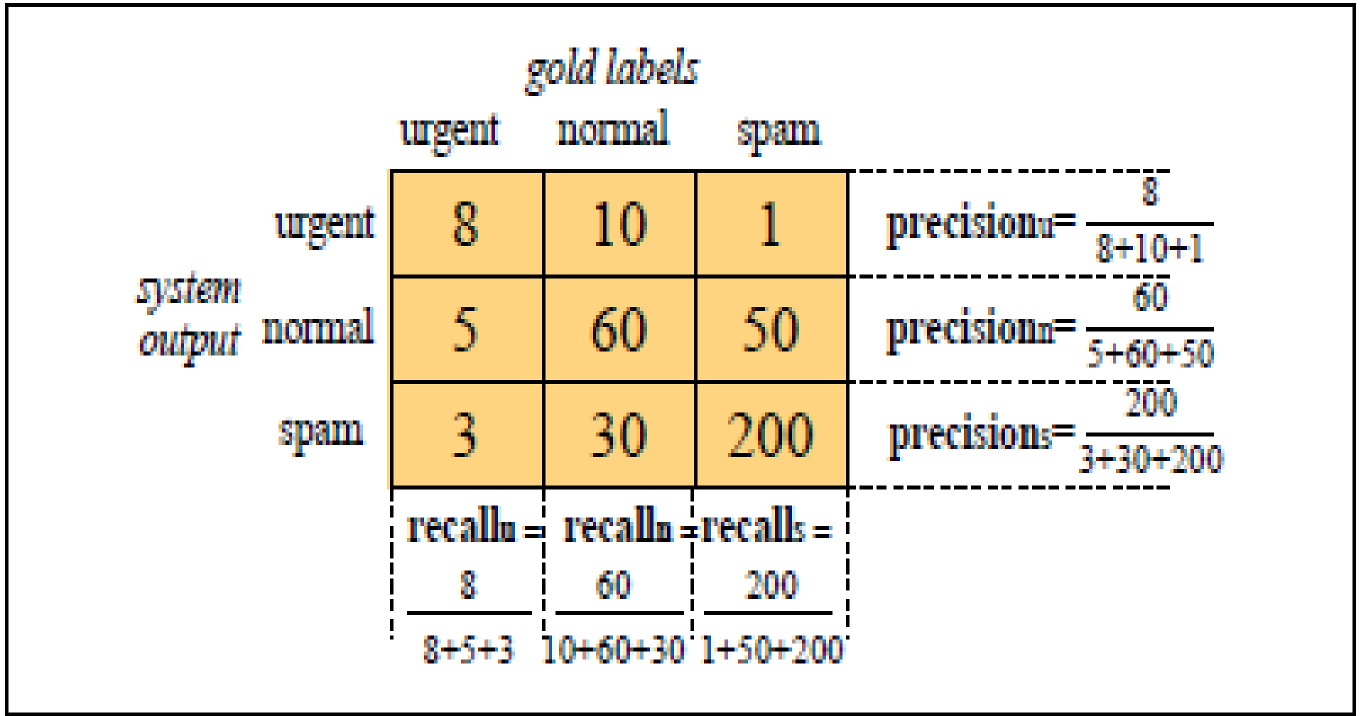


Figure 6.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2

- Precision, recall and f-score can be calculated for each class against the rest

Evaluating multi-class and multi-label classifiers

Macro-average

- Calculate Precision, P_i, for each class i, $i = 1,2, \dots N$,
  - (N different classes)
- Take the average of these $\frac{1}{N}\sum_{i=1}^{N} P_i$,
- Similarly for Recall and F-score
- Favors small classes

Micro-average

- Sum TP, FN, FP across the classes
- Use the formulas and calculate Precision, Recall and F-score from these using the formulas
- Favors large classes

To be continued

- More on classification later in this course
- More details on macro- and micro-average in IN4080

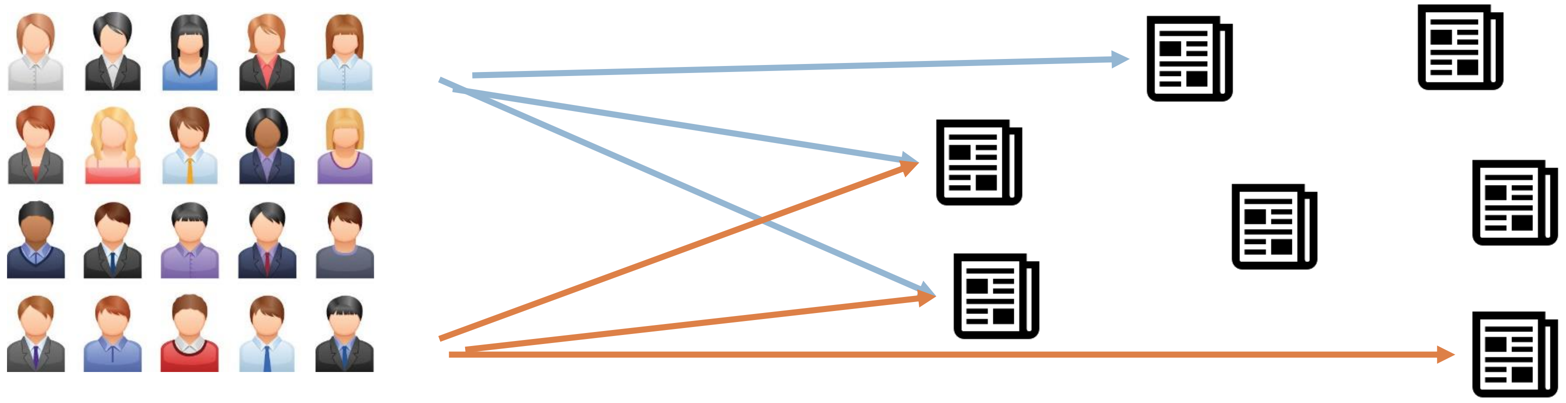
Clustering

An instance of unsupervised learning

2. Unsupervised learning - clustering

- *Can you sort the Lego bricks?*
  - (No instruction on how)
- You may choose sorting on
  - Color, or
  - Size, or
  - Shape, or
  - A combination
- I cannot know beforehand what you choose, but
- The result might me helpful

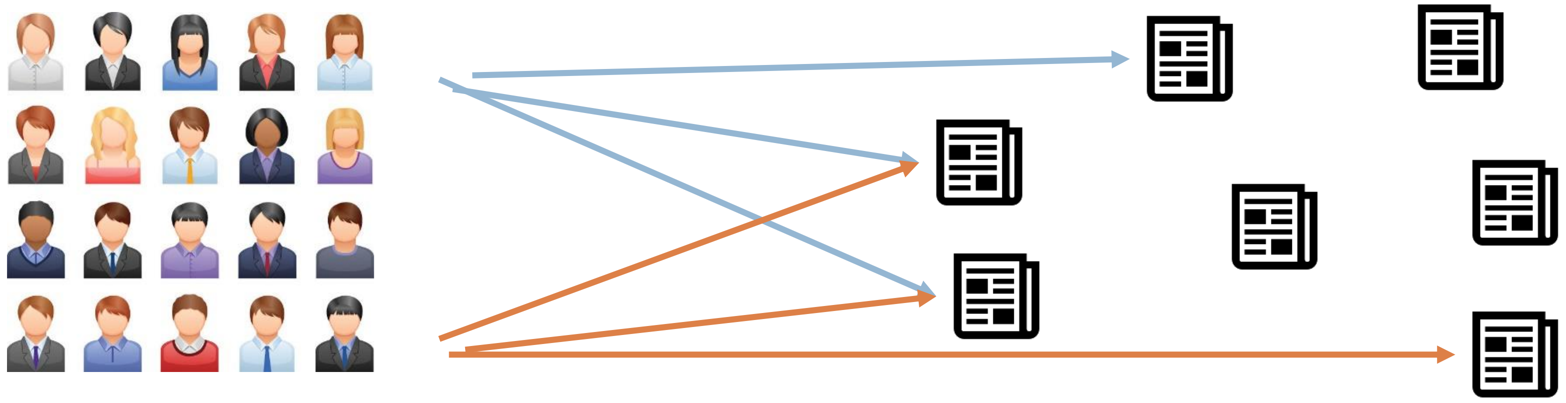
Unsupervised learning, example 2.1



- Everybody (Facebook, Schibsted, ..) collects what you are reading
- And want to use this to give you recommendations for readings which may interest you
  - (generate clicks)
- Assumption: Readers who have read the same stories before, have similar interests

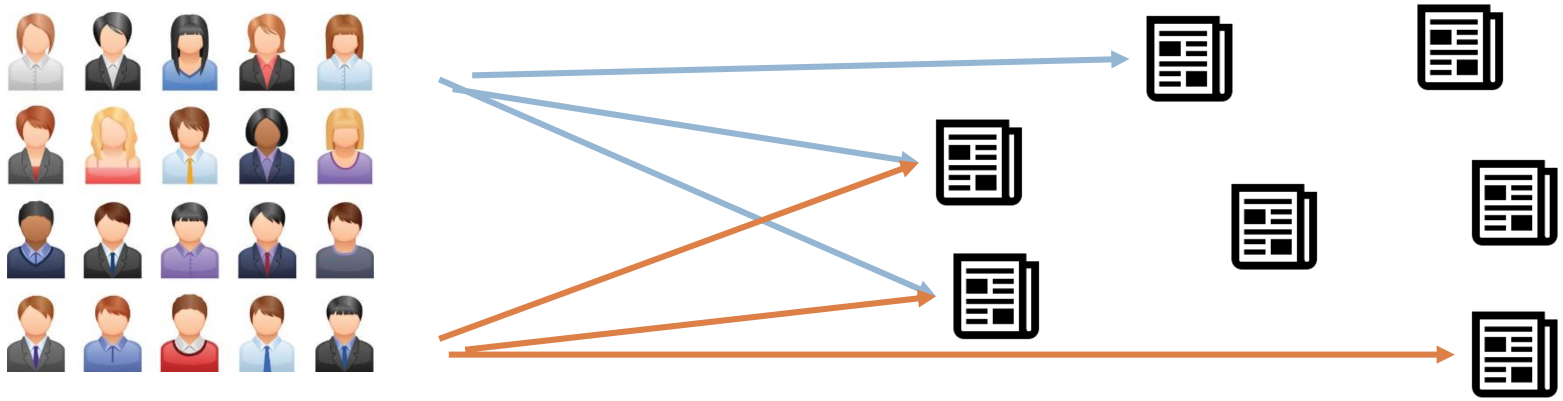
- Approach 1:
  - Compare your reading story to the reading story of all other users (One feature for each earlier story)
  - Select the k most similar readers
  - Give recommendations from what (else) they read
- This is kNN with its efficiency problems

Unsupervised learning, example 2.2



- Approach 2:
  - Assume that the readers are grouped into classes
    - Where readers in the same class have similar reading stories
  - A new reader is assigned to a group based on her reading story
  - Recommendations are made based on the groups common reading interest (Rocchio classification)
- This is much more efficient
- But how do we find the groups?
- Clustering!

Unsupervised learning, example 2.3



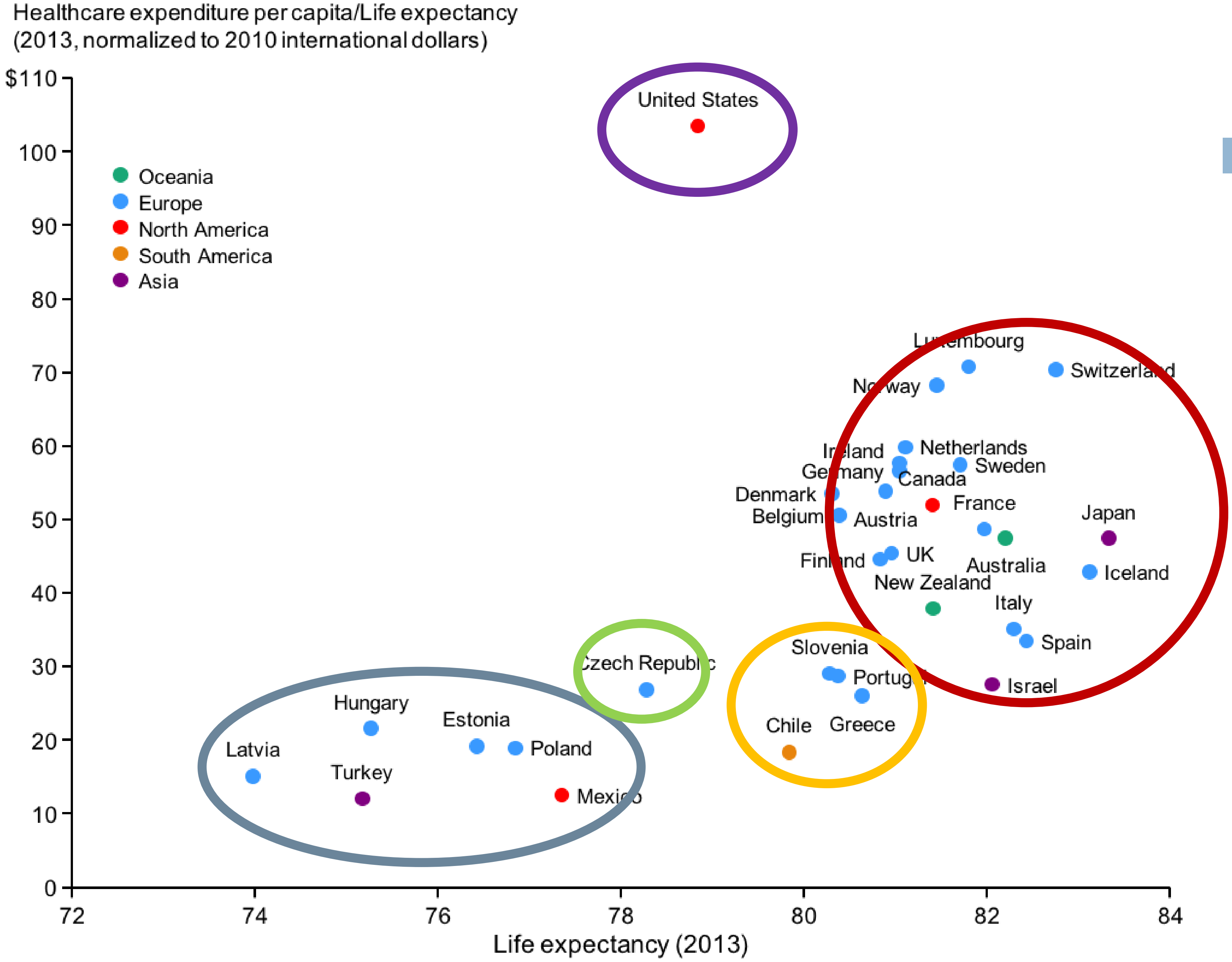
- By the way:
- It also helps if the documents are clustered
- We can cluster an initial collection of documents,
  - e.g., based on the BoW-model
- New documents can be assigned to a cluster

Applications of clustering in search

- Clustering of results
  - Interest: Fruit, Term: Apple
    What if the 100 best ranked results
    are computer related?
- Search: step-wise refinement
- And more, see IR-book

Application

- Visualization
- Data analysis
  - (Data science)



Source: Our World in Data

Clustering methods

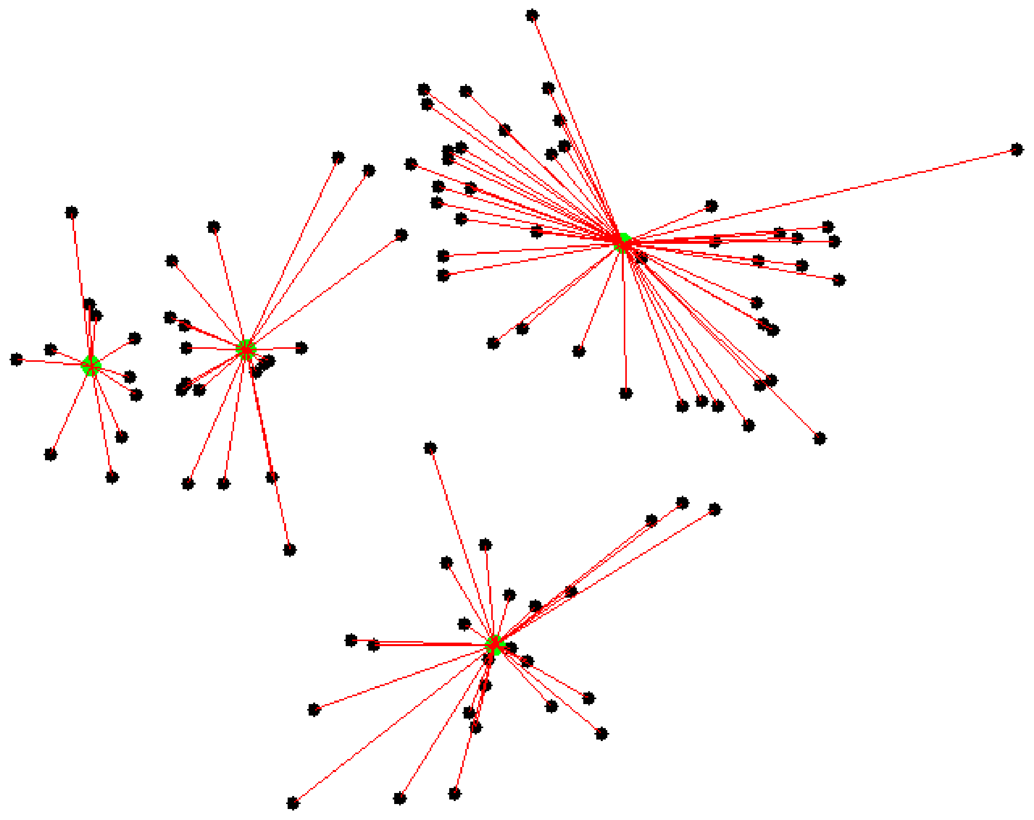
- Hierarchical
  - Creates a tree structure of hierarchically nested clusters.
- Flat
  - Tries to directly decompose the data into a set of clusters.
  - What we will focus on.
  - Given a set of objects $O = \{o1, . . . , on\}$,
    construct a set of clusters $C = \{c1, . . . , ck\}$,
    where each object oi is assigned to a cluster cj .
  - We will consider one algorithm: k-means clustering

K-means clustering

1. Decide on the number of clusters: k
2. Choose a set of arbitrary centroids: $\mu_1, \mu_2, \dots, \mu_k$
3. For each item, x, in the training data,
   - find the nearest centroid μ_i, and assign x to class C_i
4. For each resulting class C_i, calculate and find the new centroid μ_i.
5. Classify each item according to the new centroids
6. Repeat from 4

Demo

- Many demos and videos on the net.
- I like this one:
  - http://shabal.in/visuals/kmeans/1.html
- Here is
  - another one
  - and one on youtube



Why does this work? How does this work?

- The goal is a mapping

  $\gamma: O \rightarrow C = \{C_1, C_2, \dots, C_k\}$
- We need a tool, F,
  - to measure the performance of γ
- The goal is to find a γ that optimizes F, in symbols

  $$\hat{\gamma} = \underset{\gamma}{\operatorname{argmax}} F(\gamma)$$
- F is called an objective function

- Several possible objectives:
  - High similarity (=small distance) within the clusters (intra-cluster)
  - Low similarity (high distance) between the clusters (inter-clusters)

Within cluster sum of squares (intra-cluster)

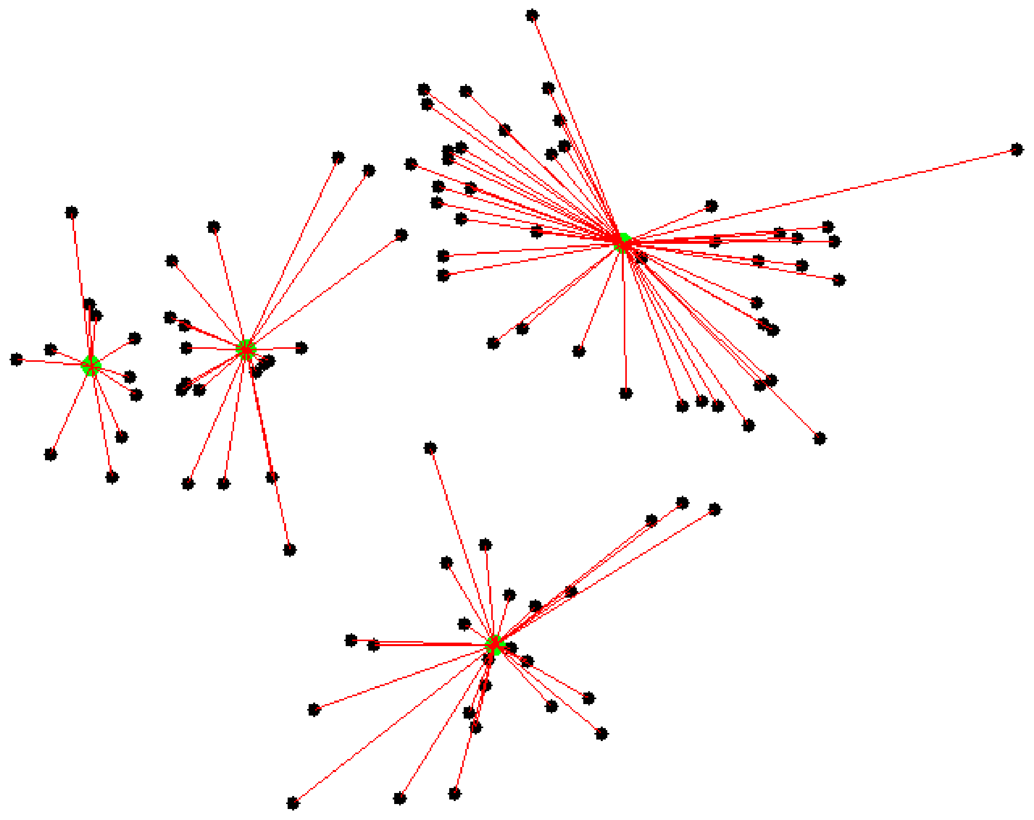
- For each cluster consider the sum of square distances:

$$SS_i = \sum_{x_j \in C_i} \left\| x_j - \mu_i \right\|^2$$

- Sum over all classes

$$WCSS = \sum_{i=1}^{k} SS_i$$

- To optimize F, is to find the γ that yields the smallest $WCSS$



Applied to *k*-means

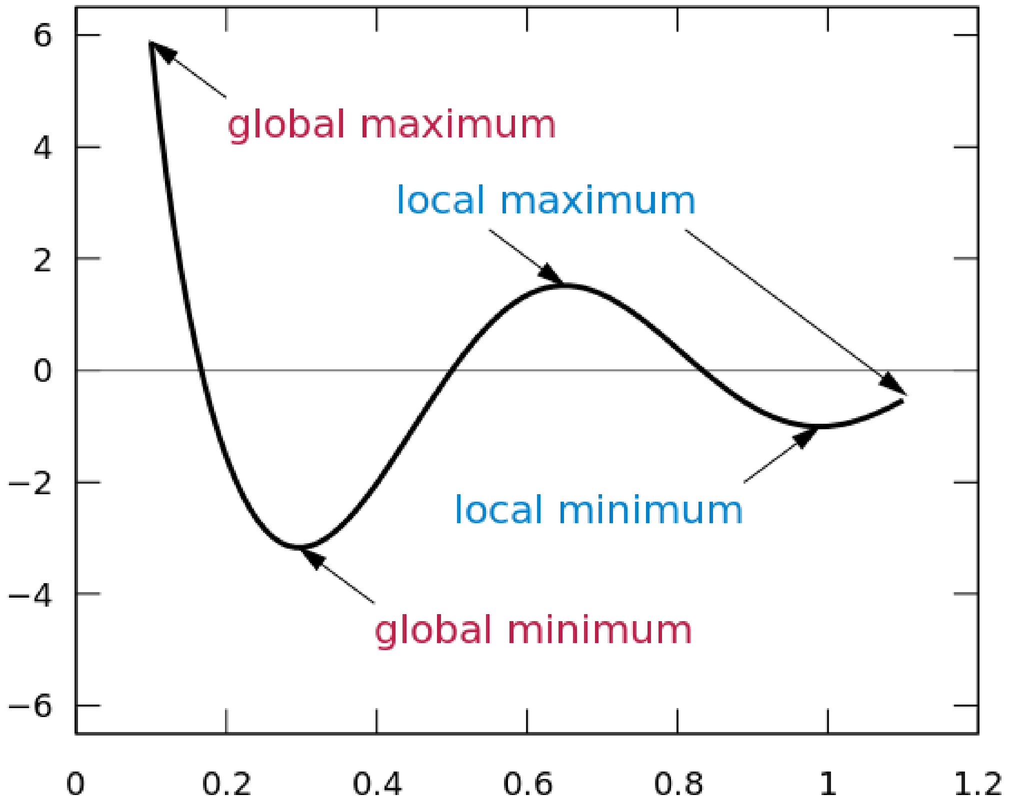
- For each iteration:

$$WCSS_{i+1} \leq WCSS_i$$

- Because:
  - Given a class, C_i, the recalulated centroid is the unique point in space that minimizes SS_i
  - If an item is moved from one class to another, its centroid-distance decreases
- Possible stopping criteria:
  - Fixed number of iterations
  - Clusters or centroids are unchanged between iterations.
  - Threshold on the decrease of the objective function (absolute or relative to previous iteration)

Properties of *k*-means

- The time complexity is linear, $O(kn)$
- Guaranteed to converge, but not to find the global optimal solution:
  - Depends on choice of initial centroids



Comments

'Seeding'

- We initialize the algorithm by choosing random seeds that we use to compute the first set of centroids, e.g:
  - pick k random objects from the collection;
  - pick k random points in the space;
  - pick k sets of m random points and compute centroids for each set; etc.
- The seeds can have a large impact on the resulting clustering.
- Outliers are troublemakers.

- No prescribed way to choose *k*.
  - In particular, more *k*-s will always give better WCSS without being intuitively better.

Intrinsic evaluation of clustering

With labeled gold-data

- Run *k*-means on the gold set (without the labels).
- Compare the clusters to the classes:
  - Purity: a good cluster will have all members from the same class

Without using gold data

- We can use some intra-cluster or inter-cluster measure,
  - E.g., WCSS to compare which intial choice of centroids is better in *k*-means

Extrinsic evaluation

- See which clustering (or lack of clustering) yields the best results in a larger task
- For example: two versions of a recommender system, and measure some of:
  - User satisfaction
  - How many recommended articles they read, or click on
  - Improvement in sales

Flat Clustering: The good and the bad

Pros

- Conceptually simple, and easy to implement.
- Efficient. Typically linear in the number of objects.

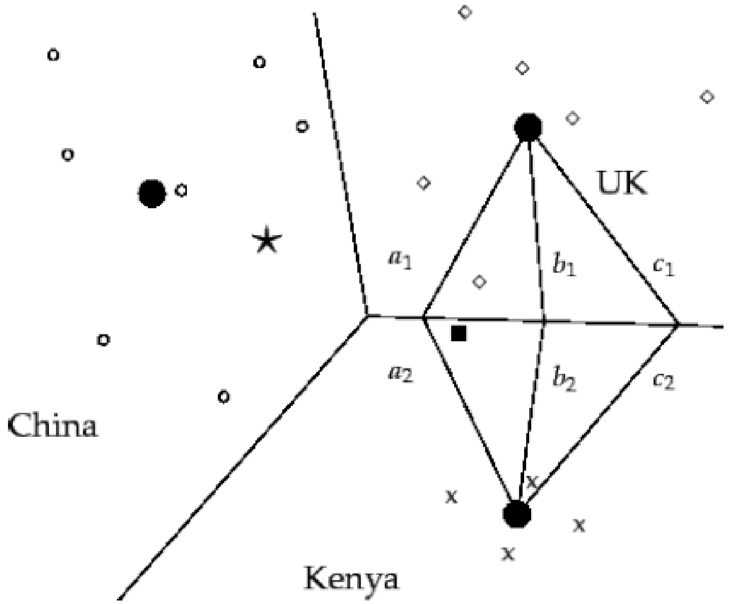
Cons

- The dependence on random seeds as in k-means makes the clustering non-deterministic.
- The number of clusters k must be pre-specified. Often no principled means of *a priori* specifying k.
- Not as informative as the more structured clusterings produced by hierarchical methods.

- In general; often difficult to evaluate clustering.

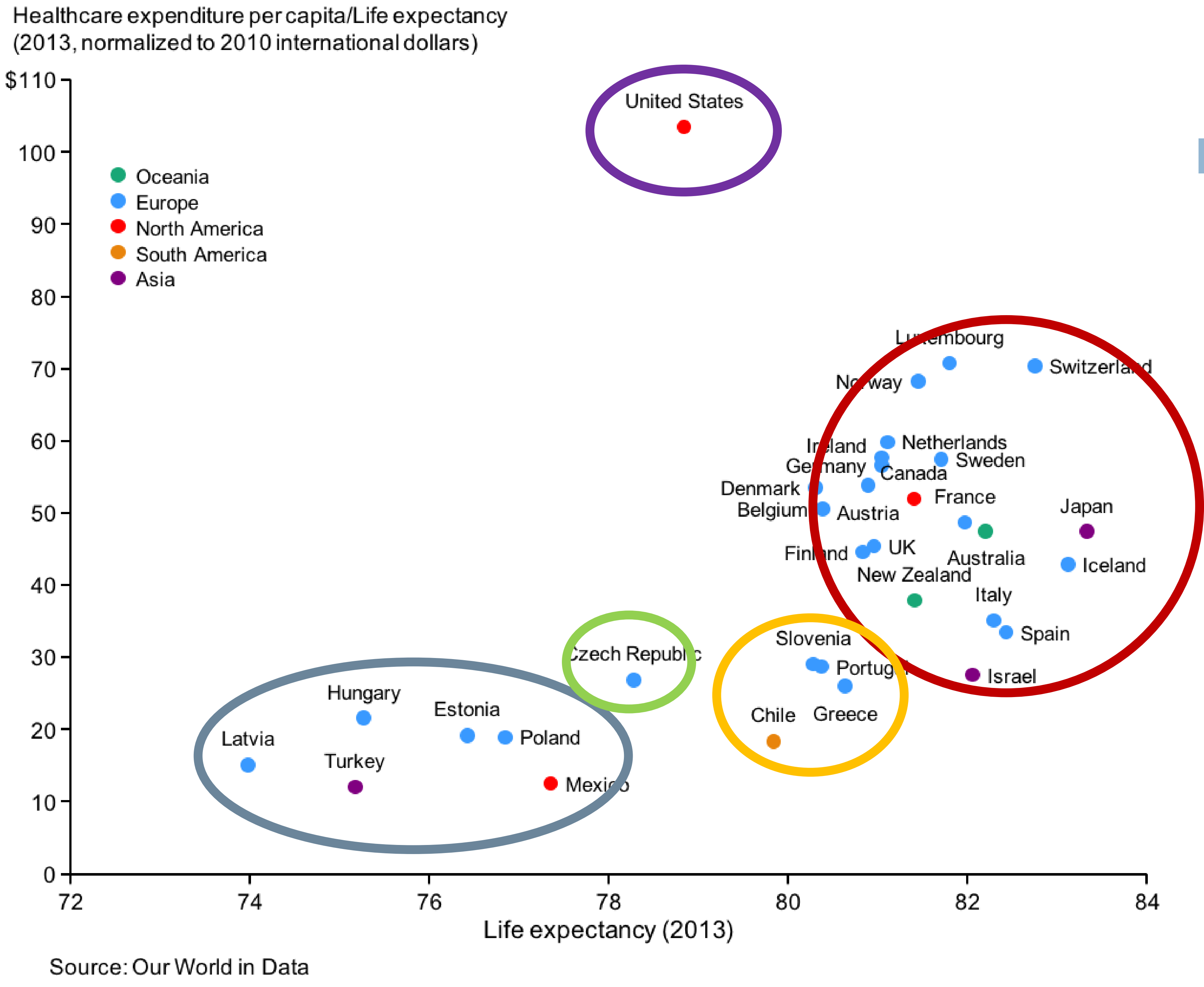
Connecting the dots

- We have seen how Rocchio classification can be thought of as a 1-Nearest-Neighbor classification with respect to the centroids.
- Note how k-means clustering can be thought of as performing Rocchio classification in each iteration.



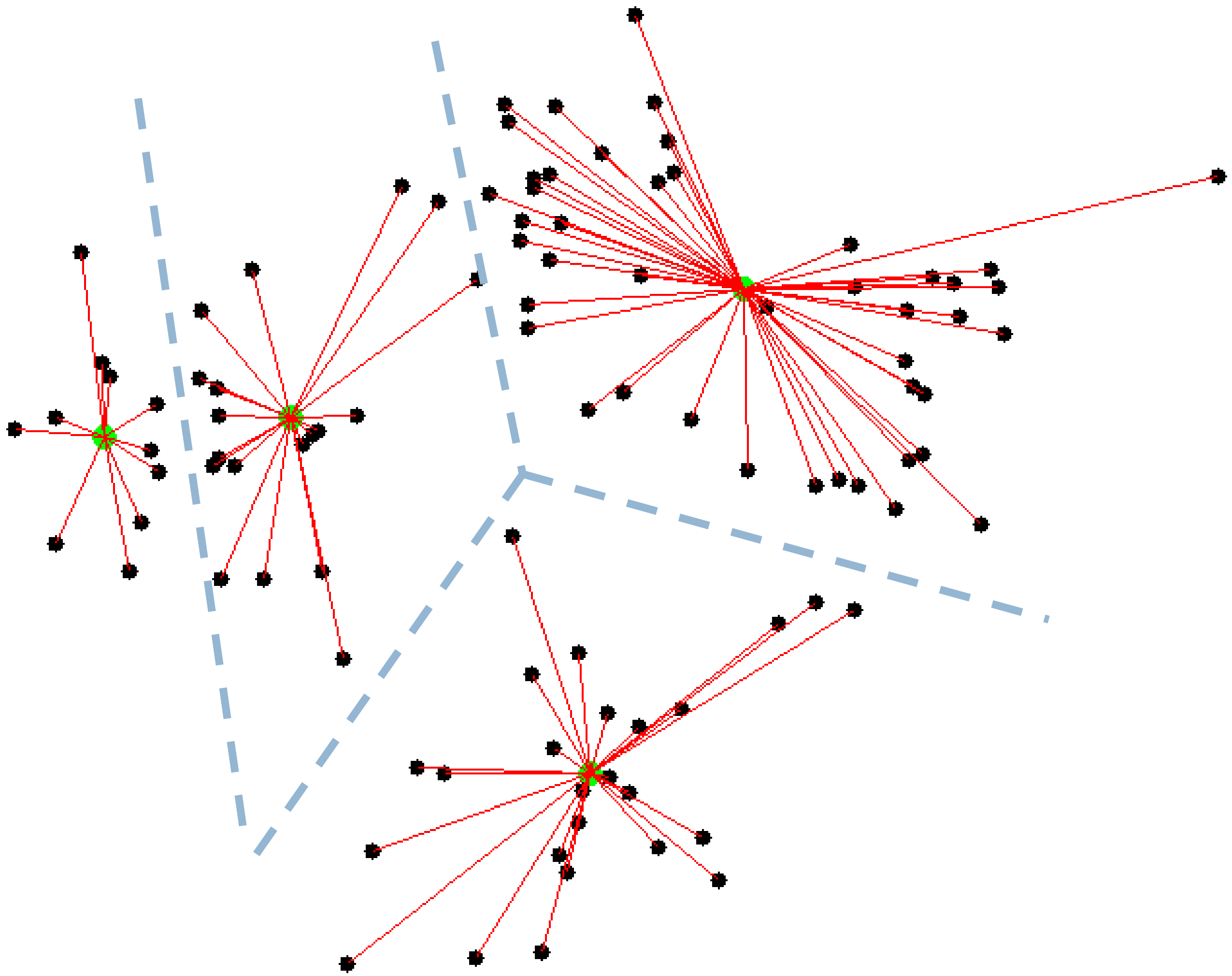
Limitations

- Similar underling assumptions as the Rocchio classifier
- Assumes regions with the same diameter



Limitations

- Similar underling assumptions as the Rocchio classifier
- A Voronoi cell for each cluster, defined by the centroid



BoW representations of text

- So far we've been assuming BoW features for representing documents.
- Often also used for representing other units of texts, like sentences.
- Many sentence-classification tasks in NLP.
- Example: polarity classification (part of sentiment analysis).

I was impressed, this was not bad!

$$\Rightarrow$$

{was, was, !, not, I, impressed, bad, this }

- What is missing with a BoW representation?

Dealing with compositionality

I was impressed, this was not bad!

$\neq$

I was not impressed, this was bad!

- Will have the same BoW representation! :(
- A simplistic but much-used approximation to capture ordering constraints: n-grams (typically bigrams and trigrams).
- Ordered sub-sequences of n words.

{was, was, !, not, I, impressed, bad, this }

vs.

{'I was', 'was impressed' ... 'was not', 'not bad', 'bad, !' }

No information sharing

- No information sharing between features.
- All features are equally distinct.

- The pizza was great
- The margeritha was awesome
- The dog was sick

- Would be nice if our BoW representations knew that *pizza* and *margeritha* are similar to each other (but not to *dog*).
- We've discussed one possible approach in this lecture... What?
- Will return to this issue next week.

Next lecture

- Focus on *words* rather than *documents*.
- Distributional models of word meaning (lexical semantics).
- Semantic spaces: Vector space models of word meaning
- Example tasks for evaluating word vectors