Machine Learning for natural language processing

Classification: k nearest neighbors

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2016

Introduction

- Classification = supervised method for classifying an input, given a finite set of possible classes.
- Today: Vector-based document classification.

Jurafsky & Martin (2015), chapter 15, Jurafsky & Martin (2009) chapter 20 and Manning et al. (2008), chapter 14

Table of contents

Motivation

- We can characterize documents by large vectors of real-valued features.
- Features are for instance words of a given vocabulary and their values reflect their occurrences in the document.
- This gives us a vector-space model of document classes.

Vectors and documents

Term-document matrix: Let V be our vocabulary, D our set of documents. A term-document matrix characterizing D with respect to V is a $|D| \times |V|$ where the cell for v_i and d_j contains the number of occurrences of v_i in d_i.

Each column in this matrix gives a vector of dimension $|V|$ that represents a document.

This is again the bag-of-words representation of documents, except that V does not contain all words from the documents in D.

Vectors and documents

Example from Jurafsky & Martin (2015), chapter 19

Term-document matrix for four words in four Shakespeare plays:

	As You Like It	Twelth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

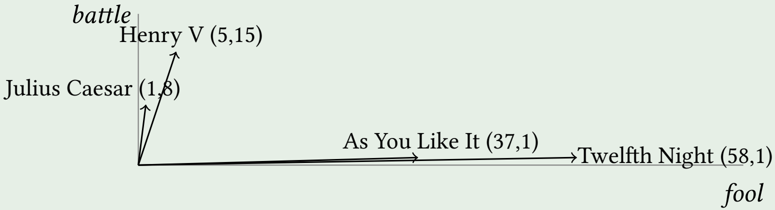
In real applications, we usually have $10.000 \leq |V| \leq 50.000$.

Vectors and documents

General idea: Two documents are similar if they have similar vectors.

Example from Jurafsky & Martin (2015), chapter 19 continued

Take only dimensions *fool* and *battle* from the previous matrix:



Vectors and documents

The above frequency counts are not the best measures for associations between terms and documents:

- A highly frequent word like *the* will not tell us anything about the class of a document since it occurs (more or less) equally frequent in all documents.
- Rare words also might be equally rare in all types of documents.
- The frequencies of such words, which are independent from the document class, should weigh less than the frequencies of words that are very typical for certain classes.

We need a weighting scheme that takes this into account.

Vectors and documents

A standard weighting scheme for term-document matrices in information retrieval is **tf-idf**:

tf-idf

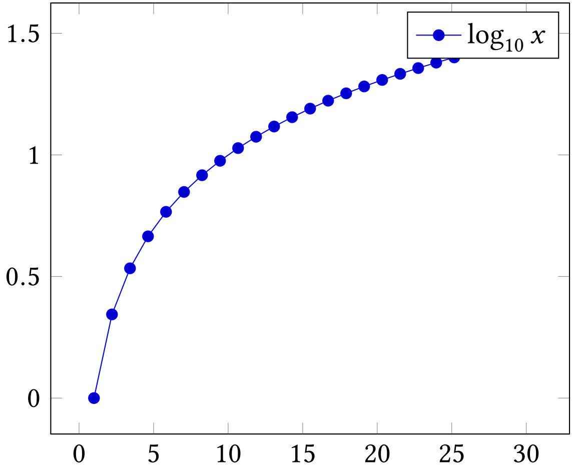
- term frequency: tf_{td} = frequency of term t in document d. This can be simply the count of t in d.
- document frequency: df_t = number of documents in which term t occurs
- inverse document frequency: $idf_t = \log\left(\frac{|D|}{df_t}\right)$
- tf-idf: $w_{td} = tf_{td}\, idf_t$

Terms occurring in lesser documents are more discriminative. Therefore their counts are weighted with a higher *idf* factor. Since $|D|$ is usually large, we use the log of the inverse document frequency.

Note, however, that terms occurring in all documents get a weight 0.

Vectors and documents

Reminder:



Measuring vector distance/similarity

Each document d is characterized with respect to a vocabulary $V = \{t_1, \ldots, t_{|V|}\}$ by a vector, for instance

$$\langle tf_{t_1,d}\, idf_{t_1}, \ldots, tf_{t_{|V|},d}\, idf_{t_{|V|}} \rangle$$

if we use tf-idf.

In order to compare documents, we need some metric for the distance of two vectors. One possibility is the **Euclidian distance**.

The length of a vector is defined as

$$|\vec{v}| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

We define the Euclidian distance of two vectors $\vec{v}$, $\vec{w}$ as

$$|\vec{v} - \vec{w}| = \sqrt{\sum_{i=1}^{n} (v_i - w_i)^2}$$

Measuring vector distance/similarity

In order to use the Euclidian distance, we first have to normalize the vectors, i.e., we use

$$|\frac{\vec{v}}{|\vec{v}|} - \frac{\vec{w}}{|\vec{w}|}| = \sqrt{\sum_{i=1}^{n}(\frac{v_i}{|\vec{v}|} - \frac{w_i}{|\vec{w}|})^2}$$

Alternatively, we can also use a metric for the similarity of vectors.

The most common metric used in NLP for vector similarity is the **cosine** of the angle between the vectors.

Measuring vector distance/similarity

Underlying idea: We use the dot product from linear algebra as a similarity metric: For vectors $\vec{v}$, $\vec{w}$ of dimension n,

$$\vec{v} \cdot \vec{w} = \sum_{i=1}^{n} v_i w_i$$

Example: dot product

Consider the following term-document matrix:

	d_1	d_2	d_3	d_4
t_1	1	2	8	50
t_2	2	8	3	20
t_3	30	120	0	2

- d_1, d_2: $\vec{v}_{d_1} \cdot \vec{v}_{d_2} = \langle 1, 2, 30 \rangle \cdot \langle 2, 8, 120 \rangle = 3618$
- d_1, d_3: $\vec{v}_{d_1} \cdot \vec{v}_{d_4} = 200$
- But: $\vec{v}_{d_2} \cdot \vec{v}_{d_4} = 500$ and $\vec{v}_{d_3} \cdot \vec{v}_{d_4} = 460$

Measuring vector distance/similarity

The dot product favors long vectors. Therefore, we use the **normalized dot product**, i.e., we divide by the lengths of the two vectors.

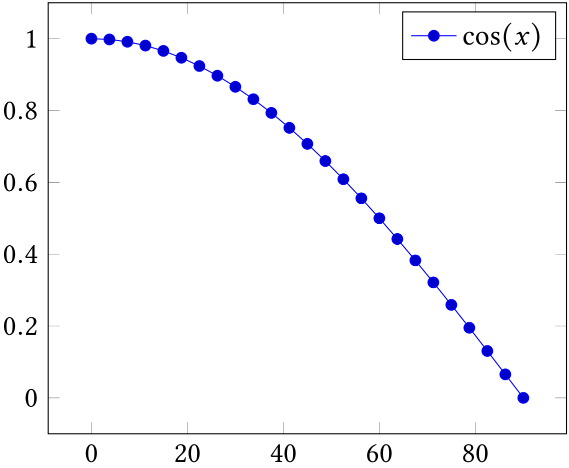
Cosine similarity metric

$$CosSim(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}|\,|\vec{w}|} = \frac{\sum_{i=1}^{n} v_i w_i}{\sqrt{\sum_{i=1}^{n} v_i^2}\sqrt{\sum_{i=1}^{n} w_i^2}} = \cos\phi$$

where ϕ is the angle between $\vec{v}$ and $\vec{w}$.

Measuring vector distance/similarity

Reminder: This is how the cosine looks like for angles between 0 and 90 (with all vector components ≥ 0, other angles cannot occur):



Measuring vector distance/similarity

Example: cosine similarity

Consider again the following term-document matrix:

	d_1	d_2	d_3	d_4
t_1	1	2	8	50
t_2	2	8	3	20
t_3	30	120	0	2
$\lvert\vec{v}_d\rvert$	30,08	120,28	8,54	53,89

Cosine values:

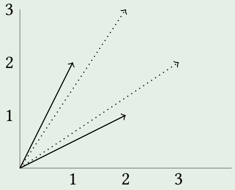
	d_1	d_2	d_3
d_2	1		
d_3	0.05	0.04	
d_4	0.09	0.08	1

(The cosine values for d_1, d_2 and for d_3, d_4 are rounded.)

Measuring vector distance/similarity

Cosine similarity is not invariant to shifts.

Cosine



Adding 1 to all components increases the cosine similarity.

The Euclidian distance, however, remains the same (if used without normalization).

An alternative is the **Pearson correlation**:

Pearson correlation

Let $\overline{v} = \frac{\sum_{i=1}^{n} v_i}{n}$ be the means of the vector components.

$$Corr(\vec{v}, \vec{w}) = \frac{\sum_{i=1}^{n}(v_i - \overline{v})(w_i - \overline{w})}{|\vec{v} - \overline{v}|\,|\vec{w} - \overline{w}|} = CosSim(\vec{v} - \overline{v}, \vec{w} - \overline{w})$$

where $\langle v_1, \ldots, v_n \rangle - c = \langle v_1 - c, \ldots, v_n - c \rangle$.

Measuring vector distance/similarity

Two other widely used similarity measures are **Jaccard** and **Dice**. The underlying idea is that we measure the overlap in the features. Therefore in both metrics, we include

$$\sum_{i=1}^{n} \min(v_i, w_i)$$

Jaccard similarity measure

$$simJaccard(\vec{v}, \vec{w}) = \frac{\sum_{i=1}^{n} \min(v_i, w_i)}{\sum_{i=1}^{n} \max(v_i, w_i)}$$

Dice is very similar except that it does not take the max but the average in the denominator:

Dice similarity measure

$$simDice(\vec{v}, \vec{w}) = \frac{2\sum_{i=1}^{n} \min(v_i, w_i)}{\sum_{i=1}^{n} v_i + w_i}$$

k nearest neighbors

In the following, we are concerned with the task of classifying a document by assigning it a label drawn from some finite set of labels.

Our training data consists of a set D of documents, each of which is in a unique class $c \in C$.

Documents are represented as vectors, i.e., our task amounts to classifying a new vector, given the classes of the training vectors.

Idea of k nearest neighbors (kNN): in order to classify a new document d, take the majority class of the k nearest neighbors of d in the training document vector space.

k nearest neighbors

Example: k nearest neighbors

Task: classify fictional texts in topic classes l (=*love*) and c (=*crime*).

Training:	Class l			Class c		new docs:	
terms	d_1	d_2	d_3	d_4	d_5	d_6	d_7
love	10	8	7	0	1	5	1
kiss	5	6	4	1	0	6	0
inspector	2	0	0	12	8	2	12
murderer	0	1	0	20	56	0	4
$\lvert\vec{v}_d\rvert$	11.36	10.05	8.06	23.35	56.58	8.06	12.69

Before comparing vectors, we should normalize them. (The definition of the cosine similarity above includes normalization.) I.e., every $\vec{v} = \langle v_1, \ldots v_n \rangle$ gets replaced with

$$\frac{\vec{v}}{|\vec{v}|} = \langle \frac{v_1}{|\vec{v}|}, \ldots, \frac{v_n}{|\vec{v}|} \rangle \text{ where } |\vec{v}| = \sqrt{\sum_{i=1}^{n} v_i^2}$$

k nearest neighbors

Example continued

Normalized data:

Training:	Class l			Class c		new docs:	
terms	d_1	d_2	d_3	d_4	d_5	d_6	d_7
love	0.88	0.8	0.87	0	0.02	0.62	0.08
kiss	0.44	0.6	0.5	0.04	0	0.74	0
inspector	0.18	0	0	0.51	0.14	0.25	0.95
murderer	0	0.1	0	0.86	0.99	0	0.32

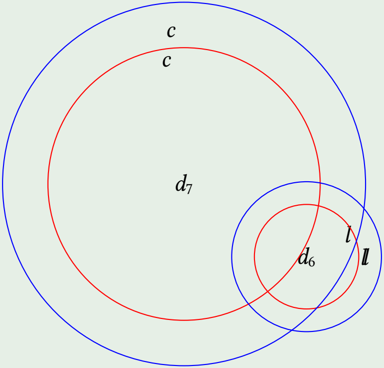
Euclidian distances:

	d_1	d_2	d_3	d_4	d_5
d_6	0.4	0.35	0.43	1.3	1.38
d_7	1.24	1.35	1.37	0.7	1.05

k nearest neighbors

Example continued

Visualization of the dimensions *love* and *murderer* including the k nearest neighbors of each of the new documents d_6, d_7 for $k = 1$ and $k = 3$:



For both ks, the majority class of the k nearest neighbors of d_6 is l while the one of d_7 is c.

k nearest neighbors

Some notation:

- $S_k(d)$ is the set of the k nearest neighbor documents of a new document d.
- $\vec{v}(d)$ is the vector of a document d.
- We define $I_c : S_k(d) \rightarrow \{0, 1\}$ for a class c and a document d as $I_c(d_t) = 1$ if the class of d_t is c, otherwise $I_c(d_t) = 0$.

Then kNN assigns the following score to each class c, given a document d that has to be classified:

$$score(c, d) = \sum_{d_t \in S_k(d)} I_c(d_t)$$

The classifier then assigns to d the class c from the set of classes C with the highest score:

$$\hat{c} = \arg\max_{c \in C} score(c, d)$$

k nearest neighbors

Choice of the k:

- usually an odd number;
- $k = 3$ or $k = 5$ are frequent choices but sometimes much larger values are also used;
- k can be choosen according to experiments during training on held-out data.

k nearest neighbors

kNN can be used as a probabilistic classifier: We can define

$$P(c|d) = \frac{\sum_{d_t \in S_k(d)} I_c(d_t)}{k}$$

Example continued

Euclidian distances:

	d_1	d_2	d_3	d_4	d_5
d_6	0.4	0.35	0.43	1.3	1.38
d_7	1.24	1.35	1.37	0.7	1.05

Probabilities:

$k = 1$ $\quad S_1(d_6) = \{d_2\}, P(l|d_6) = 1, P(c|d_6) = 0$

$\qquad S_1(d_7) = \{d_4\}, P(l|d_7) = 0, P(c|d_7) = 1$

$k = 3$ $\quad S_3(d_6) = \{d_1, d_2, d_3\}, P(l|d_6) = 1, P(c|d_6) = 0$

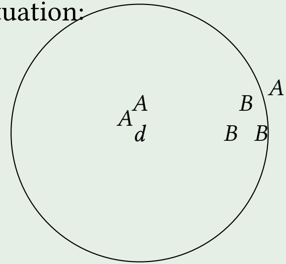
$\qquad S_3(d_7) = \{d_1, d_4, d_5\}, P(l|d_7) = \frac{1}{3}, P(c|d_7) = \frac{2}{3}$

k nearest neighbors

Problem: For $k > 1$, the majority vote does not take the individual distances of the k nearest neighbors to $\vec{v}(d)$ into consideration.

Example

$k = 5$, classes A and B, document d has to be classified. Assume we have the following situation:



The classifier would assign B. But the two A documents in $S_5(d)$ are much nearer to d than the three B documents.

k nearest neighbors

Possible solution:
weight the elements in $S_k(d)$ be their similarity to d.

This gives the following revised definition of the score, including the cos similarity measure:

$$score(c, d) = \sum_{d_t \in S_k(d)} I_c(d_t) \cos(\vec{v}(d_t), \vec{v}(d))$$

where $\vec{v}(d)$ is the vector of some document d.

References

Jurafsky, Daniel & James H. Martin. 2009. *Speech and language processing. an introduction to natural language processing, computational linguistics, and speech recognition* Prentice Hall Series in Articial Intelligence. Pearson Education International second edition edn.

Jurafsky, Daniel & James H. Martin. 2015. Speech and language processing. an introduction to natural language processing, computational linguistics, and speech recognition. Draft of the 3rd edition.

Manning, Christopher D., Prabhakar Raghavan & Hinrich Schütze. 2008. *Introduction to information retrieval.* Cambridge University Press.