Learning from Data
Lecture 4: Vector Space Models
K-Nearest Neighbor
(+ Support Vector Machines)

Malvina Nissim and Johannes Bjerva
m.nissim@rug.nl, j.bjerva@rug.nl

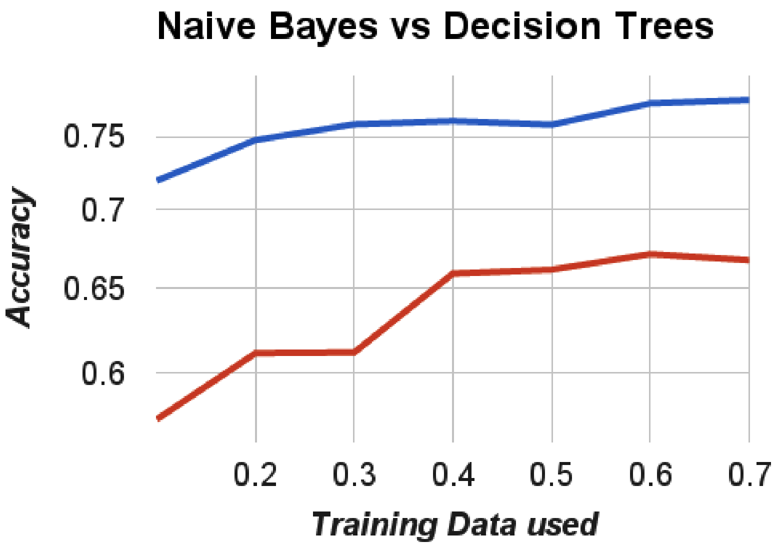
ESSLLI, 25 August 2016

Getting the updated code

START AS SOON AS YOU ENTER THE ROOM! (PLEASE)

- Approach 1: Use *git* (updateable, recommended if you have *git*)
  1. In your terminal, type: 'git clone https://github.com/bjerva/esslli-learning-from-data-students.git'
  2. Followed by 'cd esslli-learning-from-data-students'
  3. Whenever the code is updated, type: 'git pull'
- Approach 2: Download a zip (static)
  1. Download the zip archive from: `https://github.com/bjerva/esslli-learning-from-data-students/archive/master.zip`
  2. Whenever the code is updated, download the archive again.

Reflections and Concepts


Naive Bayes vs Decision Trees
Accuracy
0.75
0.7
0.65
0.6
0.2
0.3
0.4
0.5
0.6
0.7
Training Data used

Naive Bayes vs Decision Tree

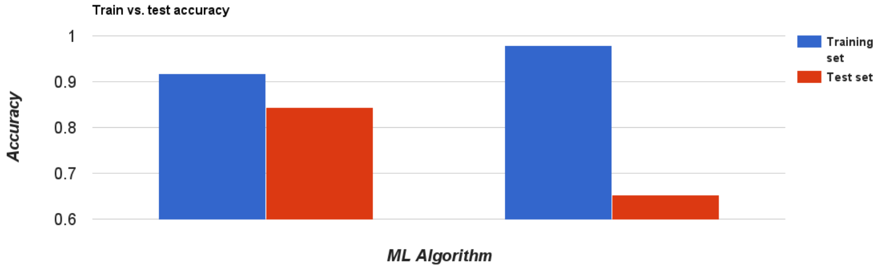
- Naive Bayes is rather robust to irrelevant features: irrelevant features cancel each other without affecting results
  (Decision Trees can heavily suffer from this.)
- Naive Bayes is good in domains with many equally important features (Decision Trees suffer from fragmentation in such cases, especially with little data)
- Most decision-tree algorithms only examine a single field at a time. This leads to rectangular classification boxes that may not correspond well with the actual distribution of records in the decision space.
  - The fact that decision trees require that features be checked in a specific order limits their ability to exploit features that are relatively independent of one another.
  - Naive Bayes overcomes this limitation by allowing all features to act "in parallel."

Decision Trees – Strengths

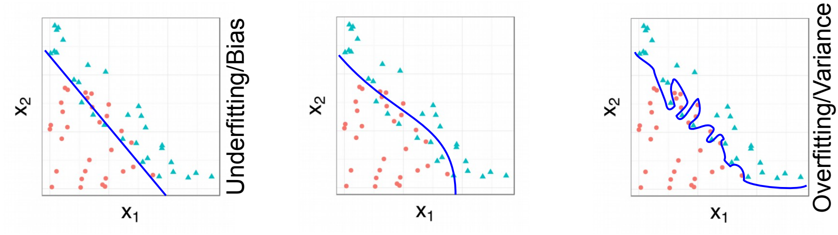
- decision trees are able to generate understandable rules
- decision trees provide a clear indication of which features are most important for prediction
- (once available) decision trees perform classification without much computation
- they can be a great tool for understanding your dataset

Decision Trees – Weaknesses

- decision trees are prone to errors in classification problems with many classes and relatively small number of training examples.
  - note: since each branch in the decision tree splits the training data, the amount of training data available to train nodes lower in the tree can become quite small.
- decision trees can be computationally expensive to train (need to compare all possible splits)
- decision trees are very prone to overfitting


Train vs. test accuracy
1
0.9
0.8
0.7
0.6
Accuracy
ML Algorithm
Training set
Test set

Over- and Underfitting: Classification Example



Underfitting/Bias

- Error on training set is high
- *Simple* hypothesis fails to generalize to new examples

Overfitting/Variance

- Error on training set is low
- *Complex* hypothesis fails to generalize to new examples

Fixes

- high variance (overfitting)
  - get more training examples
  - reduce number of features
  - (prune tree)
  - regularization
    - penalty for model complexity
    - aim at reducing training error while keeping validation error constant (NB: cross-validation)
    - works well for lots of features where each contributes a little bit to predicting y
- high bias (underfitting)
  - get more features

Generative vs Discriminative

task: to determine the language that someone is speaking

- generative approach: learn each language and determine to which language the speech belongs to
- discriminative approach: determine the linguistic differences without learning any language

Generative vs Discriminative

task: to determine the language that someone is speaking

- generative approach: learn each language and determine to which language the speech belongs to
  - *generative* because it can simulate values of any variable in the model
  - example algorithm: **Naive Bayes**
- discriminative approach: determine the linguistic differences without learning any language
  - directly estimate posterior probabilities
  - no attempt to model underlying probability distributions
  - example algorithm: **decision tree, k-nearest neighbor**

Vector space model

the representation of a set of documents as vectors in a common space

(parts of the following slides are based on slides by Yannick Parmentier)

Vector space model

- Each term t of the dictionary is considered as a *dimension*
- A document d can be represented by the weight of each vocabulary term:

$$\vec{V}(d) = (w(t_1, d), w(t_2, d), \ldots, w(t_n, d))$$

- Note that also raw frequency could be used (that's what we are doing)

Vector space model

- Each term t of the dictionary is considered as a *dimension*
- A document d can be represented by the weight of each vocabulary term:
$$\vec{V}(d) = (w(t_1, d), w(t_2, d), \ldots, w(t_n, d))$$
- Note that also raw frequency could be used (that's what we are doing)

representation of three documents using term raw frequencies: d_1, d_2, d_3

	d_1	d_2	d_3
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

Vector space model

- Each term t of the dictionary is considered as a *dimension*
- A document d can be represented by the weight of each vocabulary term:
$$\vec{V}(d) = (w(t_1, d), w(t_2, d), \ldots, w(t_n, d))$$
- Note that also raw frequency could be used (that's what we are doing)

representation of three documents using term raw frequencies: d_1, d_2, d_3

	d_1	d_2	d_3
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

- **Question**:
how do we compute the **similarity between documents** d_1**,** d_2**,** d_3?

Vector normalization and similarity

- Similarity between vectors
  $\rightarrow$ inner product $\vec{V}(d_1) \cdot \vec{V}(d_2)$

Vector normalization and similarity

- Similarity between vectors
  → inner product $\vec{V}(d_1) \cdot \vec{V}(d_2)$
- But wait, first: What about the length of a vector?
  Longer documents will be represented with longer vectors, but that does not mean they are more important
- Euclidian normalization (vector length normalization):

$$\vec{v}(d) = \frac{\vec{V}(d)}{\|\vec{V}(d)\|} \qquad \text{where } \|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Vector normalization and similarity

- Similarity between vectors
  $\rightarrow$ inner product $\vec{V}(d_1) \cdot \vec{V}(d_2)$
- Similarity given by the *cosine* measure between normalized vectors:

$$sim(d_1, d_2) = \vec{v}(d_1) \cdot \vec{v}(d_2)$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
  let's backtrack this:
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

Example (Manning et al, 09)

- $sim(d1, d2) \; = \; \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

Example (Manning et al, 09)

- $sim(d1, d2) \;=\; \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\vec{V}(d1)\| = \sqrt{115^2 + 10^2 + 2^2}$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\vec{V}(d1)\| = \sqrt{115^2 + 10^2 + 2^2}$$

$$\vec{v}(d1_1) = \frac{115}{\sqrt{115^2+10^2+2^2}} = 0.996$$

$$\vec{v}(d1_2) = \frac{10}{\sqrt{115^2+10^2+2^2}} = 0.087$$

$$\vec{v}(d1_3) = \frac{2}{\sqrt{115^2+10^2+2^2}} = 0.017$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\vec{V}(d2)\| = \sqrt{58^2 + 7^2 + 0}$$

$$\vec{v}(d2_1) = \frac{58}{\sqrt{58^2+7^2+0}} = 0.993$$

$$\vec{v}(d2_2) = \frac{7}{\sqrt{58^2+7^2+0}} = 0.120$$

$$\vec{v}(d2_3) = 0$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$
- normalising for length:

$$\vec{v}(d_i) = \frac{\vec{V}(d_i)}{\|\vec{V}(d)\|}$$

- Euclidean length:

$$\|\vec{V}(d)\| = \sqrt{\sum_{i=1}^{n} \vec{V}_i^2(d)}$$

vocabulary	$d1$	$d2$	$d3$
1: affection	115	58	20
2: jealous	10	7	11
3: gossip	2	0	6

$$\|\vec{V}(d3)\| = \sqrt{20^2 + 11^2 + 6^2}$$

$$\vec{v}(d3_1) = \frac{20}{\sqrt{20^2+11^2+6^2}} = 0.847$$

$$\vec{v}(d3_2) = \frac{11}{\sqrt{20^2+11^2+6^2}} = 0.466$$

$$\vec{v}(d3_3) = \frac{6}{\sqrt{20^2+11^2+6^2}} = 0.254$$

Example (Manning et al, 09)

- $sim(d1, d3) = \vec{v}(d1) \cdot \vec{v}(d3)$
- $\vec{v}(d1) \cdot \vec{v}(d3) = \sum_{i=1}^{n} d1_i d3_i$

$$\vec{v}(d1_1) = \frac{115}{\sqrt{115^2+10^2+2^2}} = 0.996$$
$$\vec{v}(d1_2) = \frac{10}{\sqrt{115^2+10^2+2^2}} = 0.087$$
$$\vec{v}(d1_3) = \frac{2}{\sqrt{115^2+10^2+2^2}} = 0.017$$

[i=1] $0.996 * 0.847+$
[i=2] $0.087 * 0.466+$
[i=3] $0.017 * 0.254 =$
$= 0.888$

$$\vec{v}(d3_1) = \frac{20}{\sqrt{20^2+11^2+6^2}} = 0.847$$
$$\vec{v}(d3_2) = \frac{11}{\sqrt{20^2+11^2+6^2}} = 0.466$$
$$\vec{v}(d3_3) = \frac{6}{\sqrt{20^2+11^2+6^2}} = 0.254$$

Example (Manning et al, 09)

- $sim(d1, d2) = \vec{v}(d1) \cdot \vec{v}(d2)$
- $\vec{v}(d1) \cdot \vec{v}(d2) = \sum_{i=1}^{n} d1_i d2_i$

$$\vec{v}(d1_1) = \frac{115}{\sqrt{115^2+10^2+2^2}} = 0.996$$
$$\vec{v}(d1_2) = \frac{10}{\sqrt{115^2+10^2+2^2}} = 0.087$$
$$\vec{v}(d1_3) = \frac{2}{\sqrt{115^2+10^2+2^2}} = 0.017$$

[i=1] $0.996 * 0.993 +$
[i=2] $0.087 * 0.120 +$
[i=3] $0.017 * 0.000 =$
$= 0.999$

$$\vec{v}(d2_1) = \frac{58}{\sqrt{58^2+7^2+0}} = 0.993$$
$$\vec{v}(d2_2) = \frac{7}{\sqrt{58^2+7^2+0}} = 0.120$$
$$\vec{v}(d2_3) = 0$$

Example (Manning et al, 09)

summing up:

dictionary	$\vec{v}(d_1)$	$\vec{v}(d_2)$	$\vec{v}(d_3)$
affection	0.996	0.993	0.847
jealous	0.087	0.120	0.466
gossip	0.017	0	0.254

$$sim(d_1, d_2) = 0.999$$
$$sim(d_1, d_3) = 0.888$$

New documents

- each new document n is represented using vectors in the same way

	d_1	d_2	d_3	n
affection	115	58	20	0
jealous	10	7	11	1
gossip	2	0	6	1

- $sim(n, d) = \vec{v}(n) \cdot \vec{v}(d)$

New documents

- each new document n is represented using vectors in the same way

	d_1	d_2	d_3	n
affection	115	58	20	0
jealous	10	7	11	1
gossip	2	0	6	1
class	A	A	B	?

- $sim(n, d) = \vec{v}(n) \cdot \vec{v}(d)$

New documents

- each new document *n* is represented using vectors in the same way

	d_1	d_2	d_3	n
affection	115	58	20	0
jealous	10	7	11	1
gossip	2	0	6	1
class	A	A	B	B

- $sim(n, d) = \vec{v}(n) \cdot \vec{v}(d)$
- with $n =< jealous, gossip >$
  we obtain:

$$\begin{aligned}\vec{v}(n) \cdot \vec{v}(d_1) &= 0.074\\ \vec{v}(n) \cdot \vec{v}(d_2) &= 0.085\\ \vec{v}(n) \cdot \vec{v}(d_3) &= 0.509\end{aligned}$$

Classifying new documents

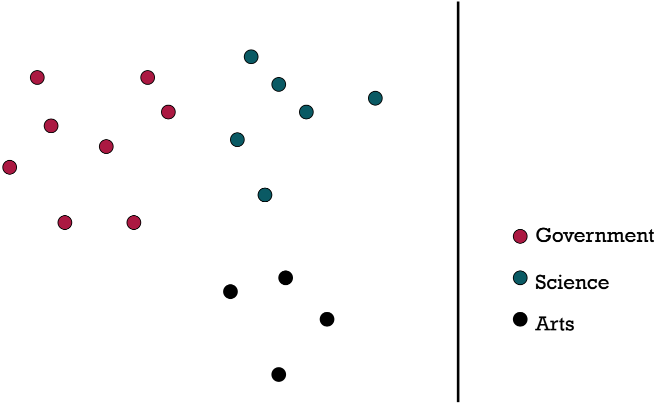
- Basic idea: similarity cosines between the new document's vector and each classified document's vector;
- NB: the decisions of many vector space classifiers are based on a notion of *distance*.

  There is a direct correspondence between cosine similarity and Euclidean distance for length-normalised vectors, so it rarely matters whether the relatedness of two documents is expressed in terms of similarity or distance

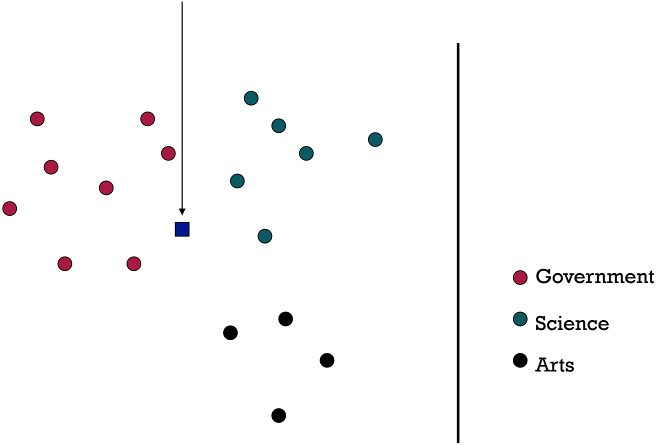
Classification Using Vector Spaces

- in vector space classification, training set corresponds to a labeled set of points (vectors)
- premise 1: documents in the same class form a contiguous region of space
- premise 2: documents from different classes dont overlap (much)
- learning a classifier = build surfaces to delineate classes in the space

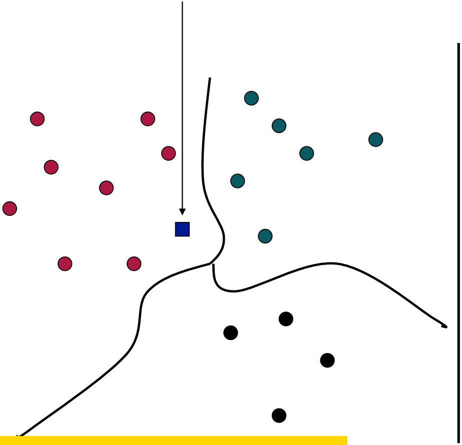
Documents in a Vector Space



Test Document of what class?



Test Document = Government



Is this similarity hypothesis true in general?

Our focus: how to find good separators

k-nearest neighbor

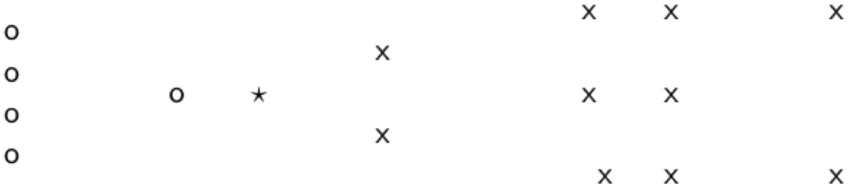
(some slides by Manning et al (2009), Intro to IR)

k Nearest Neighbor Classification

K-NN (kNN) = K-Nearest Neighbor

to classify a document d:

- define K-neighborhood as the k nearest neighbors of d
- pick the **majority class label in the K-neighborhood**



can you classify the *star*?
what is it most similar/close to?

K-Nearest Neighbor

- learning: just store the labeled training examples in dataset D (does not compute anything beyond storing the examples!)
- testing instance x (with $K = 1$):
  - compute similarity between x and all examples in D.
  - assign x the category of **the most similar example in D**.

K-Nearest Neighbor

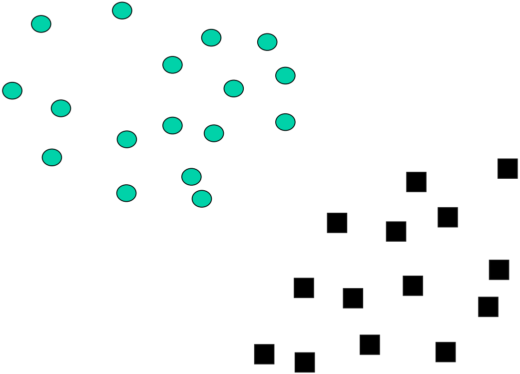
- using only the closest example (1NN) subject to errors due to:
  - a single atypical example
  - noise (i.e., an error) in the category label of a single training example
- more robust: find the k examples and return the majority category of these k. How to determine the best k?
  - in binary classification k is typically odd to avoid ties (often 3 or 5).
  - experience/knowledge about a certain classification problem
  - picking best K on development set or via cross-validation

K-NN discussion

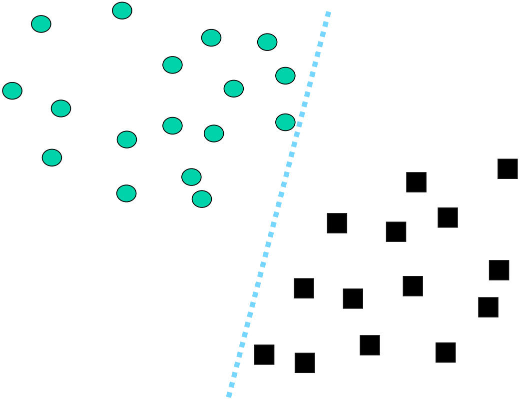
- + no training necessary
- – expensive at test time
- + it scales well with large number of classes
- – classes can influence each other (small changes to one class can have ripple effect)
- classification based only on the nearby K instances (anything farther away is ignored)

Support Vector Machines

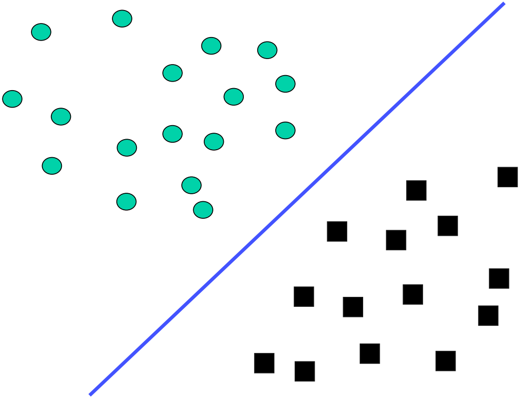
Which of the linear separators is optimal?



Best Linear Separator?



Best Linear Separator?



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)

- it lets as few instances of a class to be on the wrong side of the border as possible
- it creates the largest possible *no-man's land* between the (two) classes ("large-margin classifier": largest possible margin between decision boundary and any data point)
- some training instances are more important than others: the instances that make up the boundary are the **support vectors**
- how many instances can one allow to be on "wrong side"? (complexity and parameter setting)

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)

- it lets as few instances of a class to be on the wrong side of the border as possible
- it creates the largest possible *no-man's land* between the (two) classes ("large-margin classifier": largest possible margin between decision boundary and any data point)
- some training instances are more important than others: the instances that make up the boundary are the **support vectors**
- how many instances can one allow to be on "wrong side"? (complexity and parameter setting)

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)

- it lets as few instances of a class to be on the wrong side of the border as possible
- it creates the largest possible *no-man's land* between the (two) classes ("large-margin classifier": largest possible margin between decision boundary and any data point)
- some training instances are more important than others: the instances that make up the boundary are the **support vectors**
- how many instances can one allow to be on "wrong side"? (complexity and parameter setting)

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)

- it lets as few instances of a class to be on the wrong side of the border as possible
- it creates the largest possible *no-man's land* between the (two) classes ("large-margin classifier": largest possible margin between decision boundary and any data point)
- some training instances are more important than others: the instances that make up the boundary are the **support vectors**
- how many instances can one allow to be on "wrong side"? (complexity and parameter setting)

practice!

Practice with k-NN and SVM

options to play with:

- −−k N (number of nearest neighbours)
- −−max-train-size N (maximum number of training samples to look at)
- −−nchars N

visualisation options:

- −−cm (print confusion matrix + classification report)
- −−plot (shows CM)

example runs:

```
python run_experiment.py --csv data/langident.csv --nchars 1
--algorithms knn --k 1 --cm
python run_experiment.py --csv data/trainset-sentiment-extra.csv
--nchars 1 --algorithms svm --cm
```

- with many features (e.g. more than 2000), testing with k-NN will take a long time.
- the small (three languages) version of the langident dataset is `langident-small.csv`

Tomorrow's class

- practical work on different datasets
  - from us
  - from you
- wrap up

Practical session

- presentation of tasks and datasets (one slide per task/dataset)
- running experiments in groups
- reporting on experiments (a couple of minutes per group, depending on how many groups there are)
  - task
  - dataset
  - set up
  - features
  - classifier
  - results
  - any reflections

please, send us your data TODAY!

(and see you tomorrow)