Text Classification I

Topic Models and Naive Bayes

Document-term Matrix

Document-Term Matrix

- Bag-of-Words (TF-IDF): Document-Term Matrix

car road gas

truck road

car

gas oil

gas

oil

Toy Corpus:Six Doc.

	car	gas	oil	road	truck
0	1	1	0	1	0
1	0	0	0	1	1
2	1	0	0	0	0
3	0	1	1	0	0
4	0	1	0	0	0
5	0	0	1	0	0

Document-term Matrix

- The shape of C matrix is n by m
- m is vocab. size
- n is number of documents
- **High-dimensionality**, **Sparse**

Just Counting No Semantic

	car	gas	oil	road	truck
0	1	1	0	1	0
1	0	0	0	1	1
2	1	0	0	0	0
3	0	1	1	0	0
4	0	1	0	0	0
5	0	0	1	0	0

Cosine
Similarity is zero

Feature Selection

- Based on measures such as mutual information, keep the top ranked K features.   *choose a subset of the features*

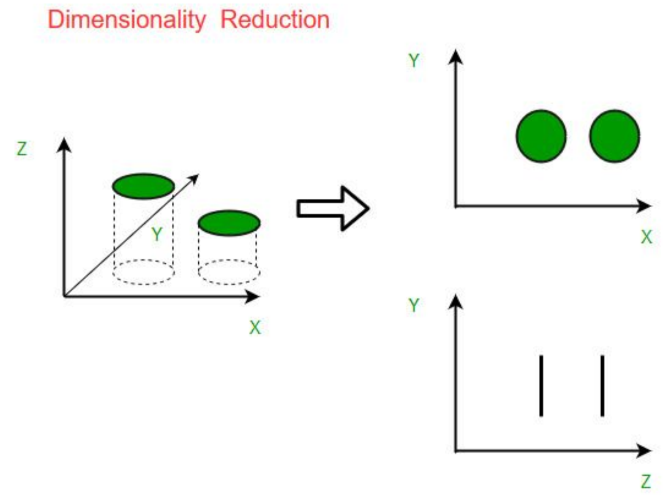
	car	gas	oil	road	truck
0	1	1	0	1	0
1	0	0	0	1	1
2	1	0	0	0	0
3	0	1	1	0	0
4	0	1	0	0	0
5	0	0	1	0	0

	car	gas
0	1	1
1	0	0
2	1	0
3	0	1
4	0	1
5	0	0

Still Sparse and discard lots of information

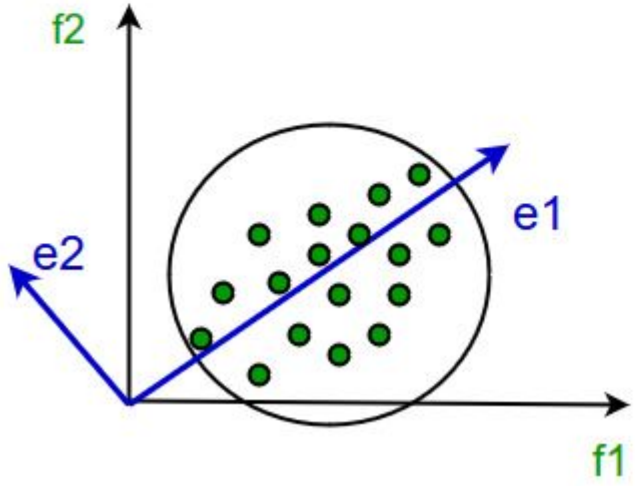
Dimensionality Reduction

- News features will be learned by combining old features
- This is: $\mathbb{R}^M \rightarrow \mathbb{R}^d$ and d < m
- Algorithms:
  - PCA
  - NMF
  - Auto-encoder
  - T-sNE
  - Kernel PCA
  - Manifold learning
  - ….

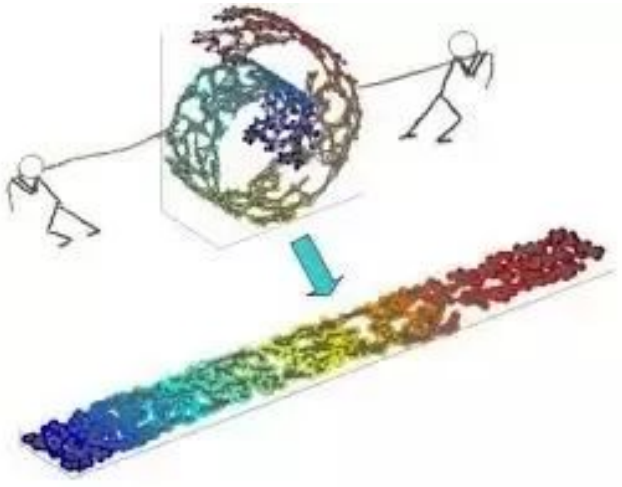


https://www.geeksforgeeks.org/dimensionality-reduction/

Linear vs Nonlinear

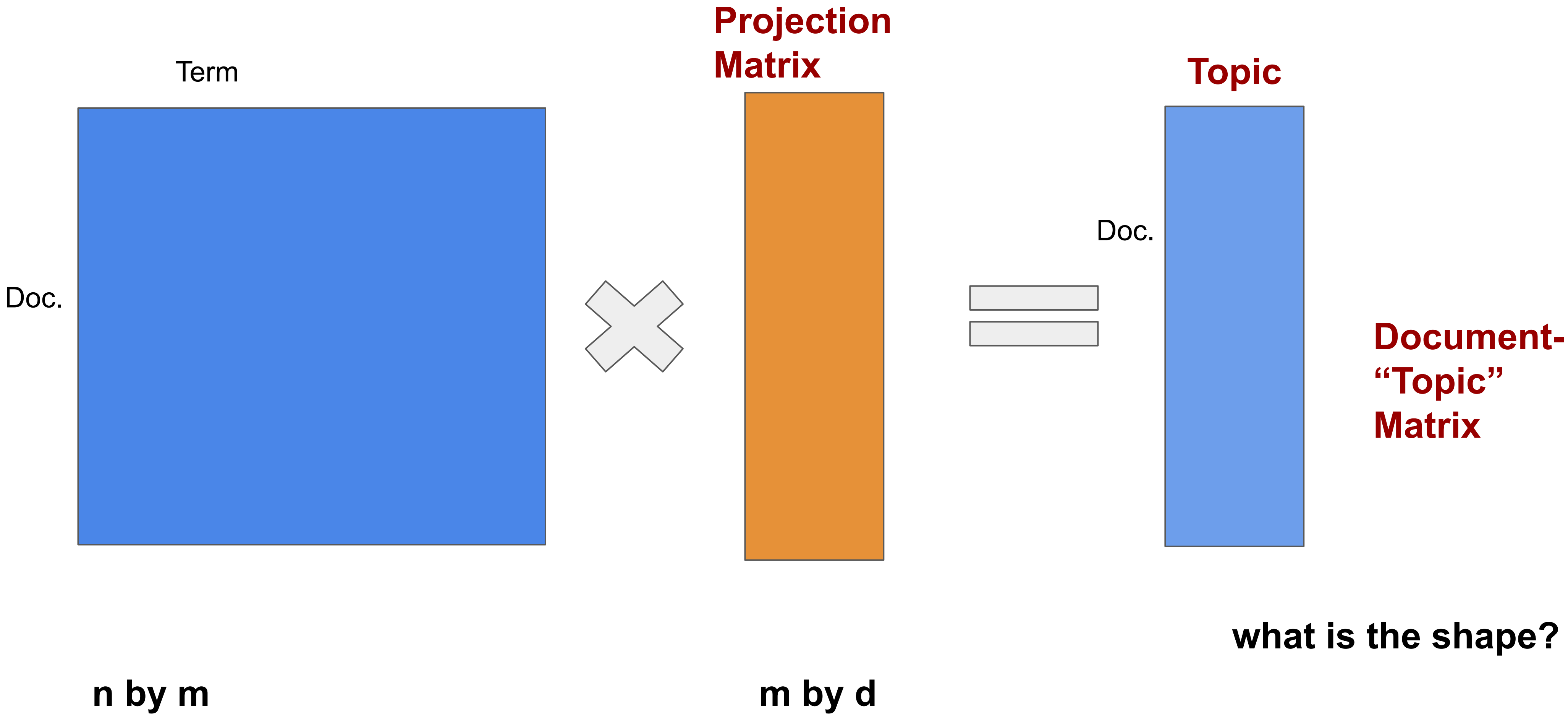


PCA



Manifold Learning (From Quora)

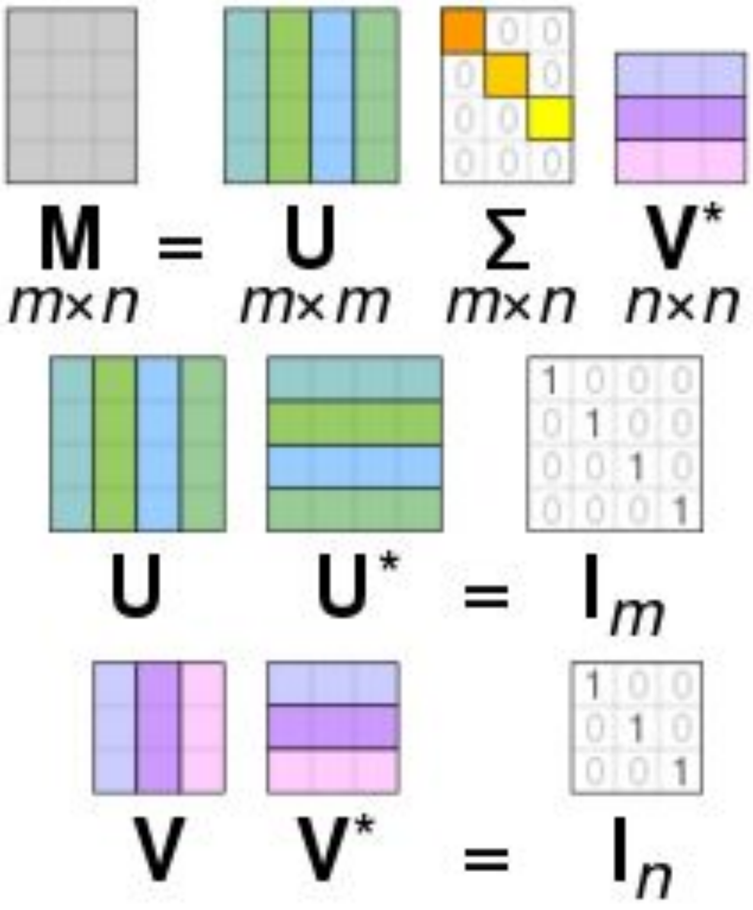
Linear Projection for Term Document Matrix



How to find project matrix

- It is also called matrix decomposition in linear algebra
- **Latent Semantic Analysis: SVD**
- Non-negative Matrix Factorization
- …..

Full SVD

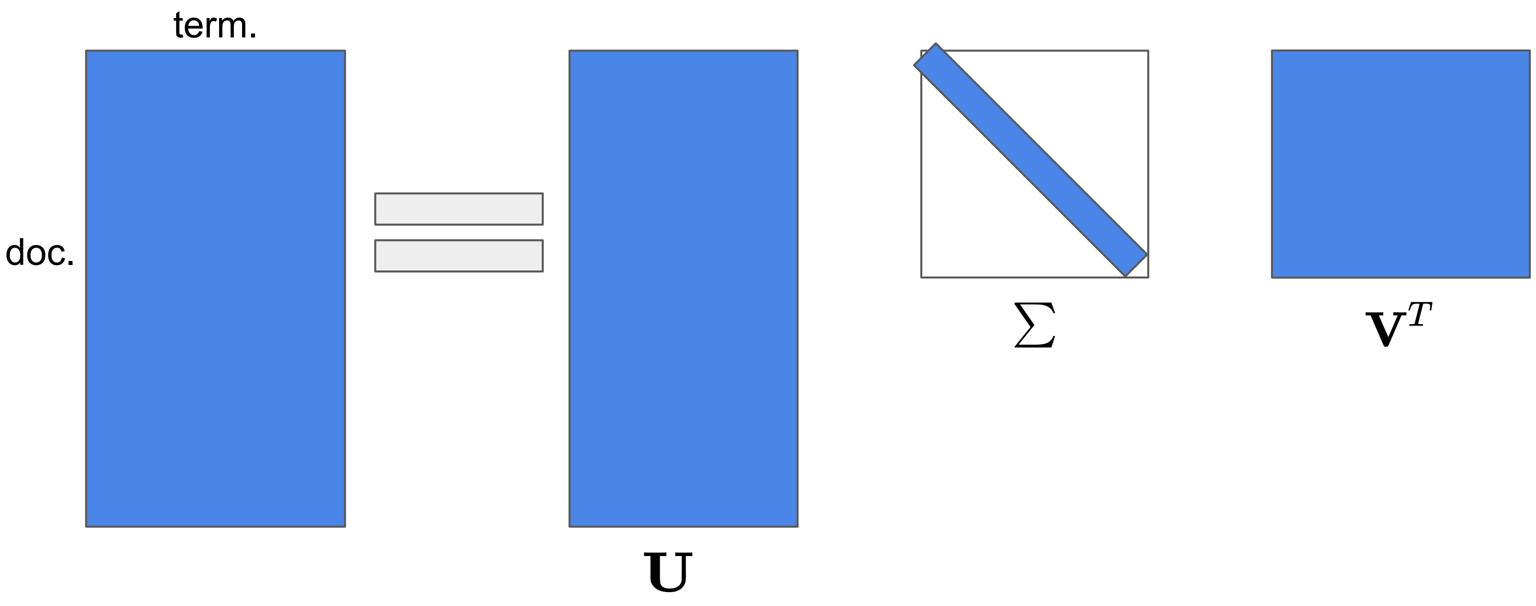


from wiki

How about m < n?

Reduced SVD

- SVD decomposes the matrix (n by m matrix) into 3 parts $\mathbf{C} = \mathbf{U} * \sum * \mathbf{V}^T$
  - $\mathbf{U}$ is a matrix of size n by r
  - $\sum$ is a diagonal matrix whose diagonal entries are known as the singular values and the size is r by r
  - $\mathbf{V}^T$ is a matrix of size of r by m
  - r is the rank of the matrix, which is usually the min value of m and n



$$\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}^{r \times r}$$

https://csiu.github.io/blog/update/2017/04/16/day51.html

Look at the previous example

	car	gas	oil	road	truck
0	1	1	0	1	0
1	0	0	0	1	1
2	1	0	0	0	0
3	0	1	1	0	0
4	0	1	0	0	0
5	0	0	1	0	0

=

	topic1	topic2	topic3	topic4	topic5
0	-0.748623	0.286454	-0.279712	-1.703299e-17	0.528459
1	-0.279712	0.528459	0.748623	-6.485281e-16	-0.286454
2	-0.203629	0.185761	-0.446563	5.773503e-01	-0.625521
3	-0.446563	-0.625521	0.203629	5.088081e-16	-0.185761
4	-0.325096	-0.219880	-0.121467	-5.773503e-01	-0.405641
5	-0.121467	-0.405641	0.325096	5.773503e-01	0.219880

representation of documents

×

	topic1	topic2	topic3	topic4	topic5
topic1	2.162501	0.000000	0.00000	0.0	0.000000
topic2	0.000000	1.594382	0.00000	0.0	0.000000
topic3	0.000000	0.000000	1.27529	0.0	0.000000
topic4	0.000000	0.000000	0.00000	1.0	0.000000
topic5	0.000000	0.000000	0.00000	0.0	0.393915

importance of the semantic dimensions

×

	car	gas	oil	road	truck
topic1	-0.440347	-0.703020	-0.262673	-4.755303e-01	-1.293463e-01
topic2	0.296174	-0.350572	-0.646747	5.111152e-01	3.314507e-01
topic3	-0.569498	-0.154906	0.414592	3.676900e-01	5.870217e-01
topic4	0.577350	-0.577350	0.577350	-2.493650e-16	-6.963379e-16
topic5	-0.246402	-0.159788	0.086614	6.143584e-01	-7.271970e-01

representation of terms

New Representation for Documents

	topic1	topic2	topic3	topic4	topic5
0	-0.748623	0.286454	-0.279712	-1.703299e-17	0.528459
1	-0.279712	0.528459	0.748623	-6.485281e-16	-0.286454
2	-0.203629	0.185761	-0.446563	5.773503e-01	-0.625521
3	-0.446563	-0.625521	0.203629	5.088081e-16	-0.185761
4	-0.325096	-0.219880	-0.121467	-5.773503e-01	-0.405641
5	-0.121467	-0.405641	0.325096	5.773503e-01	0.219880

representation of documents

	topic1	topic2	topic3	topic4	topic5
topic1	2.162501	0.000000	0.00000	0.0	0.000000
topic2	0.000000	1.594382	0.00000	0.0	0.000000
topic3	0.000000	0.000000	1.27529	0.0	0.000000
topic4	0.000000	0.000000	0.00000	1.0	0.000000
topic5	0.000000	0.000000	0.00000	0.0	0.393915

importance of the semantic dimensions

	topic1	topic2	topic3	topic4	topic5
0	-1.618898	0.456717	-0.356713	-1.703299e-17	0.208168
1	-0.604877	0.842566	0.954712	-6.485281e-16	-0.112839
2	-0.440347	0.296174	-0.569498	5.773503e-01	-0.246402
3	-0.965693	-0.997319	0.259686	5.088081e-16	-0.073174
4	-0.703020	-0.350572	-0.154906	-5.773503e-01	-0.159788
5	-0.262673	-0.646747	0.414592	5.773503e-01	0.086614

final representation of documents $\mathbf{U} \Sigma$

Vectors for documents are dense in the new learned topic space. However, the similarity between doc4 and doc5 are still tiny

```
]: print(cosine_similarity(dense_matrix[4].reshape(1, -1), dense_matrix[5].reshape(1

[[-6.10622664e-16]]
```

Projection Matrix is V

	car	gas	oil	road	truck
topic1	-0.440347	-0.703020	-0.262673	-4.755303e-01	-1.293463e-01
topic2	0.296174	-0.350572	-0.646747	5.111152e-01	3.314507e-01
topic3	-0.569498	-0.154906	0.414592	3.676900e-01	5.870217e-01
topic4	0.577350	-0.577350	0.577350	-2.493650e-16	-6.963379e-16
topic5	-0.246402	-0.159788	0.086614	6.143584e-01	-7.271970e-01

$$\mathbf{V}^T$$

	topic1	topic2	topic3	topic4	topic5
car	-0.440347	0.296174	-0.569498	5.773503e-01	-0.246402
gas	-0.703020	-0.350572	-0.154906	-5.773503e-01	-0.159788
oil	-0.262673	-0.646747	0.414592	5.773503e-01	0.086614
road	-0.475530	0.511115	0.367690	-2.493650e-16	0.614358
truck	-0.129346	0.331451	0.587022	-6.963379e-16	-0.727197

$$\mathbf{V}$$

$$\mathbf{C} = \mathbf{U} * \sum * \mathbf{V}^T \quad \Rightarrow \quad \mathbf{C} * \mathbf{V} = \mathbf{U} * \sum$$

projection matrix

Topic 1 = -0.44 * **car** - 0.70 * **gas** - 0.26 * **oil** - 0.47 * **road** - 0.13 * **truck**

Each topic is regarded as the **linear** combination of words

For a new document

car	gas	oil	road	truck
0	0	1	1	1

×

	topic1	topic2	topic3	topic4	topic5
car	-0.440347	0.296174	-0.569498	5.773503e-01	-0.246402
gas	-0.703020	-0.350572	-0.154906	-5.773503e-01	-0.159788
oil	-0.262673	-0.646747	0.414592	5.773503e-01	0.086614
road	-0.475530	0.511115	0.367690	-2.493650e-16	0.614358
truck	-0.129346	0.331451	0.587022	-6.963379e-16	-0.727197

=

topic1	topic2	topic3	topic4	topic5
-0.867549	0.195819	1.369303	0.57735	-0.026225

V

Sparse → Dense

How to reduce dimensionality

Abandon unimportant topics

Reduce Dimensionality

- Each singular value in Σ tells us how important its dimension is.

- **By setting less important dimensions to zero, we keep the important information, but get rid of the details**

- The details may
    - be noise - in that case, reduced LSI is better representation because it is less noisy
    - make things dissimilar that should be similar - again, the reduced LSI representation is a better representation because it represents similarity better.

Truncated SVD

- Zeroing out but these two largest singular values

	topic1	topic2	topic3	topic4	topic5
topic1	2.162501	0.000000	0.00000	0.0	0.000000
topic2	0.000000	1.594382	0.00000	0.0	0.000000
topic3	0.000000	0.000000	1.27529	0.0	0.000000
topic4	0.000000	0.000000	0.00000	1.0	0.000000
topic5	0.000000	0.000000	0.00000	0.0	0.393915

	topic1	topic2	topic3	topic4	topic5
topic1	2.162501	0.000000	0.0	0.0	0.0
topic2	0.000000	1.594382	0.0	0.0	0.0
topic3	0.000000	0.000000	0.0	0.0	0.0
topic4	0.000000	0.000000	0.0	0.0	0.0
topic5	0.000000	0.000000	0.0	0.0	0.0

New Representation for Documents in 2 dimensions

	topic1	topic2	topic3	topic4	topic5
0	-0.748623	0.286454	-0.279712	-1.703299e-17	0.528459
1	-0.279712	0.528459	0.748623	[illegible]	-0.286454
2	-0.203629	0.185761	-0.446563	5.773503e-01	-0.625521
3	-0.446563	-0.625521	0.203629	5.088081e-16	-0.185761
4	-0.325096	-0.219880	-0.121467	-5.773503e-01	-0.405641
5	-0.121467	-0.405641	0.325096	5.773503e-01	0.219880

representation of documents

$$\mathbf{U}_d$$

	topic1	topic2	topic3	topic4	topic5
topic1	2.162501	0.000000	0.0	0.0	0.0
topic2	0.000000	1.594382	0.0	0.0	0.0
topic3	0.000000	0.000000	0.0	0.0	0.0
topic4	0.000000	0.000000	0.0	0.0	0.0
topic5	0.000000	0.000000	0.0	0.0	0.0

importance of the semantic dimensions

$$\Sigma_d$$

	topic1	topic2	topic3	topic4	topic5
0	-1.618898	0.456717	-0.0	-0.0	0.0
1	-0.604877	0.842566	0.0	0.0	-0.0
2	-0.440347	0.296174	-0.0	0.0	-0.0
3	-0.965693	-0.997319	0.0	0.0	-0.0
4	-0.703020	-0.350572	-0.0	-0.0	-0.0
5	-0.262673	-0.646747	0.0	0.0	0.0

The new feature space is 2

Now, we can compute the similarity for doc 4 and doc 5

```
# compute the new similarity
print(cosine_similarity(lowdim_dense_matrix[4].reshape(1, -1), lowdim_dens
```

```
[[0.75020516]]
```

For a new document

car	gas	oil	road	truck
0	0	1	1	1

	topic1	topic2	topic3	topic4	topic5
car	-0.440347	0.296174	-0.569498	5.773503e-01	-0.246402
gas	-0.703020	-0.350572	-0.154906	-5.773503e-01	-0.159788
oil	-0.262673	-0.646747	0.414592	5.773503e-01	0.086614
road	-0.475530	0.511115	0.367690	-2.493650e-16	0.614358
truck	-0.129346	0.331451	0.587022	-6.963379e-16	-0.727197

	topic1	topic2
0	-0.867549	0.195819

$$V_d = V[:,0:d]$$

Here, d is less than the original matrix rank.

Why we use LSA as text vectors

- LSA try to capture semantic information (talk about the same topic)
  - do not need documents use same words
  - project doc in a reduced vector space
- Try to addresses the linguistic characteristic: **synonymy** and **semantic relatedness**.

How LSA addresses synonymy and semantic relatedness

- The dimensionality reduction forces us to ignore a lot of details.
- We try to map different words (original vector space) to the same dimension in the reduced space.
- The “cost” of mapping synonyms to the same dimensions is much less than the cost of collapsing unrelated words.
- SVD selects the “least costly” mapping. (Eckart-Young theorem)

	topic1	topic2	topic3	topic4	topic5
car	-0.440347	0.296174	-0.569498	5.773503e-01	-0.246402
gas	-0.703020	-0.350572	-0.154906	-5.773503e-01	-0.159788
oil	-0.262673	-0.646747	0.414592	5.773503e-01	0.086614
road	-0.475530	0.511115	0.367690	-2.493650e-16	0.614358
truck	-0.129346	0.331451	0.587022	-6.963379e-16	-0.727197

Implementation

- Given a corpus, get the document-term matrix
- Compute SVD of the matrix $\mathbf{C} = \mathbf{U} * \sum * \mathbf{V}^T$
- The original corpus are represented in the reduced space (dim is d):

$$\mathbf{U}_d \sum_d$$

- The new documents can be firstly transformed in the original vector space $\mathbf{q}$
- Map them into the reduced space $\mathbf{q}\mathbf{V}_d$

Other approaches for document representation

- Other Matrix Decomposition methods:
  - NMF
- Probabilistic Model:
  - Topic Model: Latent Dirichlet Allocation
- Deep learning based Model:
  - RNN, CNN

Naive Bayes

Pre Study

Josh Wills
@josh_wills

Follow

Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.

9:55 AM - 3 May 2012

1,626 Retweets **1,294** Likes

52 1.6K 1.3K

Basics of Probability

$P(A)$

- $P(A|B)$: probability that A happens
- : probability that A happens, given that B happens (conditional probability)
- Some rules:
    - Complement: $P(A^C) = 1 - P(A)$
    - Disjunction: $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
    - Conjunction: $P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$
    - If A and B are independent, $P(A \cap B) = P(A)P(B)$
    - Total probability: $P(B) = \sum_{i=1}^{k} P(A|B_i)P(B_i)$

Conditional Probability

- If: the patient has symptom of toothache
- Then: conclude cavity with probability P
- where P is the following conditional probability

$$P(\text{cavity}|\text{toothache})$$

- To compute p(cavity|toothache), we can compute

$$p(\text{cavity} \land \text{toothache}) / p(\text{cavity})$$

Density Estimation

- Density Estimation task
    - To construct an estimate of an unobservable underling probability density function, based on some observed data
- Data
    - Data sample x drawn i.i.d (independent identically distributed) from set **X** according to some distribution d, $x_i, \ldots, x_m \in \mathbf{X}$
- Problem
    - To find a distribution p out of a set P that best estimates the true distribution d

Argmax/Argmin

argmax stands for the argument of the maximum, that is to say, the set of points of the which the given function attains its maximum value.

$$\underset{x}{\operatorname{argmax}}\ f(x) = \{x \mid \forall y : f(y) \leq f(x)\}$$

$$\underset{x}{\operatorname{argmax}}\ (-|x|) = \{0\}$$

Maximum-Likelihood Estimation (MLE)

- Likelihood: probability of observing sample under distribution d, which, given the independence assumption is

$$Pr[x_1, \ldots, x_m] = \prod_{i=1}^{m} p(x_i)$$

- MLE Principle: select a distribution maximizing the sample probability

$$p_* = argmax_{p \in \mathcal{P}} \prod_{i=1}^{m} p(x_i)$$ Likelihood

$$p_* = argmax_{p \in \mathcal{P}} \sum_{i=1}^{m} logp(x_i)$$ Log-Likelihood

Maximum-Likelihood Estimation (MLE)

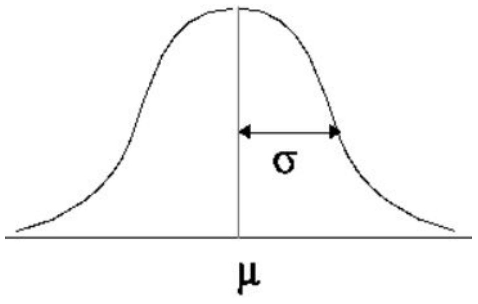
- Given training data D, MLE is to find the best hypothesis h that maximizes the likelihood of the training data

$$h_{ML} = argmax_{h \in (H)} P(D|h)$$

- What if you have some ideas about your hypothesis/parameters?

Example: Gaussian Distribution

- Task: find the most likely Gaussian distribution, given sequence of m real-valued observations: 3.23, 1.23, 0.55, 1.23, ….
- Normal distribution



$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2}$$

- Log-likelihood: $l(p) = -\frac{1}{2} m log(2\pi\sigma^2) - \sum_1^m \frac{(x_i-\mu)^2)}{2\sigma^2}$
- Solution (estimate mean and stand dev):

$$\frac{\partial l(p)}{\partial \mu} \Leftrightarrow \mu = \frac{1}{m} \sum x_i \qquad \frac{\partial l(p)}{\partial \sigma^2} \Leftrightarrow \sigma^2 = \frac{1}{m} \sum (x_i - \mu)^2$$

Bayes Theorem

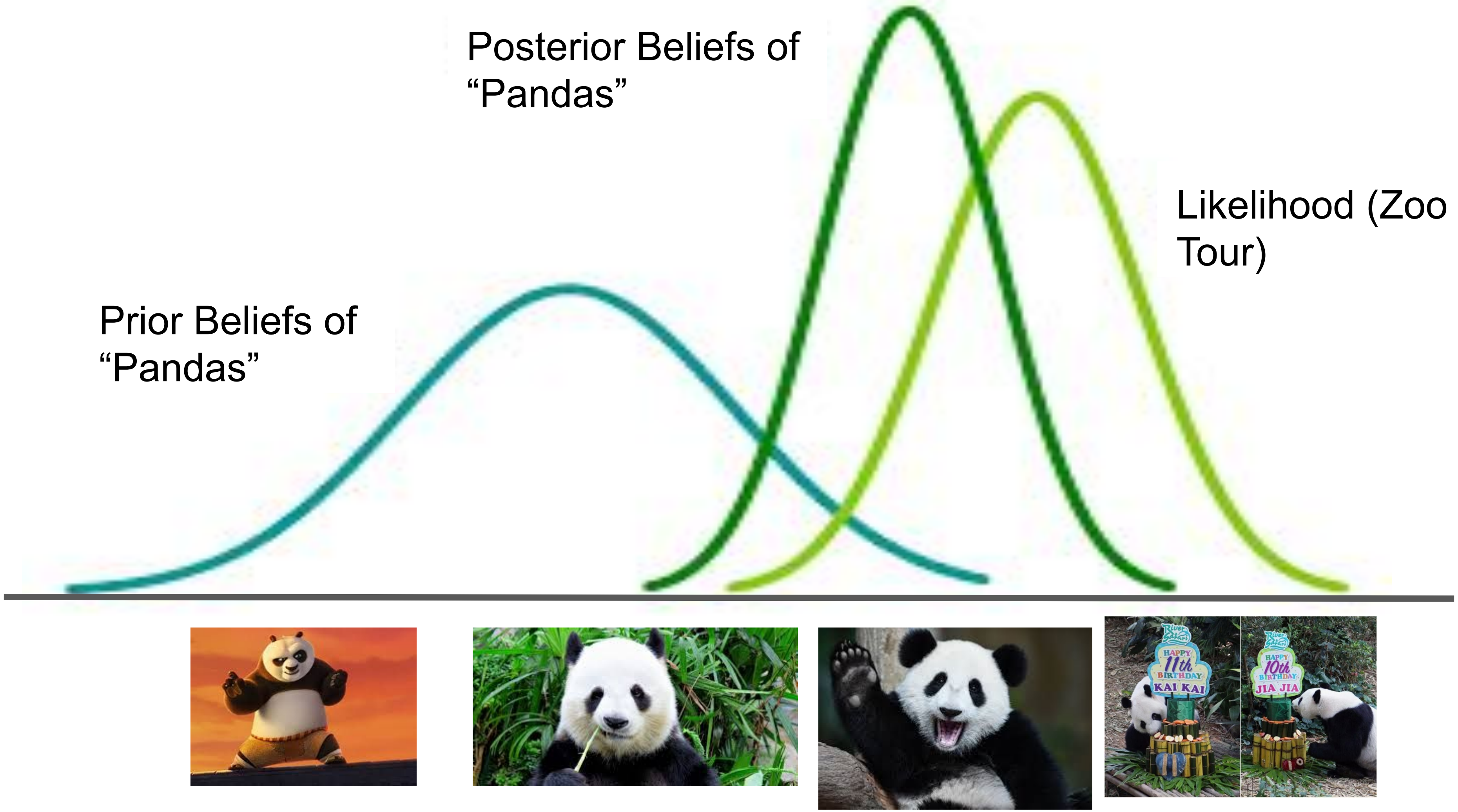
Bayes' Rule

- By definition of conditional probability:

$$\boxed{p(h|D)} = \frac{p(h,D)}{p(D)} = \frac{\boxed{p(D|h)}\boxed{p(h)}}{p(D)}$$

  - P(h): prior probability of hypothesis h
  - P(h|D): posterior probability of h given evidence D
  - P(D|h): likelihood of D given h
  - P(D): prior probability of evidence D

Reverse Probability

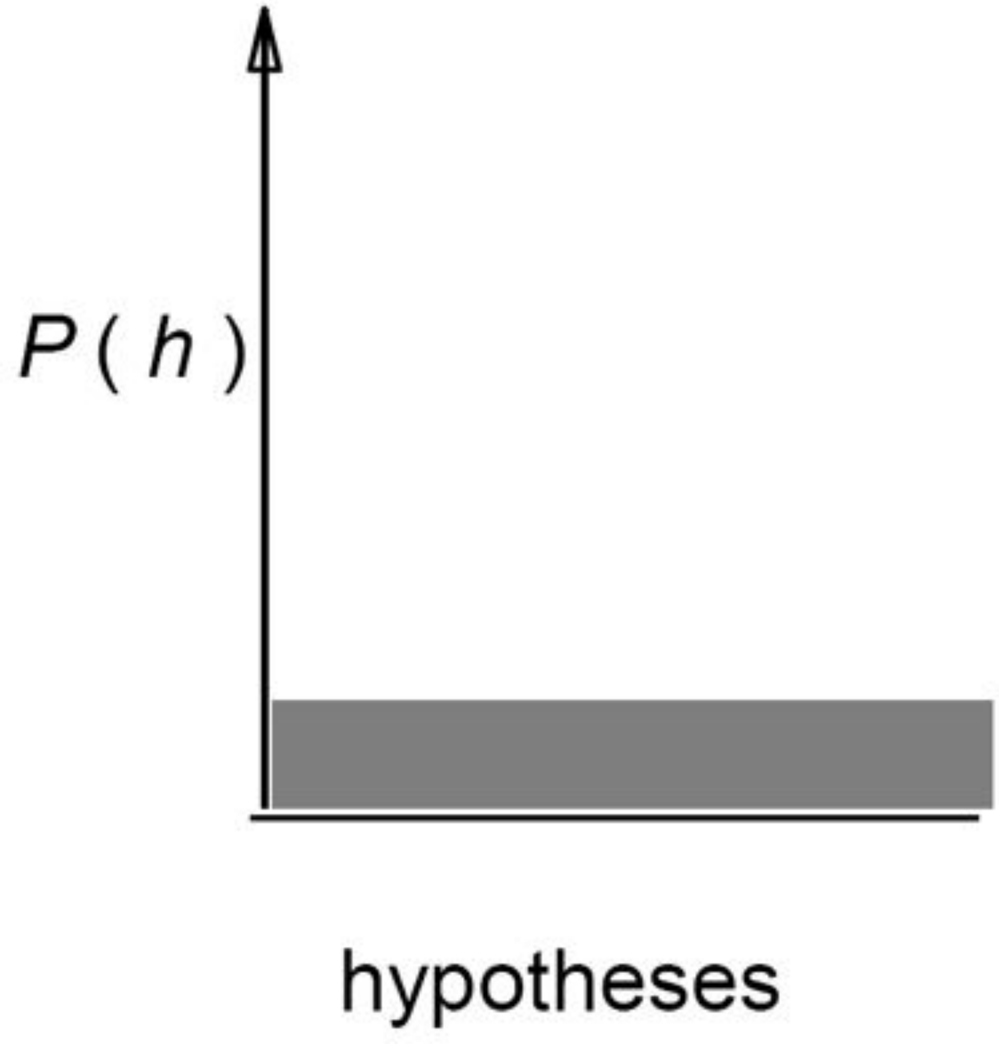

Posterior Beliefs of "Pandas"
Likelihood (Zoo Tour)
Prior Beliefs of "Pandas"
HAPPY 11th BIRTHDAY
KAI KAI
HAPPY 10th BIRTHDAY
JIA JIA

Example: BT5153 Student Model

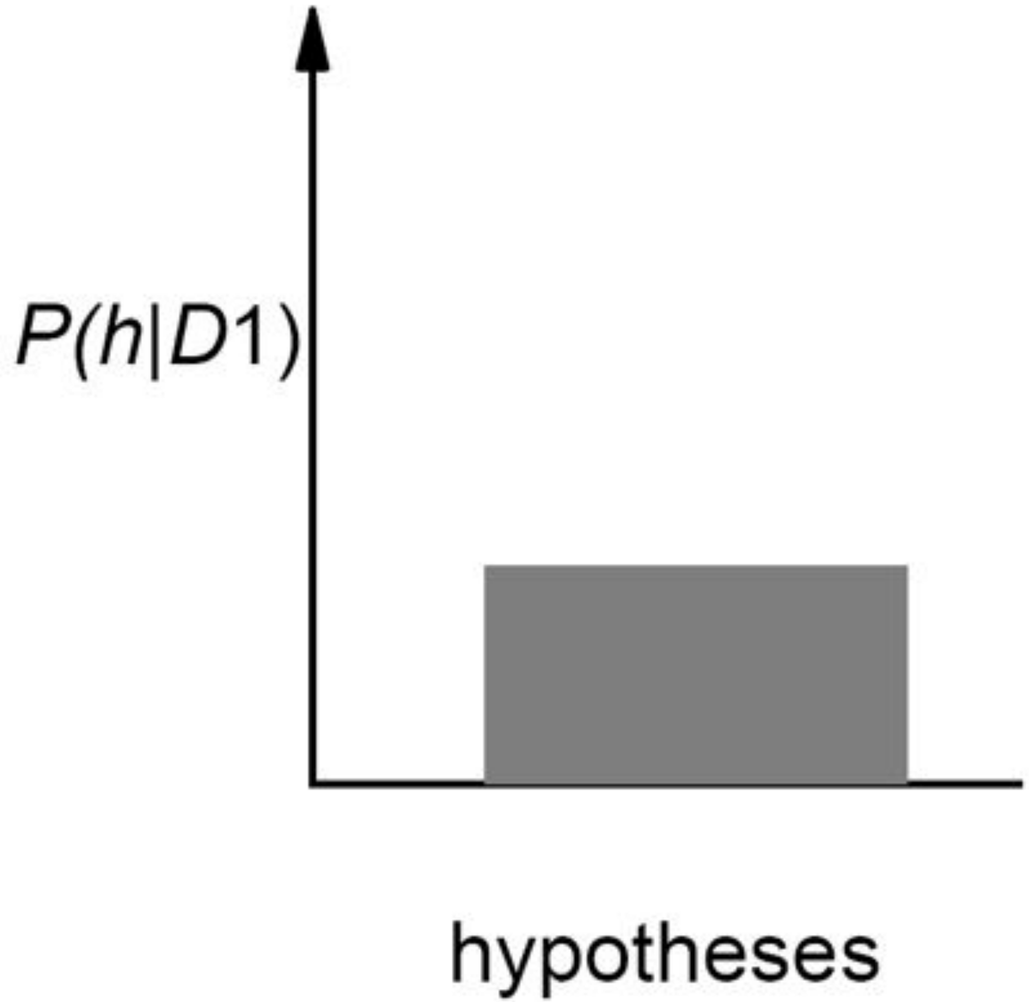
- Given:
  - The faculty knows that taking BT5153 causes that you stay in library **80%** of the time
  - Prior probability of any MSBA student taking BT5153 is **1/100**
  - Prior probability of any MSBA student staying in library is 1/10
- If a MSBA student stay in the library, what is the probability he/she took the BT5153?
  - D(Evidence) - Stay in Library h(hypothesis) - Taking BT5153

$$p(h|D) =$$

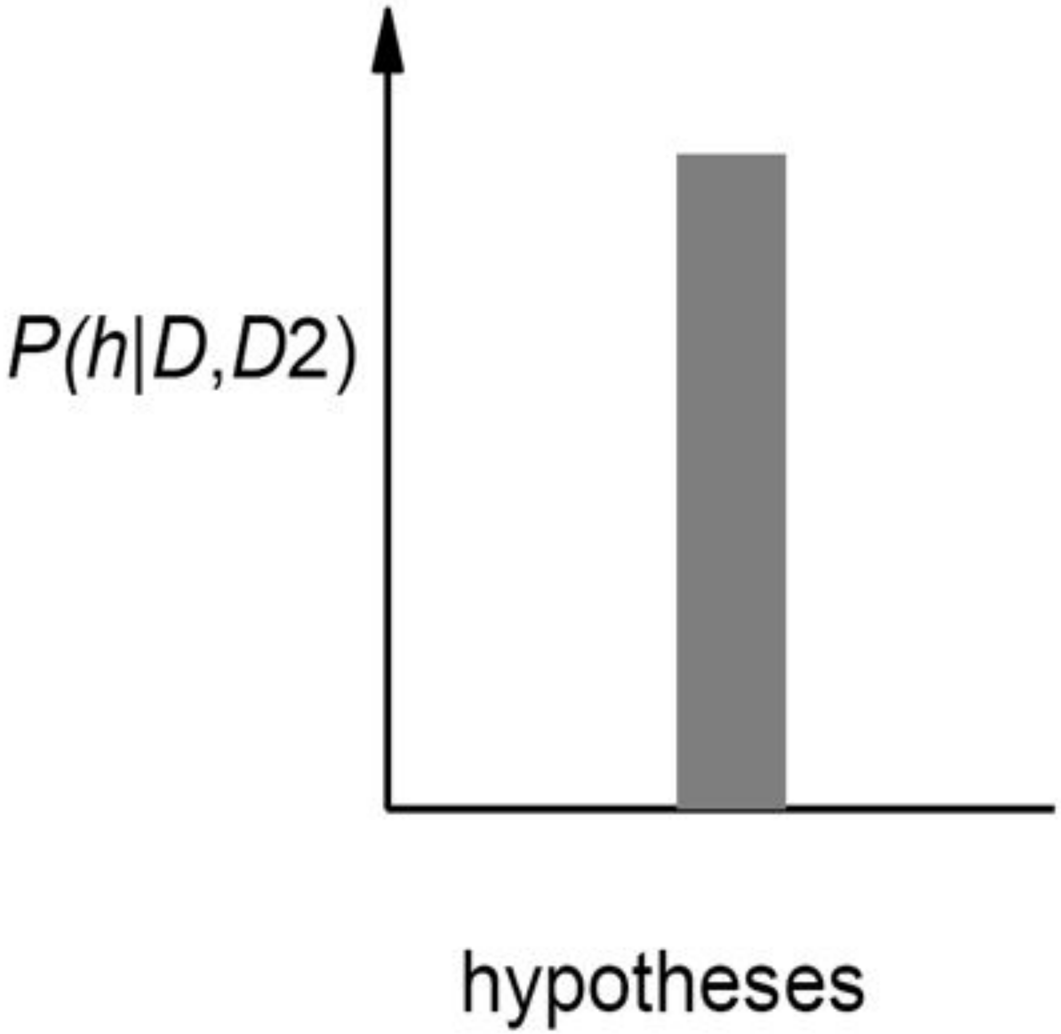
Evolution of Posterior Probabilities



Evolution of Posterior Probabilities



Evolution of Posterior Probabilities



Maximum A Posteriori

Find the most probable hypothesis given the training data (Maximum A Posteriori hypothesis H_{map})

$$h_{\text{MAP}} = \arg\max_{h \in \mathcal{H}} P(h|\mathcal{D})$$

$$= \arg\max_{h \in \mathcal{H}} \frac{P(\mathcal{D}|h)P(h)}{P(\mathcal{D})}$$

$$h_{\text{MAP}} = \arg\max_{h \in \mathcal{H}} P(\mathcal{D}|h)\boxed{P(h)}$$

Prior encodes the knowledge /preference

MAP vs MLE

- MLE: Finding a hypothesis h that maximizes the likelihood of the training data

$$h_{ML} = argmax_{h \in (H)} P(D|h)$$

- MAP: Finding a hypothesis h that maximizes the posterior probability given the training data

$$h_{MAP} = argmax_{h \in \mathcal{H}} P(h|D)$$

- When will MLE and MAP give the same results?

Classification Using Bayes Rule

$$\mathbf{d} = [d_1, d_2, \ldots, d_n]$$

Given multiple attribute values , what is the most probable value

features

$$h_{MAP} = \underset{h_i \in \mathbb{H}}{\operatorname{argmax}}\, p(h_i | d_1, d_2, \cdots, d_n)$$

$$= \underset{h_i \in \mathbb{H}}{\operatorname{argmax}} \frac{p(h_i) p(d_1, d_2, \cdots, d_n | h_i)}{p(d_1, d_2, \cdots, d_n)}$$

$$= \underset{h_i \in \mathbb{H}}{\operatorname{argmax}}\, p(h_i) p(d_1, d_2, \cdots, d_n | h_i)$$

Problem: too much data needed to estimate $p(d_1, d_2, \ldots, d_n | h_i)$ when n is large

Curse of Dimensionality

Naïve Bayes Classifier

- Hard to estimate $p(\mathbf{d}|h_i)$ for high dimensional data $\mathbf{d}$

- Conditional Independence assumption
  - All attributes are **conditionally independent**
  - assumption often *violated in practice*
  - even then, it usually works well

- Successful application: classification of text documents, Diagnosis

Conditional Independence

- $p(d_1,d_2,\ldots,d_n|\hbar_i) = p(d_1|\hbar_i) \times p(d_2|\hbar_i,d_1) \times p(d_3|\hbar_i,d_1,d_2) \times \ldots \times p(d_n|\hbar_i,d_1,d_2,\ldots,d_{n-1})$

- **Naïve Bayes** (conditionally independence) assumption : attributes are **independent**, given the class
  - $p(d_2|\hbar_i,d_1)=p(d_2|\hbar_i)$
  - $p(d_3|\hbar_i,d_1,d_2)=p(d_3|\hbar_i)$

    $\ldots,$
  - $p(d_n|\hbar_i,d_1,d_2,\ldots,d_{n-1})=p(d_n|\hbar_i)$
  - $\boldsymbol{p(d_1,d_2,\ldots,d_n|\hbar_i)=p(d_1|\hbar_i)*p(d_2|\hbar_i)\ldots p(d_n|\hbar_i)}$

Naïve Bayes Classifier

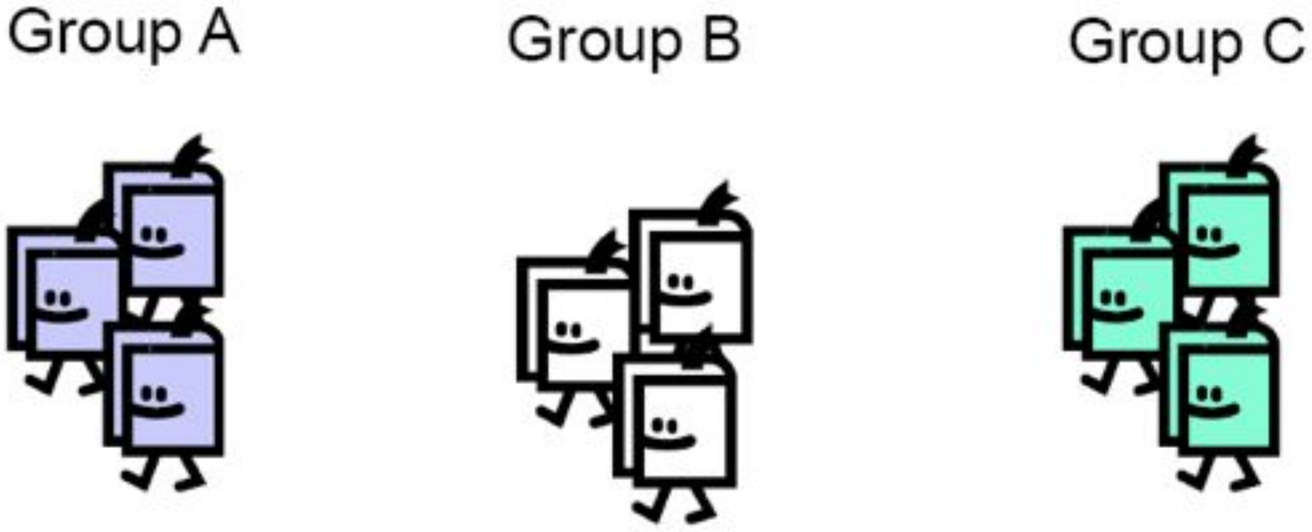
Based on Bayes' rule + assumption of conditional independence

$$
\begin{aligned}
h_{NB} &= \underset{h_i \in \mathbb{H}}{\operatorname{argmax}}\, p(h_i | d_1, d_2, \cdots, d_n) \\
&= \underset{h_i \in \mathbb{H}}{\operatorname{argmax}}\, \frac{p(h_i) p(d_1, d_2, \cdots, d_n | h_i)}{p(d_1, d_2, \cdots, d_n)} \\
&= \underset{h_i \in \mathbb{H}}{\operatorname{argmax}}\, p(h_i) p(d_1, d_2, \cdots, d_n | h_i) \\
&= \underset{h_i \in \mathbb{H}}{\operatorname{argmax}}\, p(h_i) \prod_{j=1}^{n} p(d_j | h_i)
\end{aligned}
$$

Text Classification using Naive Bayes

Text Classification

- Given text of newsgroup article, guess which newsgroup it is taken from.
- Naïve Bayes turns out to work well on this application.
- Key issue : how do we represent examples? what are the attributes?



Text Classification

- Class h_j: Binary classification (+/−) or multiple classes possible H (j = 1,2, …,k)

- How about attributes?

Example

- 1000 training documents that someone has 700 classified as “dislikes” (h_0) and 300 classified as “likes” (h_1).
- Suppose document 1 is “**This is a very interesting document**”

$$h_{NB} = \max_{h_j \in \{\text{like,dislike}\}} p(h_j) \times p(d_1 = \text{this}|h_j) \times$$
$$p(d_2 = \text{is}|h_j) \cdots \times p(d_6 = \text{document}|h_j)$$

$p(like)$=300/1000=0.3

$p(dislike)$=1−$p(like)$=0.7

- How to estimate $p(d_i|h_j)$?

Parameter Estimation

- Learning by Maximum Likelihood Estimate
  - Simply count the frequencies in the data

$$p(d_i = w|h_j) = \frac{count(w,h_j)}{\sum_{d\in\mathcal{V}} count(d_i,h_j)}$$

The count of the specific word di=w in the mega-doc

The count of total words in the mega-doc

  - Create a mega-document for class hj by concatenating all the docs in this class
  - Compute the frequency of the word w in the mega-document

New Word Problem

- What if some words do not exit a certain category: h

$$p(d_i = newword|h) = 0$$

- The predicted likelihood will be zero

$$p(\boldsymbol{d}|h_i) = p(d_1|h_i)*p(d_2|h_i)...p(d_n|h_i) = p(d_1|h_i)*p(d_2|h_i)...*0*p(d_n|h_i) = 0$$

How to Solve it?

Additive Smoothing

smoothing parameter

$$p(d_i = w|h_j) = \frac{count(w,h_j)+\alpha}{\sum_{d\in\mathbb{V}} count(d_i,h_j)+\alpha V}$$

Vocab. Size

- A weighted estimation of
  - Relative frequency: $\frac{count(w,h_j)}{\sum_{d\in\mathcal{V}} count(d_i,h_j)}$
  - Uniform probability: $\frac{1}{V}$

`sklearn.naive_bayes`.MultinomialNB

class `sklearn.naive_bayes.`**`MultinomialNB`**(*alpha=1.0*, *fit_prior=True*, *class_prior=None*) [source]

Naive Bayes classifier for multinomial models

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

Read more in the User Guide.