Text Classification

Computational Methods for Text Analysis

Pestova Alena

HSE Saint-Petersburg

04.10.2021

Document-term matrix

Terms with the highest DF (document frequency)

Docs	вовочк	дет	класс	урок	учител	учительниц	школ
1	2	0	0	1	0	1	0
2	2	0	0	0	0	1	0
3	1	1	0	0	0	1	2
4	3	1	0	1	0	1	0
5	3	0	0	0	0	1	0
6	1	0	0	0	0	1	0

TF-IDF

Term-frequency-inversed document frequency - is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

$$TF \times IDF = tf \log_2 \frac{D}{df} \tag{1}$$

- tf term frequency - is the relative frequency of term t within document d (raw count of a term in a document / the total number of terms in document)
- df document frequency - the frequency of the word in the collection of documents
- $\frac{D}{df}$ inverse document frequency - a measure of how much information the word provides, i.e., if it is common or rare across all documents.
- D the number of all documents in the corpus

Document frequency

- We can just take the document frequency of each word and put to the matrix
- The problem - the length of documents varies greatly
- The second problem - not all words characterize the document (stop-words and just common words)

TF-IDF

Normalization: instead of simple word count (DF) we can weighted frequency (TF-IDF)

	Terms						
Docs	вовочк	дет	класс	урок	учител	учительниц	школ
1	0.6897996	0.0000000	0	0.4192463	0	0.4022703	0.0000000
2	0.9197328	0.0000000	0	0.0000000	0	0.5363604	0.0000000
3	0.2759198	0.4781918	0	0.0000000	0	0.3218162	0.4736512
4	0.6897996	0.3984932	0	0.2794975	0	0.2681802	0.0000000
5	1.0346994	0.0000000	0	0.0000000	0	0.4022703	0.0000000
6	0.6897996	0.0000000	0	0.0000000	0	0.8045405	0.0000000

Vocabulary

- we may need to limit the vocabulary size (too many features, memory and time complexity, overfitting)
- some preprocessing may help (for ex, lemmatization)
- cur the vocabulary with some threshold?

Vector space model: summary

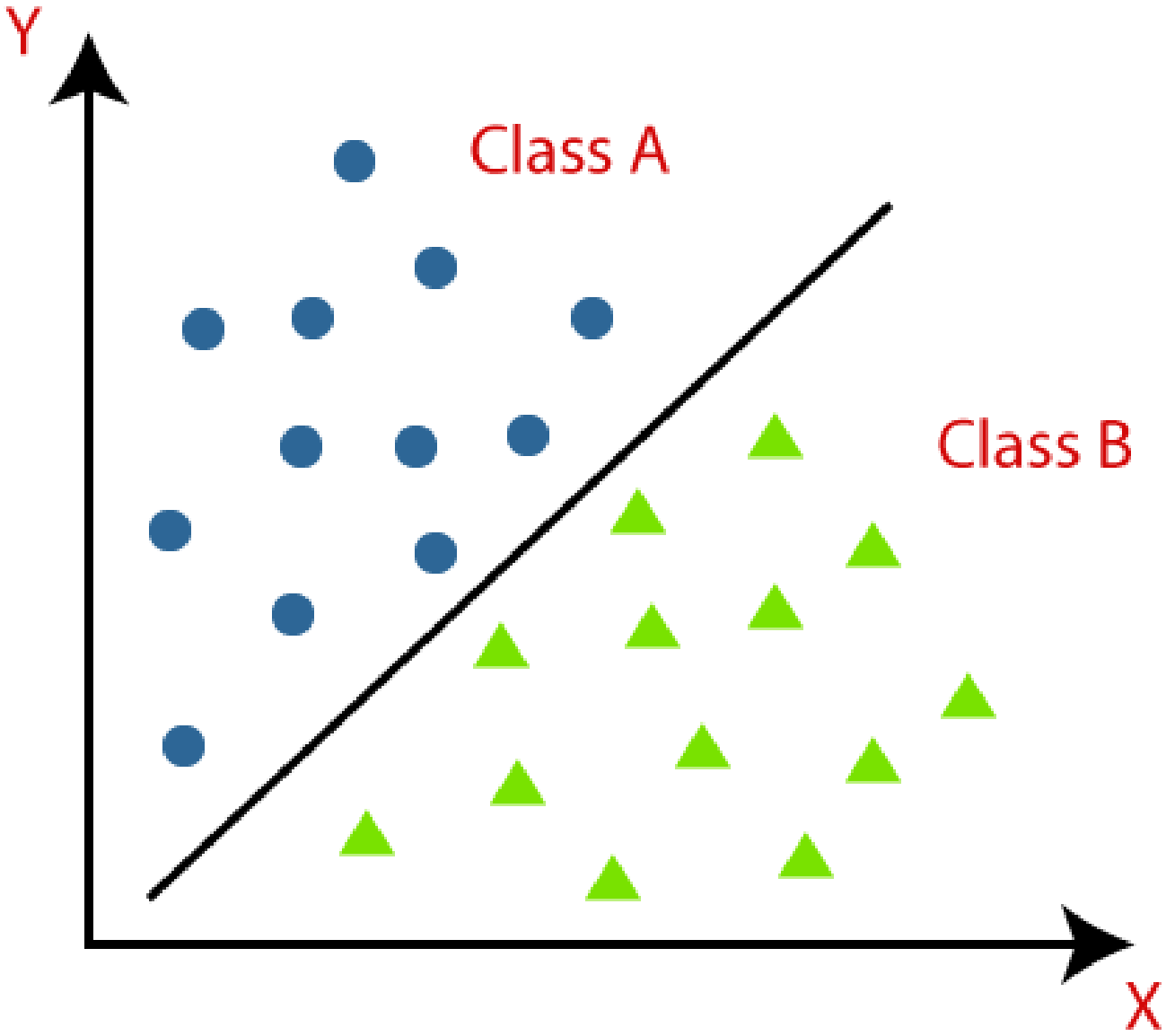
- Text prerpocessing is necessary before building the vectors
- We can represent the document as the document-term matrix
- We need to think about the vocabulary size (number of columns in this matrix)
- Values - just counts of each word in the doc / normalized count
- or TF-IDF which helps to reflect how important a word is to a document in a collection or corpus
- When we have this matrix we can compare documents (with cosine distance), find similar docs
- Or, for ex, we can use this matrix as the data for text classification algorithms

Text Classification

Task:

- We have some predefined list of classes
- It is necessary to automatically assign a class to each document

Vectors classification



Areas of application of the classification in NLP

- For the whole documents:
  - Defining the language of the text
  - Defining the topic of the text (from the list of predefined topics)
  - Sentiment classification (for ex, classifying the review to positive/negative)
  - Defining the author of the text (from the predefined list)
- For the tokens(words):
  - part-of-speech tagging (POS-tagging)
  - Removing homonymy (choosing the meaning of a word)
  - Named entity recognition
  - Relations extraction

Task for ML

Task: learn to predict the properties of an object (text) that are difficult to formalize, but important for a person.

target define the set of the classes (?)

features represent the object as a set of features

model Based on the statistics of the distribution of properties in texts,

- build a model that predicts the labels of new objects (which the model has not yet seen).

Task for ML

Task: learn to predict the properties of an object (text) that are difficult to formalize, but important for a person.

target define the set of the classes (?)

features represent the object as a set of features

model Based on the statistics of the distribution of properties in texts,

- build a model that predicts the labels of new objects (which the model has not yet seen).
- PROFIT!

ML algorithm types

Supervised learning We have features and target classes - model learns to predict these target classes.
Example: classification, regression.

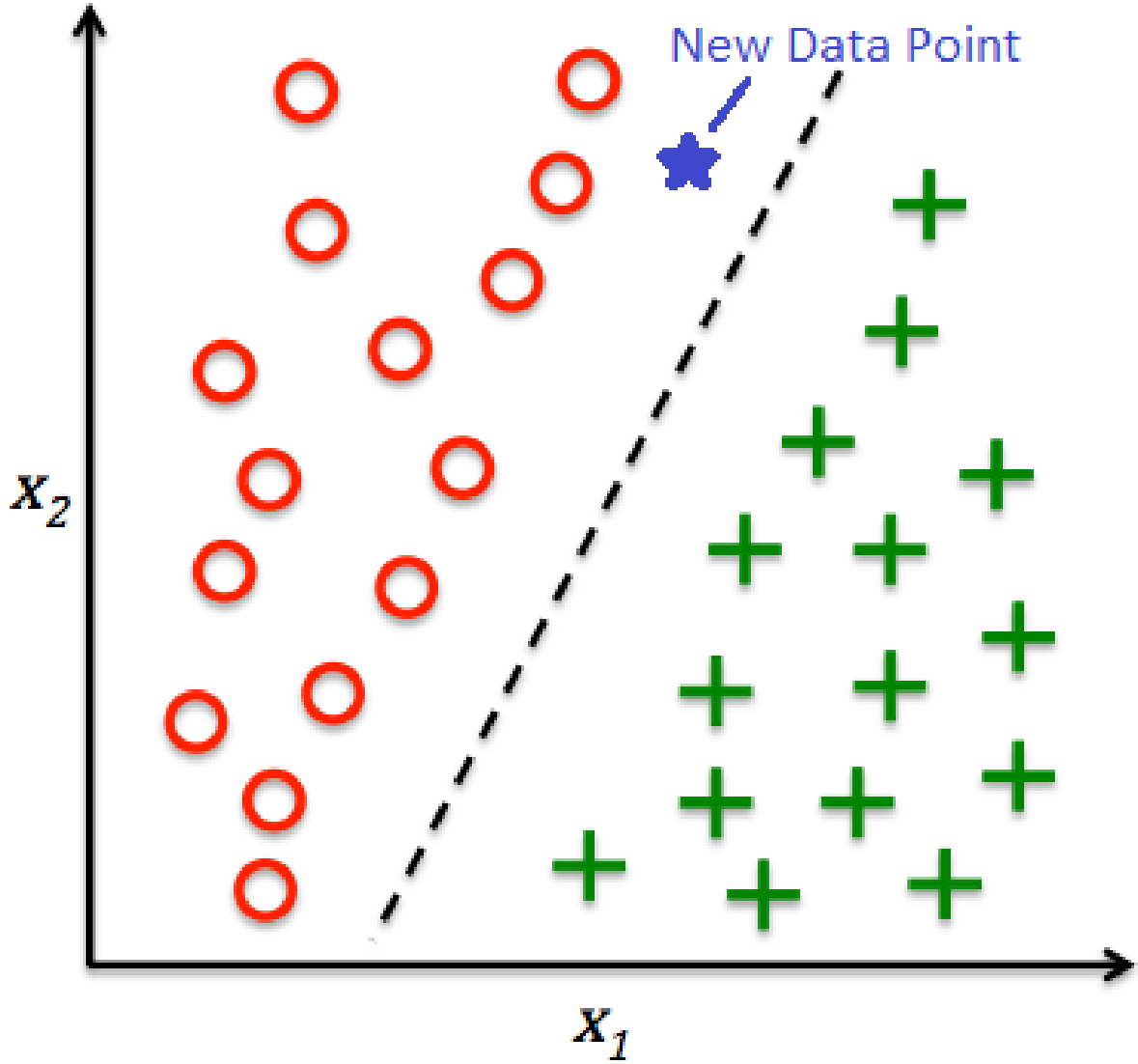
Unsupervised learning No data with target classes, only features. The model learns to predict based on general assumptions about distribution of properties in texts without human preparation labeled samples.
Example: clusterization, topic modelling.

Classification task

Task:

- We have a predefined list of classes
- It is necessary to automatically assign each object to one of classes
- Each object is represented as a set of features.

new objects classification



Train-test splitting

Training Set For training of the model.

Validation Set For unbiased evaluation of the model (for tuning hyperparameters on one algorithm)

Test Set For final evaluation of the model.

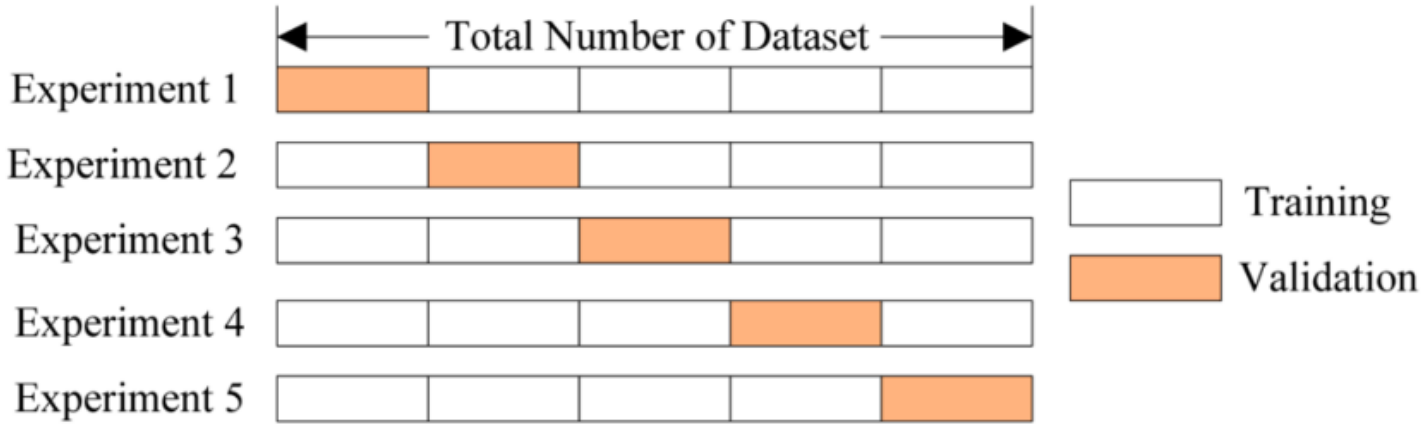
Train-val-test splitting

- Put aside a part of a dataset (10 20%) for validation (validation dataset).
- Train classifier on what is left (training dataset).
- Optimize hyperparameters (such as tree depth in decision tree, for ex), by the metric on validation dataset.
- If we have many hyperparameters, there is a present danger of overfitting to validation metric. Put aside one more dataset (test dataset).
- Test your FINAL model on test set.

Proportions for train/val/test - smth like 0.7/0.15/0.15, 0.8/0.2/0.2

Cross-Validation

Split to train and test. Optimize hyperparameters on train with cross-validation. Test the final model on test set.



Better evaluation (all the data is used in train and validation), but slower (is not efficient when we have the big amount ob observations and heavy algorithms)

Bayes Theorem

$$P(A \text{and} B) =$$

$$P(B)P(A|B) = P(A)P(B|A)$$

Bayes Theorem

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

$$posterior = \frac{likelihood \cdot prior}{evidence} \tag{3}$$

- $P(\text{manager}|\text{shy}) = ?$
- $P(\text{librarian}|\text{shy}) = ?$

- $P(\text{manager}) = \frac{10\text{mln}}{80\text{mln}} = 0.125$
- $P(\text{librarian}) = \frac{0{,}5\text{mln}}{80\text{mln}} = 0.00625$

Bayes theorem for text classification

$$P(\text{label}|\text{features}) = \frac{P(\text{features}|\text{label})P(\text{label})}{P(\text{features})} \tag{4}$$

Naive Bayes classifier

- Task: Having a set of features (E), we need to choose the most probable hypothesis (class, H). Denominator P(E) is a constant and does not affect the results

$$P(\text{label}|\text{features}) \propto P(\text{features}|\text{label})P(\text{label})$$

Maximum a posteriory estimation

Naive assumption

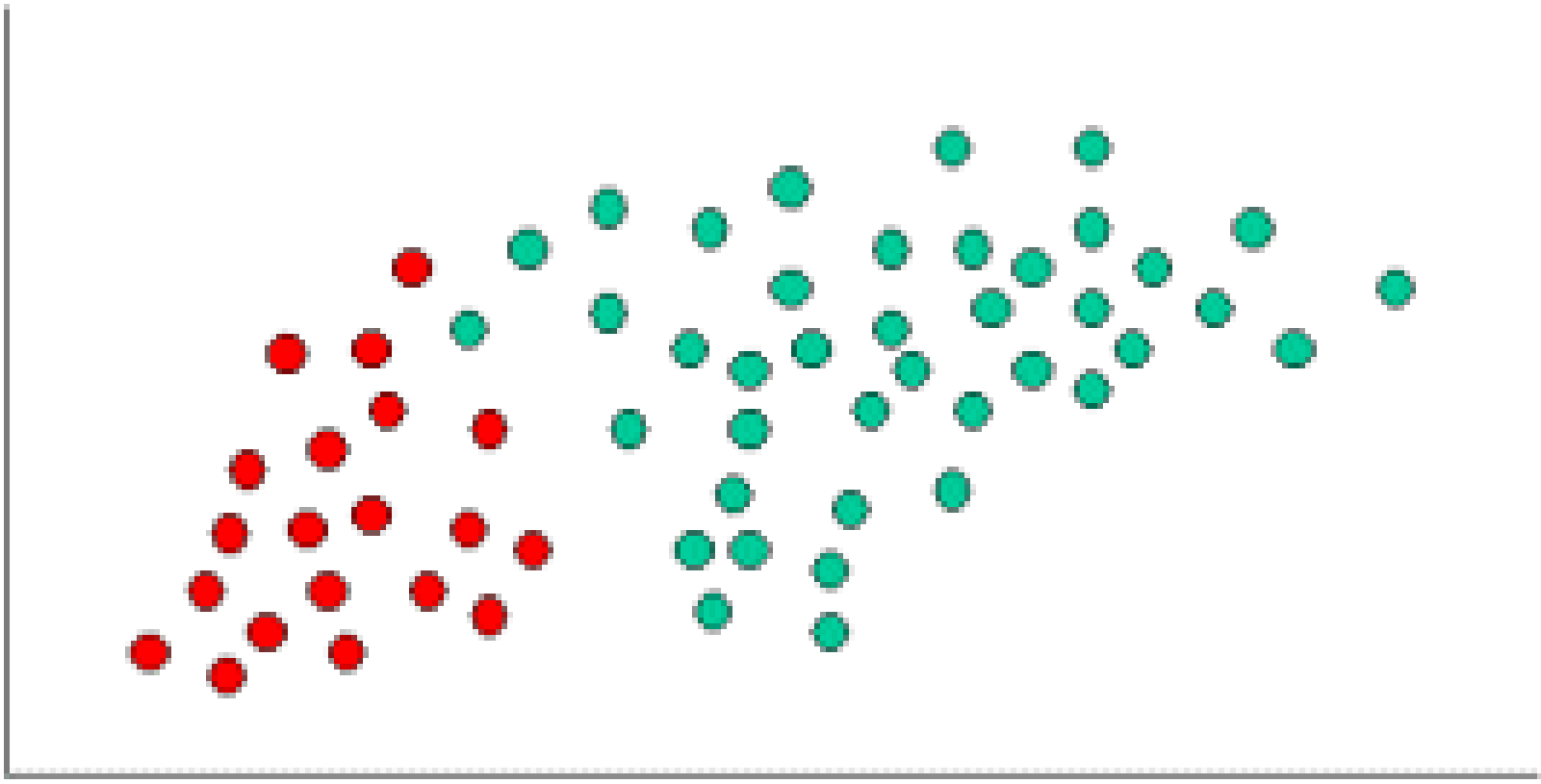
- Calculate likelihood:

$$P(\text{features}|\text{label}) =$$

- Bayes assumption: All features are independent:

$$= \prod_{f \in \text{features}} P(f|\text{label})$$

Naive Bayes example

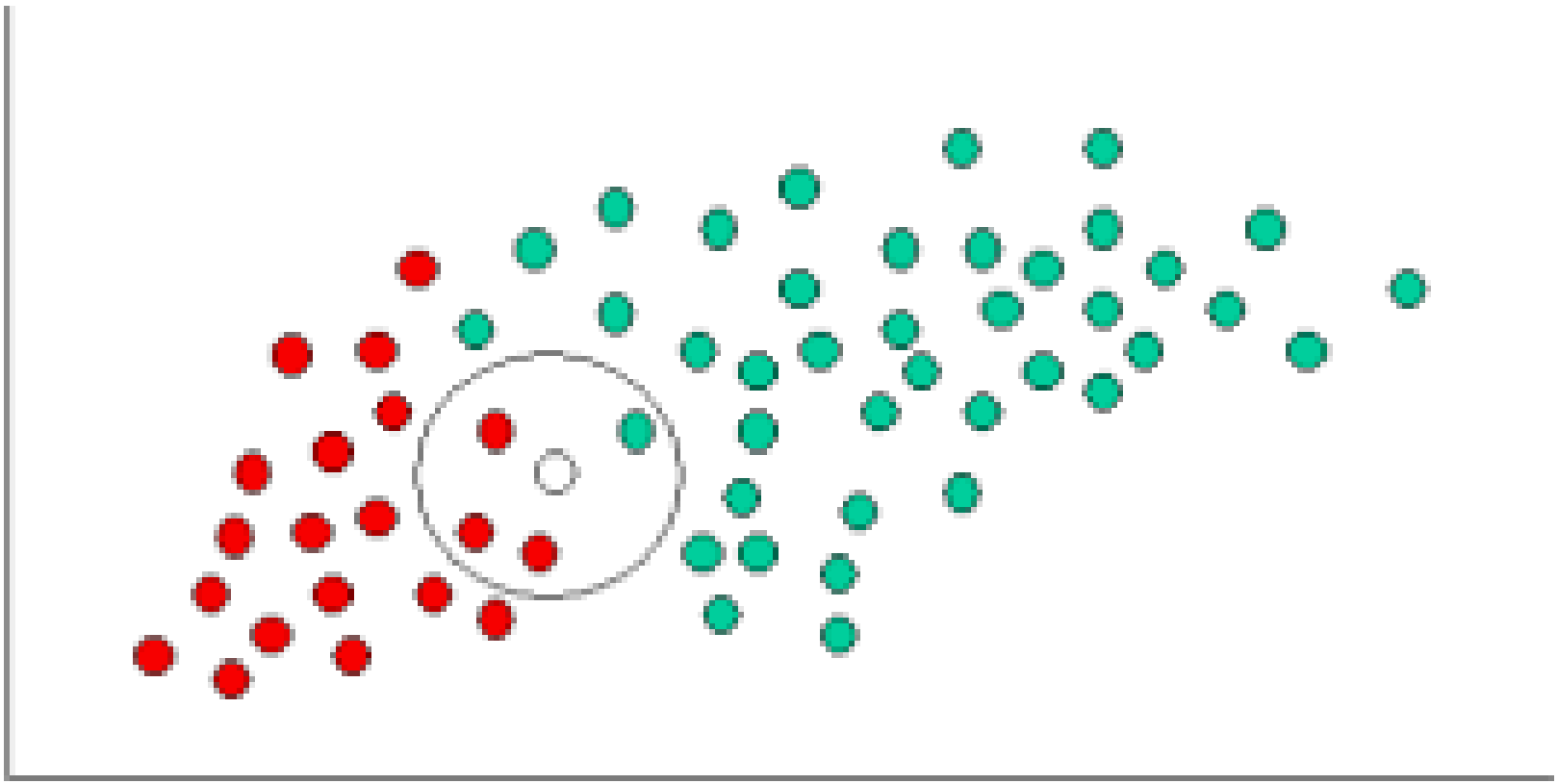


Naive Bayes example

$$\text{Prior}(\text{Green}) \propto \frac{f(\text{Green})}{\text{total}} = \frac{40}{60}$$

$$\text{Prior}(\text{Red}) \propto c\frac{f(\text{Red})}{\text{total}} = \frac{20}{60}$$

Naive Bayes example



Naive Bayes example

- $$\text{Likelihood}(X|\text{Green}) \propto \frac{f(\text{Green near X})}{f(\text{Green})} = \frac{1}{40}$$
- $$\text{Likelihood}(X|\text{Red}) \propto \frac{f(\text{Red near X})}{f(\text{Red})} = \frac{3}{20}$$

Naive Bayes example

$$\text{Posterior}(\text{Green}|\text{X}) \propto \text{Likelihood}(\text{X}|\text{Green}) \times \text{Prior}(\text{Green}) = \frac{4}{6} \times \frac{1}{40} = \frac{1}{60}$$

$$\text{Posterior}(\text{Red}|\text{X}) \propto \text{Likelihood}(\text{X}|\text{Red}) \times \text{Prior}(\text{Red}) = \frac{2}{6} \times \frac{3}{20} = \frac{1}{20}$$

Naive Bayes is generative classifier:

- build the model of each class
- determines the probability that the observed data is generated by the model this class

Naive Bayes

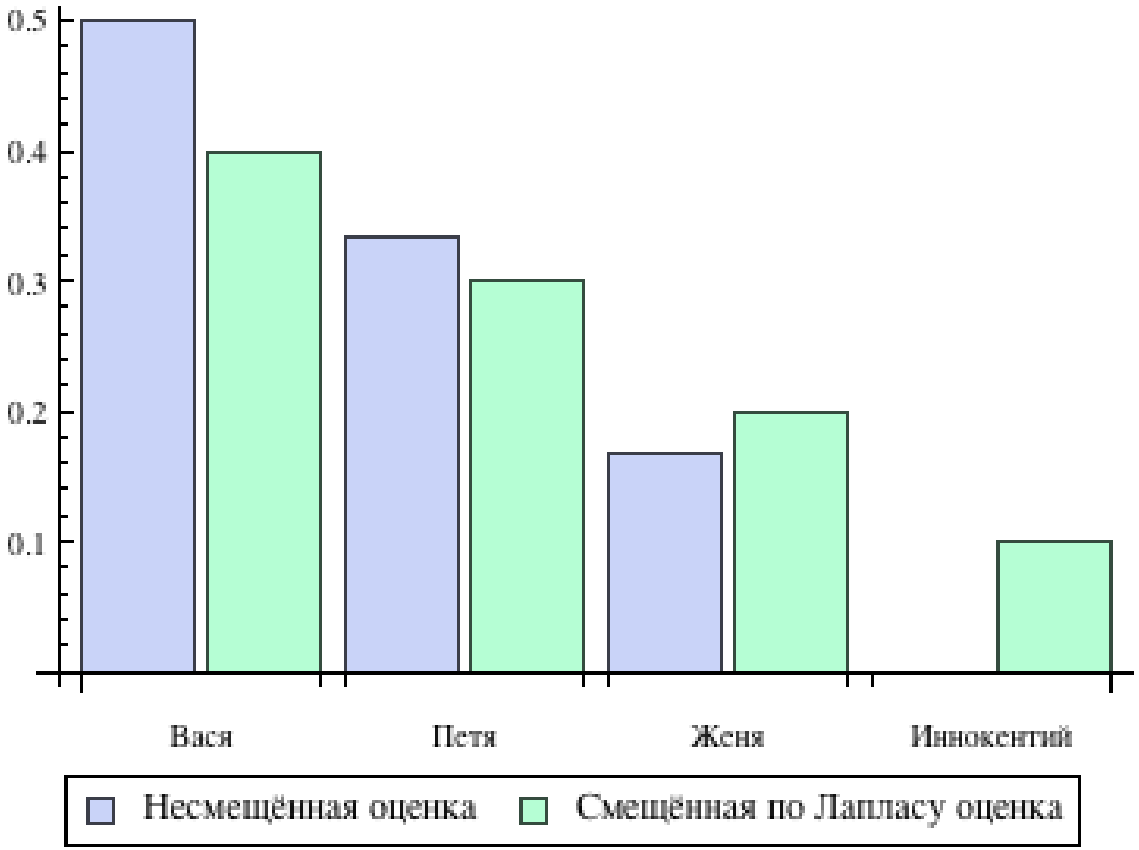
Advantages:

- Suitable for large number of attributes, small training set
- Works surprisingly well for a lot of things (and for texts)
- Computationally efficient (learns and classifies quickly)

Disadvantages:

- The independence assumption is wrong.
- Null values problem (attribute does not appear in train sample) - requires smoothing

Laplas smoothing



Sparse data problem

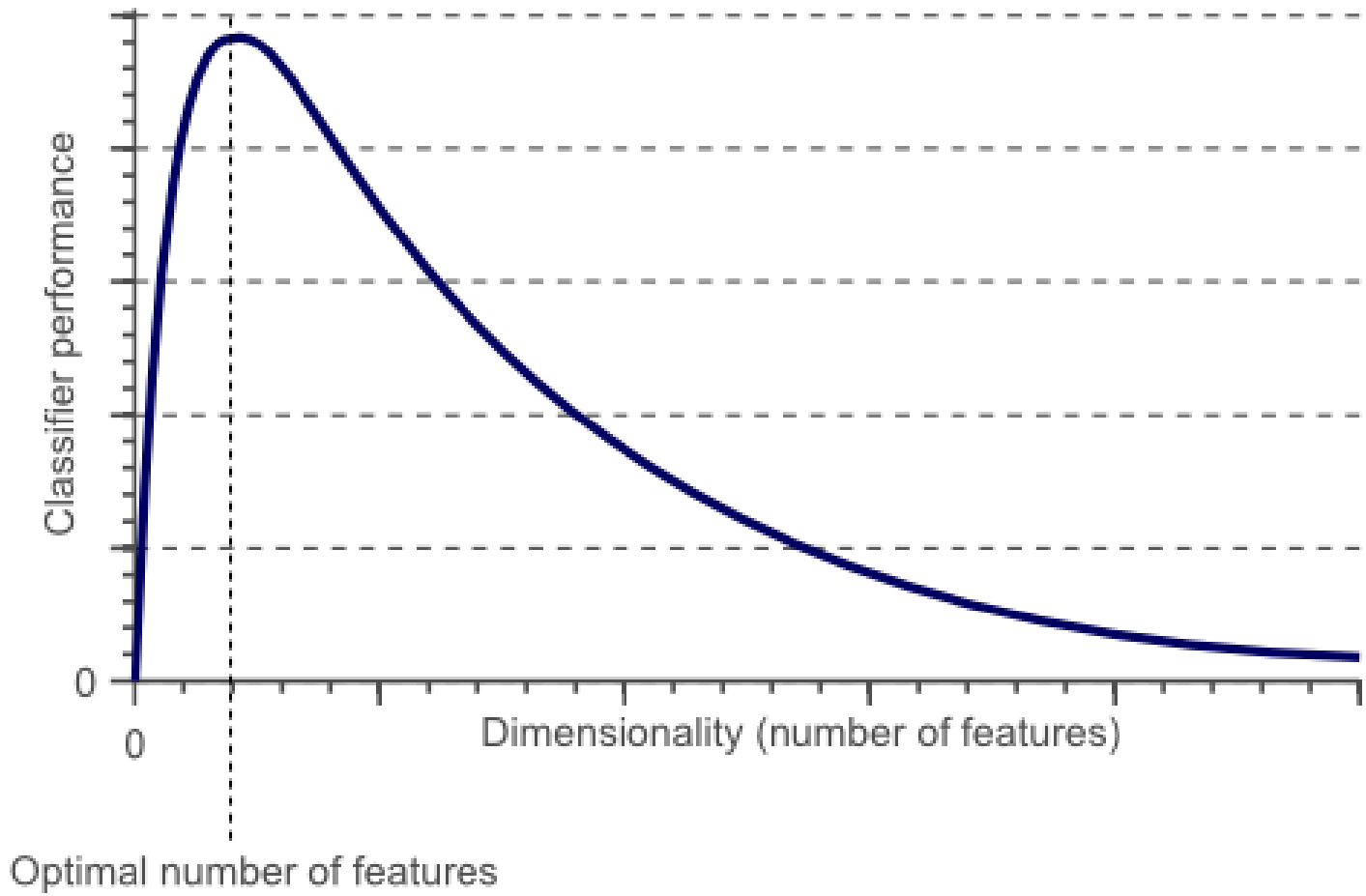
	Terms				
Docs	выгребать	выгребной	выгружать	выгрузка	выгрызать
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

Curse of dimensionality

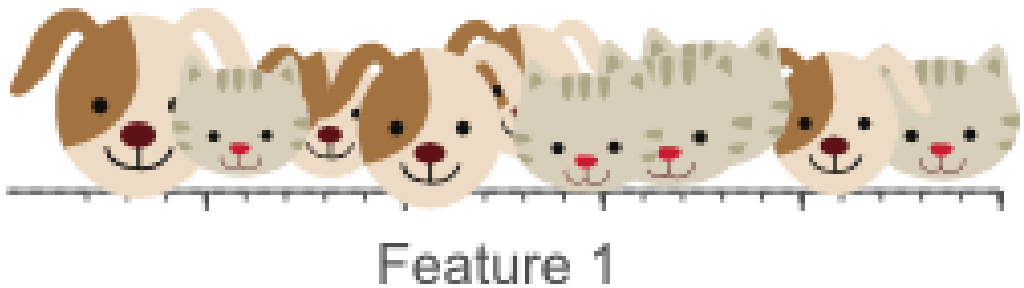
```
A document-term matrix (1530 documents, 13322 terms)

Non-/sparse entries: 68859/20313801
Sparsity           : 100%
Maximal term length: 66
Weighting          : term frequency (tf)
```

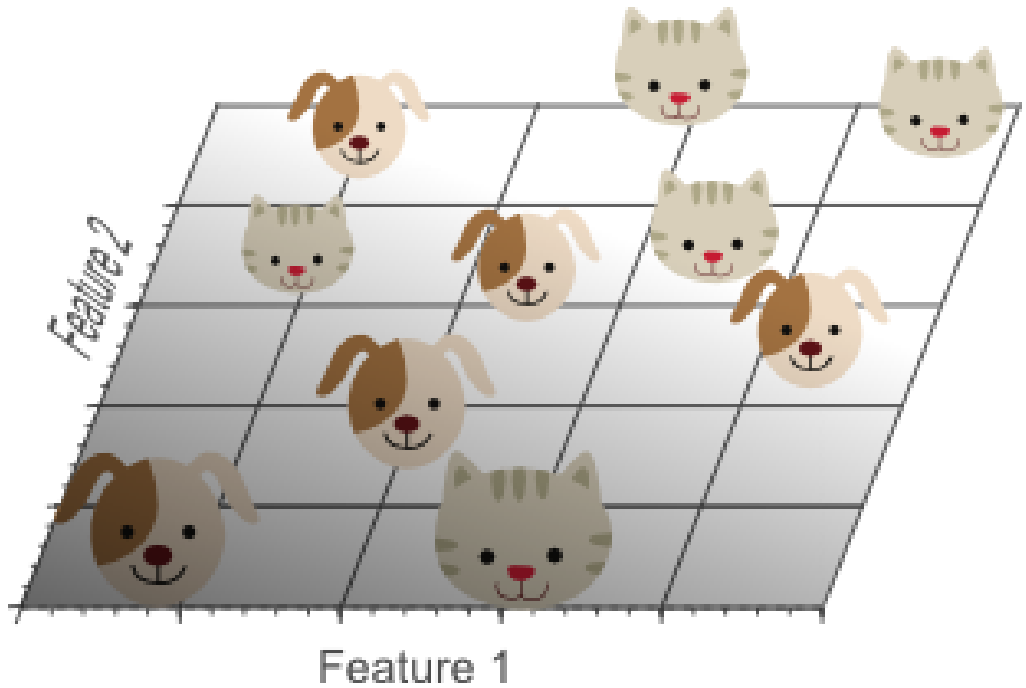
Hughes phenomenon



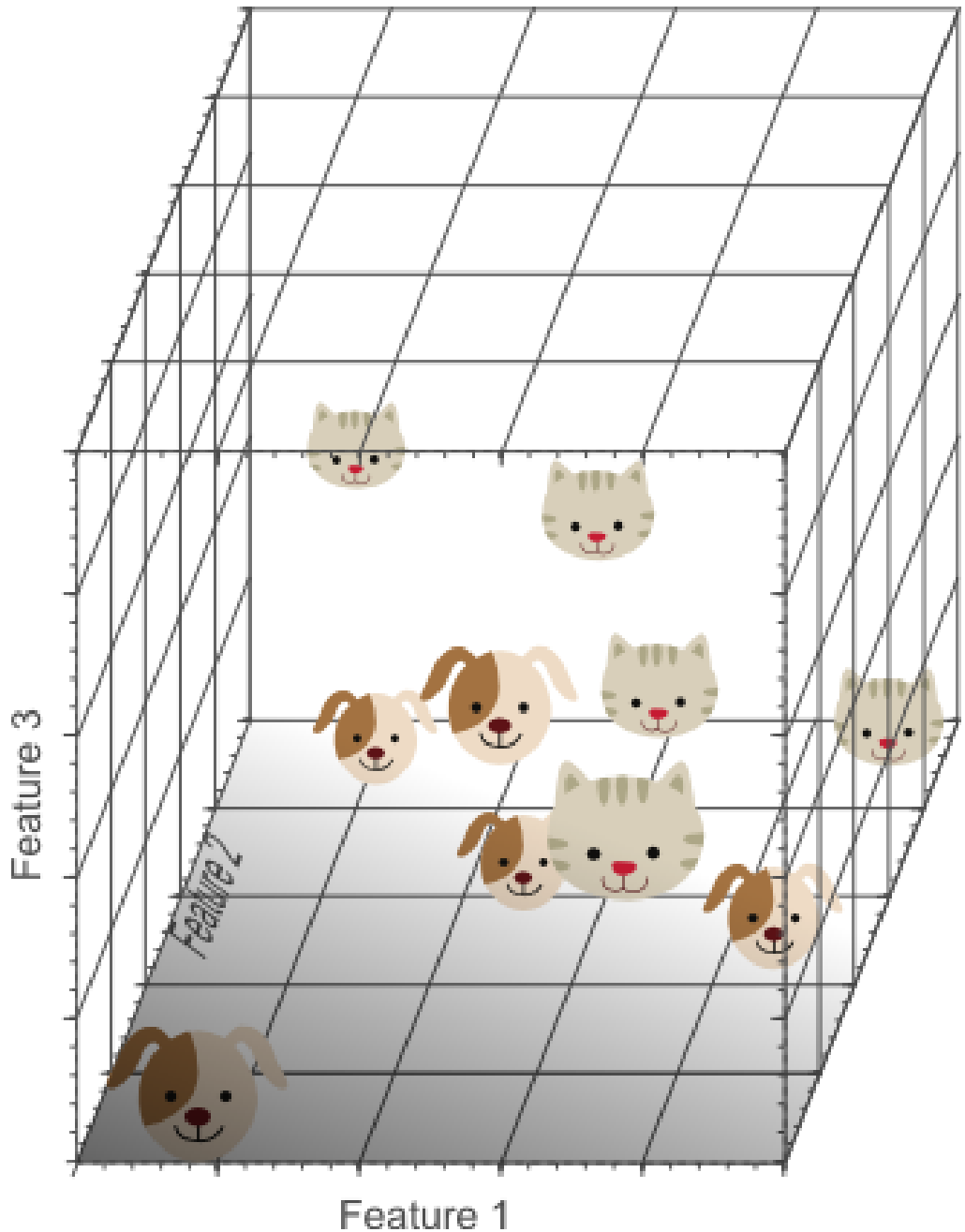
Curse of dimensionality



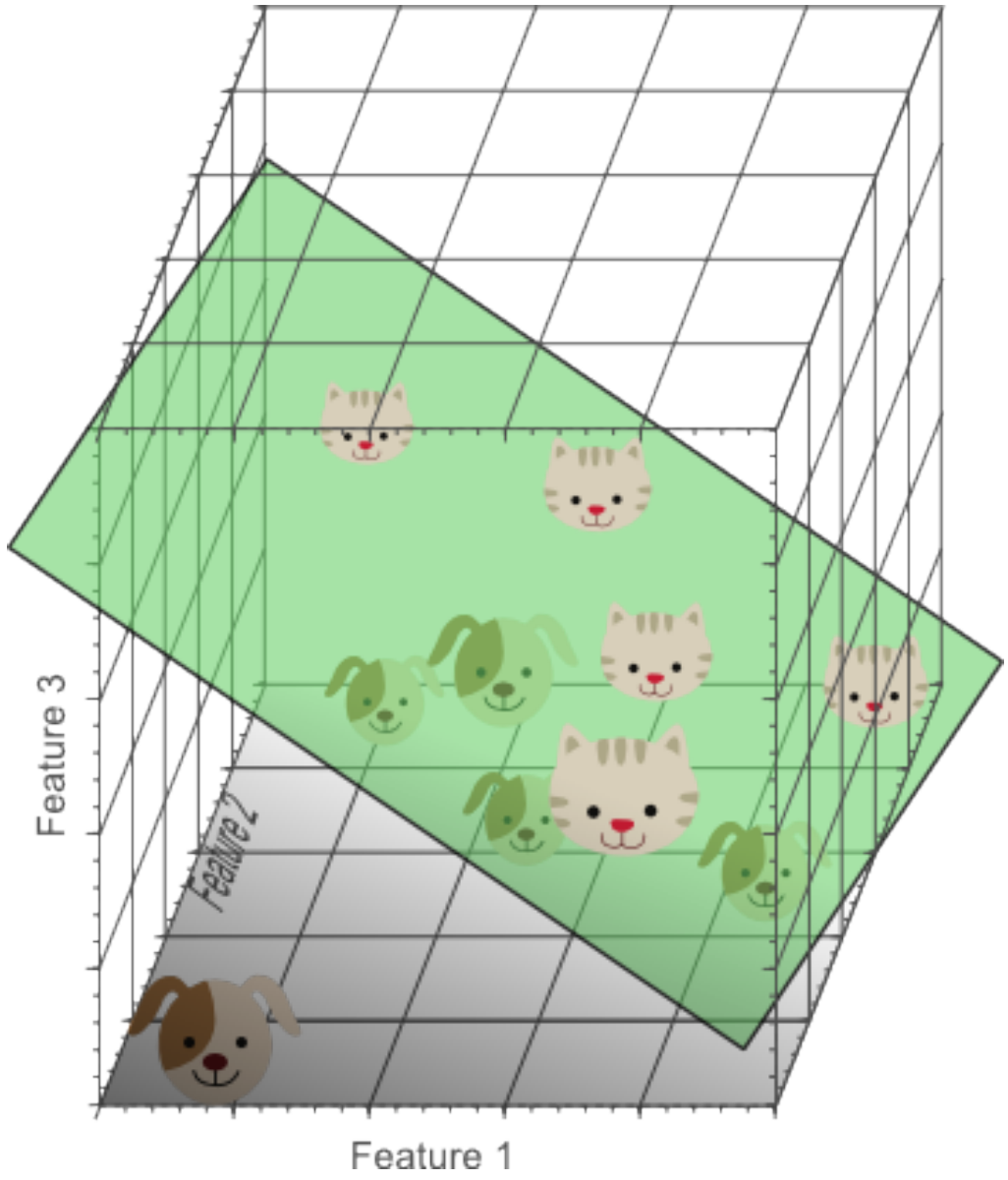
Curse of dimensionality



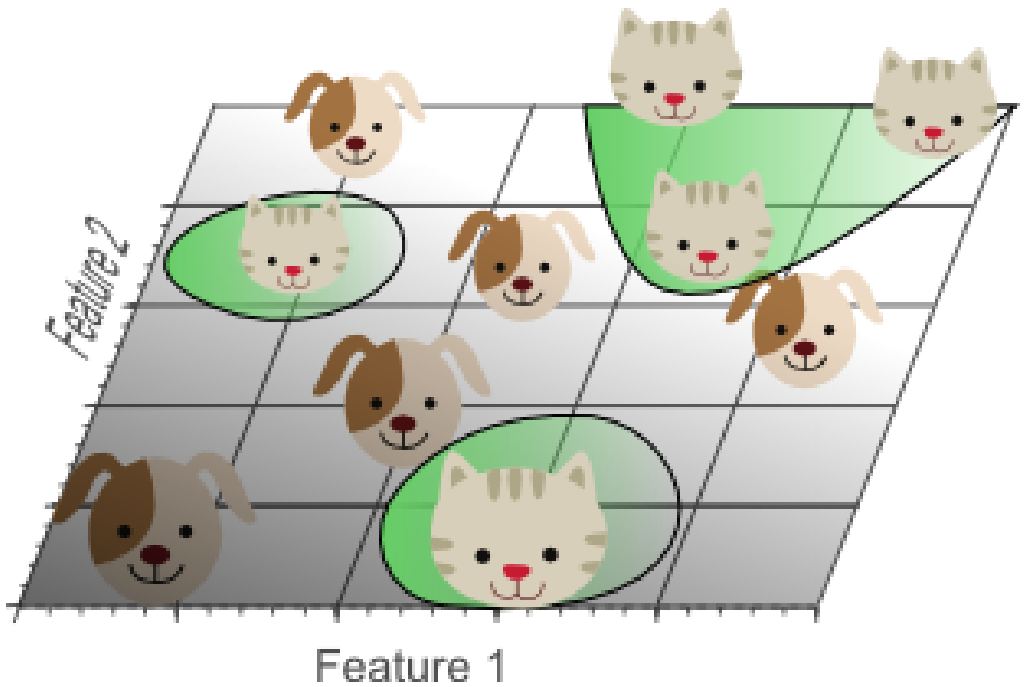
Curse of dimensionality



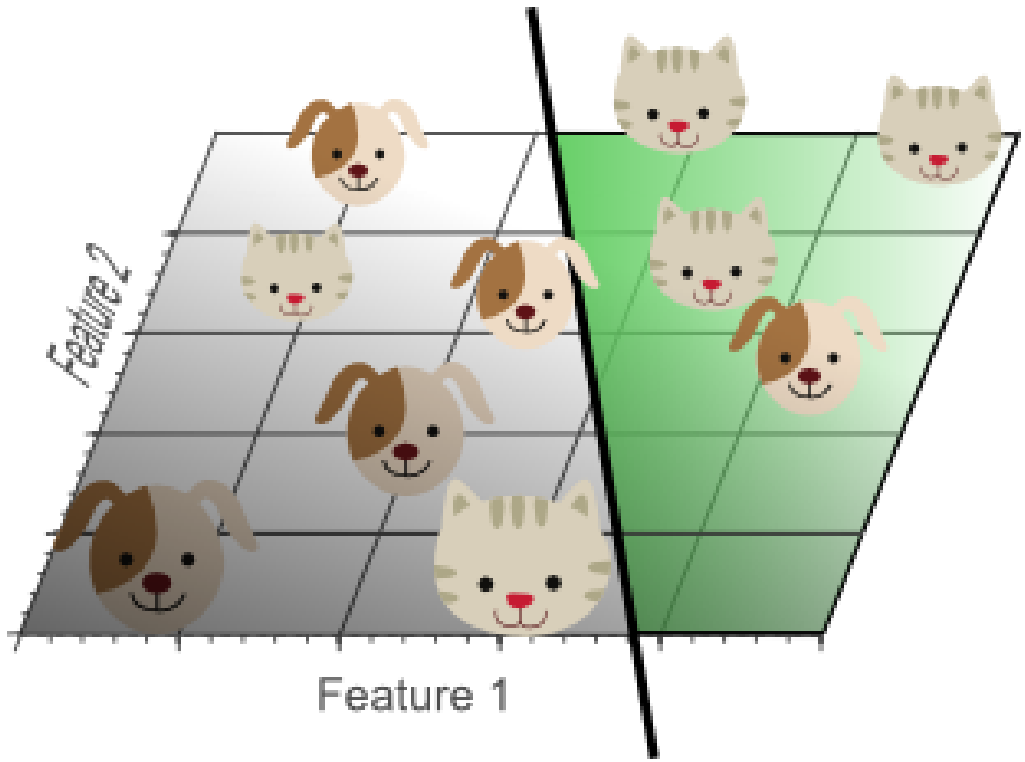
Curse of dimensionality



Overfitting



Overfitting



Dimensionality reduction

In machine learning problems that involve learning a "state-of-nature"from a finite number of data samples in a high-dimensional feature space with each feature having a range of possible values, typically an enormous amount of training data is required to ensure that there are several samples with each combination of values. In an abstract sense, as the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially.
What is more, distances between the points become different from what we except of them. When a measure such as a Euclidean distance is defined using many coordinates, there is little difference in the distances between different pairs of samples.

Dimensionality reduction

- The matrix of document-terms is very large and sparse
- Words that are close in meaning do not necessarily occur in the same the same documents:
  - synonymy
  - polysemy
  - noise

Dimensionality reduction

- The matrix of document-terms is very large and sparse
- Words that are close in meaning do not necessarily occur in the same the same documents:
  - synonymy
  - polysemy
  - noise
- We need to reduce the dimension of the matrix (make fewer columns).

Dimensionality reduction

- drop stop-words
- lemmatization
- Dimensionality reduction algorithms (for ex, PCA)

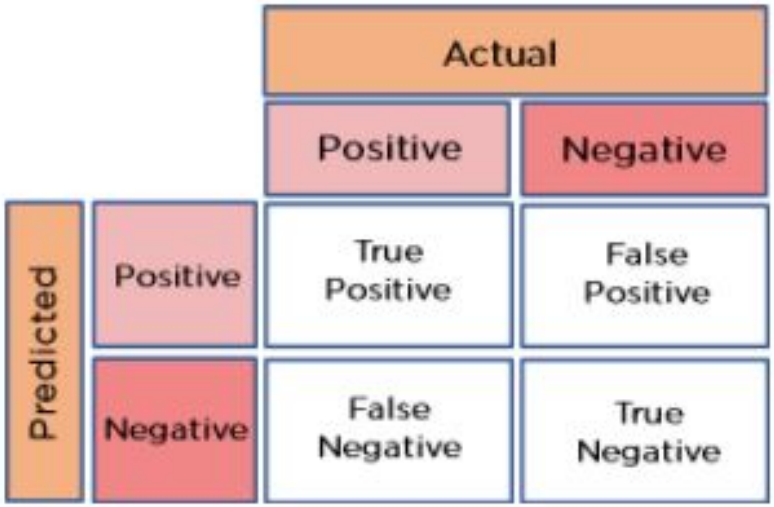
Accuracy

$$Accuracy = \frac{P}{N} \tag{5}$$

- P the number of documents where the classifier accepted the right decision
- N the size of the training set

Contingency table

Table of correct and incorrect decisions on documents of this class:



TP true positive - right prediction of positive class (class 1)

TN true negative - right prediction of negative class (class 0)

FP false positive - wrong prediction of positive class (true class is 0)

FN false negative - wrong prediction of negative class (true class is 1)

Precision and recall

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

Precision is the fraction of relevant instances among the retrieved instances

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

Recall is the fraction of relevant instances that were retrieved.

Precision and recall

- 1 Explain it like fishing with a net. You use a wide net, and catch 80 of 100 total fish in a lake. That's 80% recall. But you also get 80 rocks in your net. That means 50% precision, half of the net's contents is junk.
- 2 You could use a smaller net and target one pocket of the lake where there are lots of fish and no rocks, but you might only get 20 of the fish in order to get 0 rocks. That is 20% recall and 100% precision.

Classification tasks: cancer (0-no cancer, 1-cancer) and spam (0-not spam, 1-spam).
Which metric (precision/recall) is more suitable for each task?

Confusion matrix

	0.91	0.96	0.94	0.75	1.00	0.83	0.85	0.97	1.00	0.86	1.00	0.79	1.00	0.75	1.00	1.00	0.96	0.90	0.81	0.89	0.94	0.98	0.86	0.89	0.94	0.92	0.96
0.80		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
0.95	1	94	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
1.00	2	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.29	3	0	0	6	0	0	3	2	0	1	0	0	0	0	0	0	1	1	0	0	1	0	1	3	0	2	0
1.00	4	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.50	5	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	2	0	1	1
0.92	6	1	0	0	0	0	152	0	0	1	0	0	0	0	0	0	0	1	4	2	3	0	0	0	0	2	0
0.97	7	1	0	1	0	0	0	256	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	2	0
0.33	8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
0.97	9	0	0	0	0	0	0	0	0	69	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
0.82	10	0	0	0	0	0	2	0	0	0	18	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0.87	11	0	0	0	0	0	0	0	0	0	0	34	0	4	0	0	0	0	0	0	0	0	0	1	0	0	0
1.00	12	0	0	0	0	0	0	0	0	0	0	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.57	13	0	0	0	0	0	0	0	0	0	0	9	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0
0.63	14	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	3	0	0	0	0	0	0	0	0	0
0.50	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	1	0	0	0	0	0	0
0.77	16	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	47	0	1	3	4	0	0	2	0	1	0
0.87	17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	69	1	2	5	0	0	0	0	0	0
0.97	18	0	0	0	0	1	4	0	0	1	0	0	0	0	0	0	0	0	197	1	0	0	0	0	0	0	0
0.78	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	35	183	13	0	0	2	0	1	0
0.97	20	0	0	0	0	0	10	3	0	1	0	0	0	0	0	0	0	0	0	4	702	0	0	0	0	6	0
0.93	21	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56	0	2	0	0	0
0.29	22	0	0	1	0	0	2	0	0	6	0	0	0	0	0	0	0	0	1	1	1	0	6	2	0	1	0
0.91	23	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	3	6	0	0	115	0	0	0
1.00	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0
0.93	25	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	4	5	0	0	0	1	196	0
0.98	26	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	78

F1-score

$$F_\alpha = \frac{(1+\alpha)PR}{\alpha P + R} \tag{8}$$

$$F_1 = \frac{2PR}{P+R} \tag{9}$$

Kappa

$$\kappa = \frac{P_{\text{observed}} - P_{\text{expected}}}{1 - P_{\text{expected}}} \quad (10)$$

	cats	dogs	
cats	20	5	25
dogs	10	15	25
	30	20	

Kappa

$$\kappa = \frac{P_{\text{observed}} - P_{\text{expected}}}{1 - P_{\text{expected}}} \quad (10)$$

	cats	dogs	
cats	20	5	25
dogs	10	15	25
	30	20	

$$P_{\text{observed}} = (20 + 15)/50 = 0.7 \quad (11)$$

$$P_{\text{expected}} = ((25 * 30)/50 + (25 * 20)/50))/50 = (15 + 10)/50 = 0.5 \quad (12)$$

$$\kappa = \frac{0.7 - 0.5}{1 - 0.5} = 0.4 \quad (13)$$

Text Classification: summary

- Preparing the texts - lemmatization, tokenization, dropping or not dropping smth (important step, affect the classification results)
- Prepare the documents as the observations with a set of features (document-term matrix)
- Split the data into train-val-test (or just train-test)
- Choose the metric for evaluation that is more suitable for your task
- (*) if data is very unbalanced in terms of classes - Balancing your data (dropping data/data augmentation techniques)
- Choose one/several algorithms, train them on train with different hyperparameters
- Choose the best algorithm/algorithms according to your metric
- Test final model(s) on test set
- (*) interpret results, look at the most important features (words)