Week 2: From text to vectors

Text Analytics and Natural Language Processing
Instructor: Benjamin Batorsky

Please download Week 2 notebook from Canvas!

Schedule for materials (anticipated!)

- Morning Mondays - Assignments/Notebook
- Afternoon Mondays - Slides
- Morning Thursday
  - Instructor notebook
  - Instructor assignment (for assignment due that week)
  - Final slides

Assignment 1

Extended deadline: 6:30pm on Wednesday 7/1

End of submission period: 11:59 Wednesday 7/1

Assignment 2: 6:30pm on Monday 7/6

General notes on assignments

- Take a look at the assignment ahead of time
- Order of operations for help:
  - Read the notes/comments
  - Check Piazza for an answer to your question
  - Write a new Piazza question
  - If no answers/still stuck: Contact us via Canvas
- When doing assignments: We want to know what you’re thinking
  - Add as markdown/comments
  - If we don’t understand your choice AND you don’t explain it, we can’t give you points!
- Get familiar with using Collab
  - Loading/saving files
  - Restarting runtimes
  - How to handle timeouts
  - Tracking RAM usage

Log-likelihood complexities (Assignment 1 review)

Example 1:

Corpus A: 10 documents, 100 unique words

Corpus B: 100 documents, 1000 unique words

Count of “good” in A = 10

Count of “good” in B = 10

Is the likelihood of observing “good” in corpus A significantly different from the likelihood of observing “good” in B?

Log-likelihood complexities (Assignment 1 review)

Example 1:

Corpus A: 10 documents, 100 unique words

Corpus B: 100 documents, 1000 unique words

Count of “good” in A = 10

Count of “good” in B = 10

Is the likelihood of observing “good” in corpus A significantly different from the likelihood of observing “good” in B?

```
a = 10
b = 10
c = 100
d = 1000
e1 = c*(a+b)/(c+d)
e2 = d*(a+b)/(c+d)
2*((a*log(a/e1)) + (b*log(b/e2)))
```

```
22.13822182965609
```

Log-likelihood complexities (Assignment 1 review)

Example 2:

Corpus A: 10 documents, 100 unique words

Corpus B: 100 documents, 1000 unique words

Count of “bad” in A = 10

Count of “bad” in B = **<u>100</u>**

Is the likelihood of observing “bad” in corpus A significantly different from the likelihood of observing “bad” in B?

Log-likelihood complexities (Assignment 1 review)

Example 2:

Corpus A: 10 documents, 100 unique words

Corpus B: 100 documents, 1000 unique words

Count of “bad” in A = 10

Count of “bad” in B = **100**

Is the likelihood of observing “bad” in corpus A significantly different from the likelihood of observing “bad” in B?

```
a = 10
b = 100
c = 100
d = 1000
e1 = c*(a+b)/(c+d)
e2 = d*(a+b)/(c+d)
2*((a*log(a/e1)) + (b*log(b/e2)))
```

```
0.0
```

Log-likelihood complexities (Assignment 1 review)

- Measure is “likelihood”, how likely it is to encounter a particular word
- Subject to the number of words in the corpus
  - Which is subject to your vocabulary!

Differences between en_core_web_sm and English model

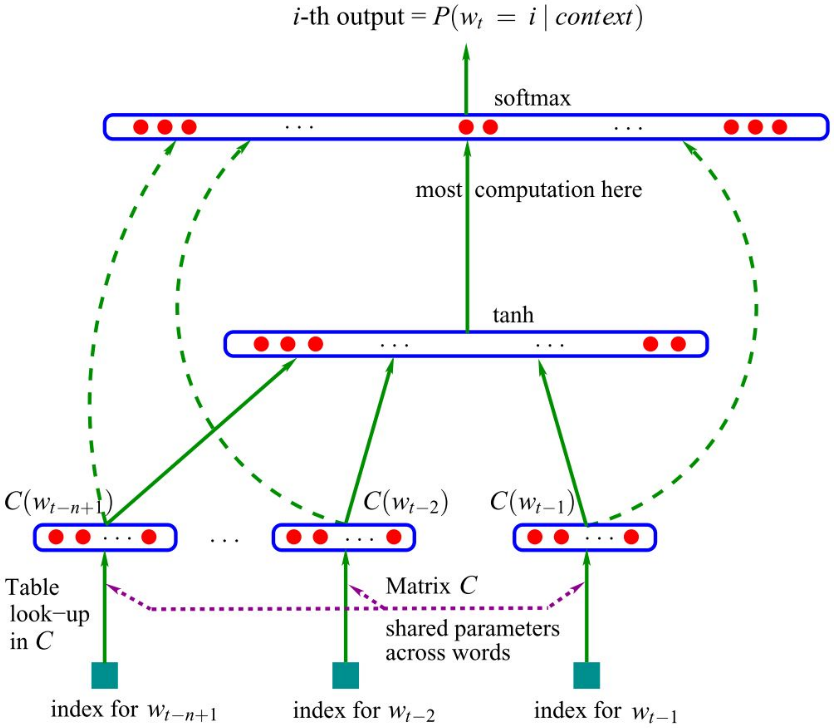
- Base English model
  - Time to process doc of 100 words: ~
  - Gives you
    - Basic tokenization
    - .is_stop
    - .is_url
- en_core_web_sm (simple trained English model)
  - Time to process doc of 100 words: ~
  - Gives you
    - .ents (NER)
    - .pos (Part of Speech)
    - .lemma_ (lemmatized form)

Review: History of NLP

- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
    - Also expansion of available data

2000s: Advent of Neural Models for NLP

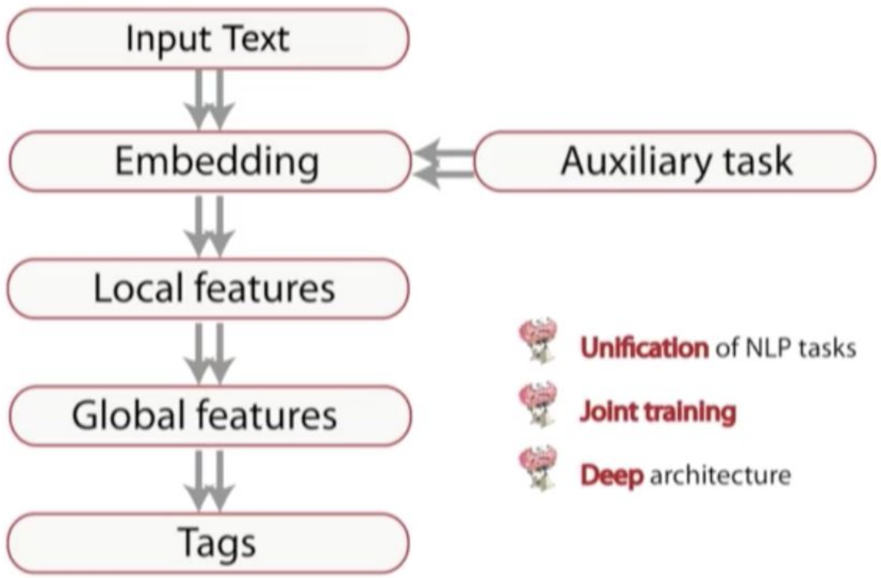
- Focused on language modelling task
  - Given context words, predict next word
  - “Do you want to go out for <mask>”
  - “I ate so much I am so <mask>”
  - “I’m so tired, I think I’ll take a <mask>”
- Apply Neural Net architectures to language modelling task
  - Neural Net: Model that connects inputs to outputs through sets of computations
- Bengio et. al. (2001) [A Neural Probabilistic Language Model]
  - Context words fed into a matrix that “represents” the information
  - These representations then fed into a computation layer
  - Output: Prediction of the target word
    - “Full”: 70% likely, “tired” 20% likely, etc



[A Neural Probabilistic Language Model]
(https://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf)

2008: Multi-task learning

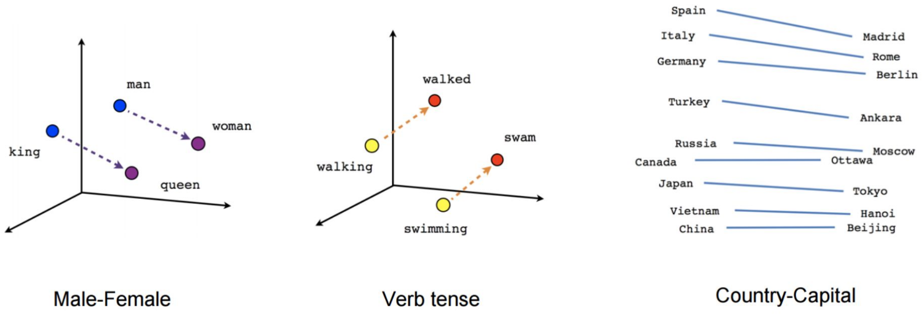
- Multi-task learning: One model, multiple related tasks
  - Trains representations (e.g. lookup tables) jointly so that it performs well on both
- Collobert and Weston (2008)
  - Word lookup table trained jointly from two different tasks
  - Basically the precursor to the idea of word embeddings



https://www.facebook.com/722677142/posts/10155393881507143/

2013: Word embeddings

- Mikolov paper
  - Efficient computation of embedding technique used in the 2000s
  - Word2Vec implementation
- Huge amount of adaptation/modifications
  - GloVe: Co-occurance based
  - Sub-word/character-level
  - Use of different corpora for training
  - Out-of-vocabulary handling
  - Training implementation in open-source libraries (e.g. gensim)
- Major revolution: It's fairly easy to make more informed representations of words
- Caveat: Widespread use raises issues of bias
  - Extensive research finds training on most large datasets introduces gender/racial bias
  - Not an issue with the method particularly, more with irresponsible use



[1301.3781] Efficient Estimation of Word Representations in Vector Space

Review: Tokenization

- Introduced spaCy
  - Language models including various components
- Token: Useful semantic unit
- How to construct these?
  - Lower/uppercase
  - Non-alpha characters (Numbers, punctuation, whitespace)
  - Non-typical tokens (e.g. entities, URLS)
  - Stemming vs lemmatization
- Intro to vectorization (CountVectorizer)
- Let’s review in notebook!

Document-Term and Term-Term matrices

- Information Retrieval: Extract signal from noise
  - Useful representations of the documents/terms
- Document representation in vocabulary space
  - Document - Term Matrix (DTM)
  - DTM = Documents x vocabulary
- Term representation in vocabulary space
  - How often two terms occur in the same document
  - DTM * inverse DTM = Term-Term matrix (TTM)
- We can do some neat things with just these vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	
strawberry	0	...	0	0	1	60	19	
digital	0	...	1670	1683	85	5	4	
information	0	...	3325	3982	378	5	13	

Figure 6.5 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Issues with raw word counts

- NLP: Turn text into information

Top ten words for negative movie reviews

```
[('the', 15365),
 ('a', 7548),
 ('and', 6978),
 ('to', 6780),
 ('of', 6402),
 ('is', 4952),
 ('it', 4354),
 ('i', 4248),
 ('in', 4203),
 ('this', 3837)]
```

Issues with raw word counts

- NLP: Turn text into information
- Raw word count = each word counted the same
    - “This book is about biology” vs “This book is about history”
- Ways to reduce the noise
    - Reducing to common forms
    - Stripping uninformative words (“the”, “and”)
- More standard way up upweighting important words, discounting unimportant ones

Top ten words for negative movie reviews

```
[('the', 15365),
 ('a', 7548),
 ('and', 6978),
 ('to', 6780),
 ('of', 6402),
 ('is', 4952),
 ('it', 4354),
 ('i', 4248),
 ('in', 4203),
 ('this', 3837)]
```

Term Frequency - Inverse Document Frequency (TF-IDF)

- Term frequency: Count of term (or token) within a document
- Document frequency: Count of documents within which a term appears
  - Usually divided by the total number of documents
- Inverse document frequency: (Number of documents) / DF
  - Higher DF = Lower weight, Lower DF = Higher weight
  - Log-transformed to handle very frequent/infrequent terms
    - Preserves the “order” of IDF
- TF*IDF, term count weighted by how “informative” that term is
  - In Scikit-learn: Normalized by the Euclidean Norm to handle document length variability

TF-IDF

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Figure 6.8 A tf-idf weighted term-document matrix for four words in four Shakespeare plays, using the counts in Fig. 6.2. For example the 0.049 value for *wit* in *As You Like It* is the product of $\text{tf} = \log_{10}(20+1) = 1.322$ and $\text{idf} = .037$. Note that the idf weighting has eliminated the importance of the ubiquitous word *good* and vastly reduced the impact of the almost-ubiquitous word *fool*.

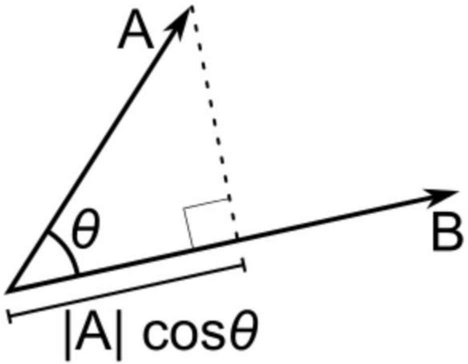
https://web.stanford.edu/~jurafsky/slp3/6.pdf

Implementation in sklearn

Cosine similarity

- Document-term vector: Document in vocabulary space
    - 100 tokens in vocab = 100 dimensional vector
- Similarity: Compare document-term vectors
- Dot product of vectors: One vector's projection on another
- Measure vector orientation to one another, regardless of magnitude
- Can't really plot to find angle
- Equivalent: Dot product divided by vector norm

$$\vec{a} \cdot \vec{b} = \|\vec{b}\| \|\vec{a}\| \cos\theta$$



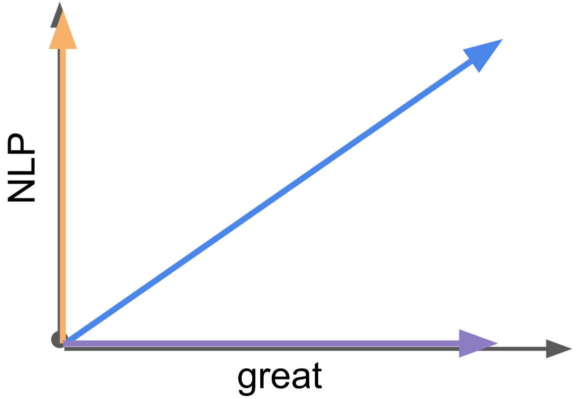
http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

Cosine similarity for documents and words

Doc 1: “NLP is great”

Doc 2: “NLP is hard”

Doc 3: “Math is great”



sim(Doc 1, Doc 2) = cos(45) = .71

sim(Doc 2, Doc 3) = cos(90) = 0

As document-term matrix (DTM)

NLP	great
1	1
1	0
0	1

$$||Doc1|| = \sqrt{1^2 + 1^2} = \sqrt{2}$$

$$||Doc2|| = \sqrt{1^2 + 0^2} = \sqrt{1}$$

$$Doc1^t \cdot Doc2 = 1 * 1 + 1 * 0 = 1$$

sim(Doc 1, Doc 2) = 0.71

Cosine similarity example

Further issues with word counts (including TF-IDF)

Book, author, year	Unique words	Words	Words per unique word
Sense & Sensibility by Jane Austen (1811)	7,265	119,893	16.5
A Tale of Two Cities by Charles Dickens (1859)	10,778	137,137	12.7
The Adventures of Tom Sawyer by Mark Twain (1876)	7,896	71,122	9
The Hobbit by JRR Tolkien (1937)	6,911	96,072	13.9
The Lion, The Witch, and The Wardrobe by C.S. Lewis (1950)	3,520	39,166	11.1
Harry Potter and The Sorcerer's Stone by J.K. Rowling (1998)	6,185	77,883	12.6
Twilight by Stephenie Meyer (2005)	8,507	119,270	14

http://www.tylervigen.com/literature-statistics

In [7]:
```
%%time
cosine_similarity([[1]*3,
                  [0.5]*3])
```

CPU times: user 285 µs, sys: 12 µs, total: 297 µs
Wall time: 290 µs

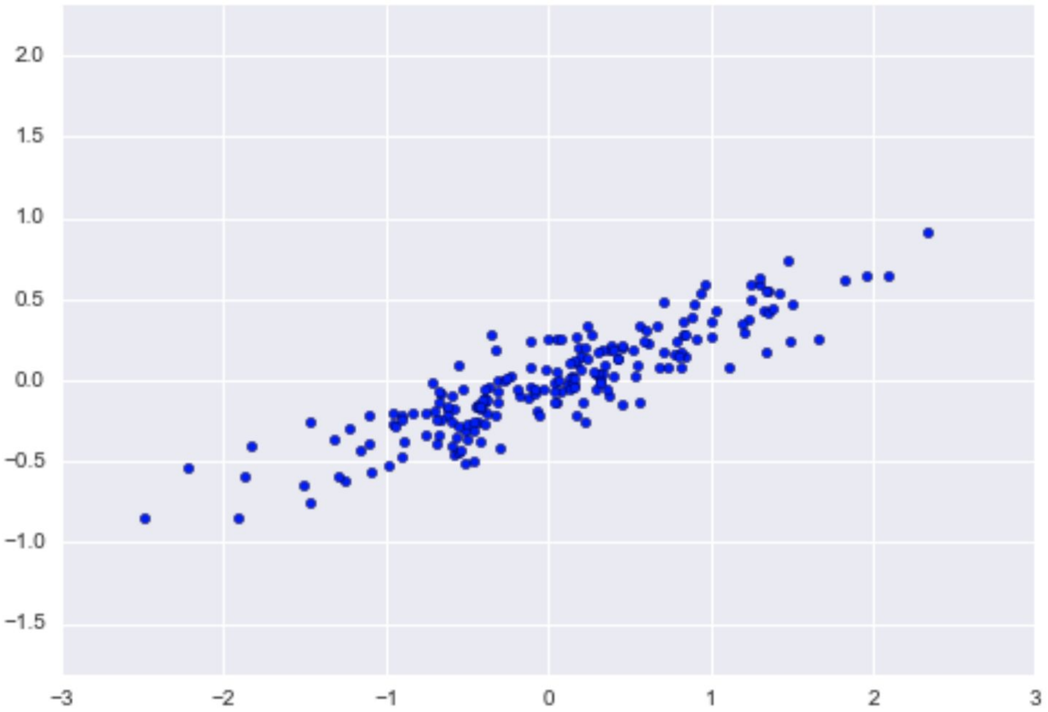
In [9]:
```
%%time
cosine_similarity([[1]*300,
                  [0.5]*300])
```

CPU times: user 385 µs, sys: 56 µs, total: 441 µs
Wall time: 411 µs

Topic models

- “Topic models are algorithms for discovering the main themes that pervade a large and otherwise unstructured collection of documents” (Blei 2012)
- Document = f(topics), Topics = g(words)
  - Typically number of topics << size of vocabulary
  - Want to minimize the information lost by representing in this way
- Typically for unsupervised problems
  - Creating topics when you don’t already have them labelled

Extracting axes of variation in data

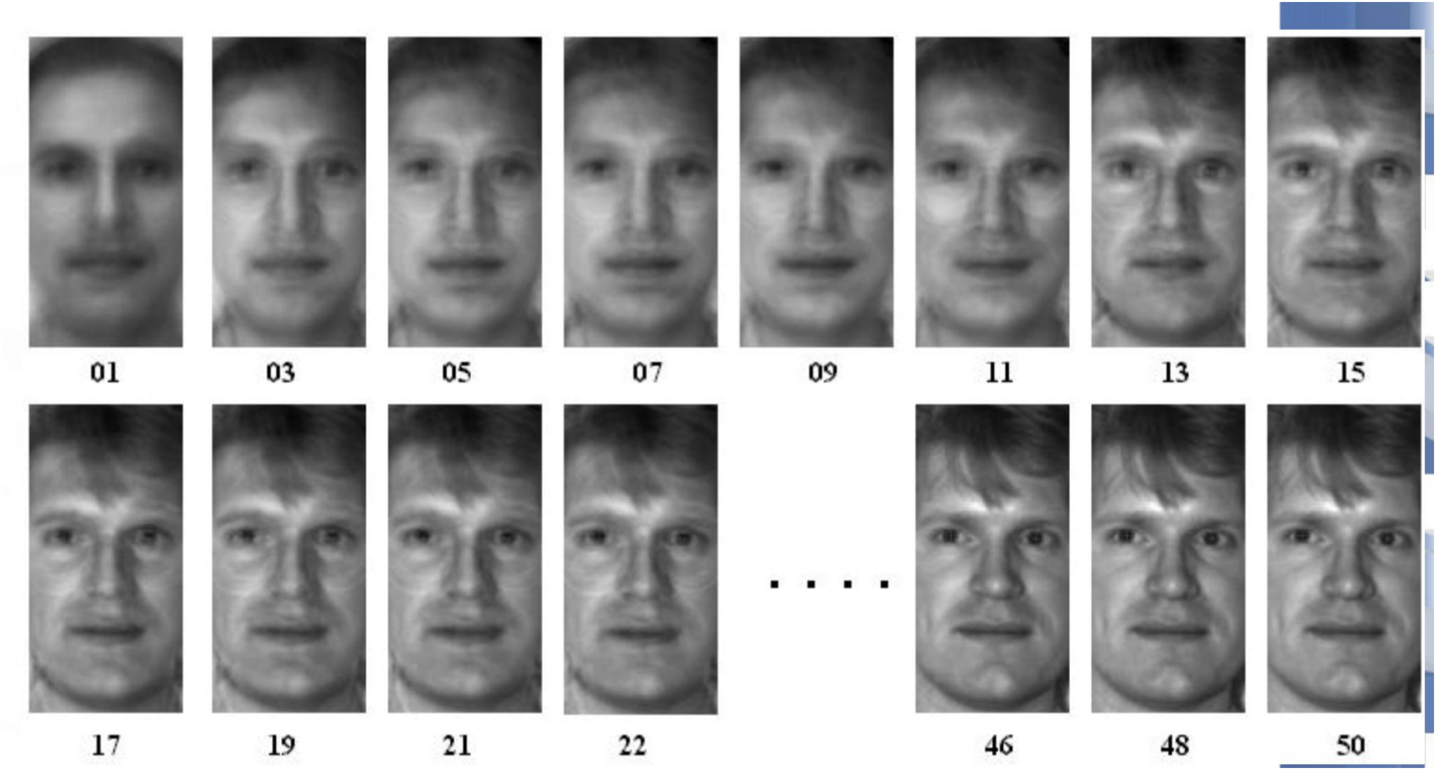


https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Extracting axes of variation in data

https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Application in image processing



https://www.cec.uchile.cl/~jruizd/faces/reconstruction/rec.htm

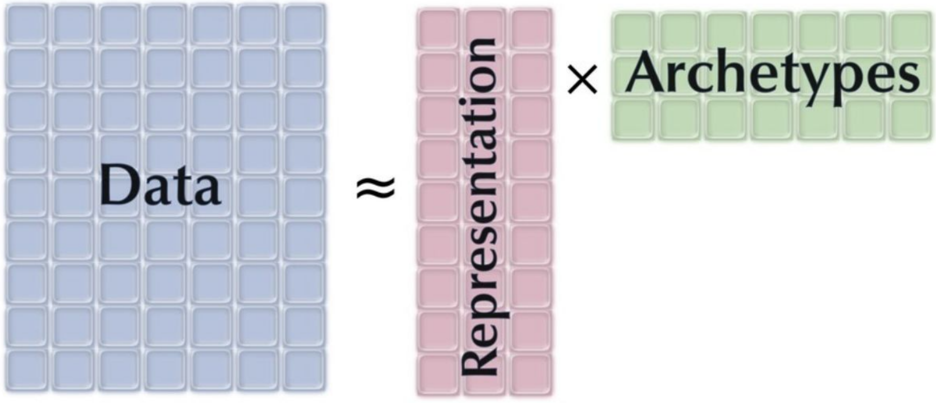
Intro to Matrix Factorization (Leland McInnes)

https://youtu.be/9iol3Lk6kyU?t=228

Distilling text vectors with matrix factorization

- Matrix factorization: Decomposing a matrix into archetypes and values
- In NLP: Extracting "latent" structure of the association between terms and documents
- The number of archetypes is typically lower than the number of features

Visual of factorization



X U V

Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{D} \mathbf{Loss}\left(X_{ij},\ (UV)_{ij}\right)$$

Leland McInnes: Bluffer's Guide to Matrix Factorization
https://www.youtube.com/watch?v=9iol3Lk6kyU
https://speakerdeck.com/lmcinnes/a-guide-to-dimension-reduction

Latent Semantic Indexing (LSI)

Data

	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
voyage	1	0	0	1	1	0
trip	0	0	0	1	0	1

Archetypes

	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
voyage	−0.70	0.35	0.15	−0.58	0.16
trip	−0.26	0.65	−0.41	0.58	−0.09

Representation

	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

https://nlp.stanford.edu/IR-book/pdf/18lsi.pdf

LSI vs Non-negative Matrix Factorization (NMF)

LSI

Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left(X_{ij}-(UV)_{ij}\right)^2$$

with no constraints

NMF

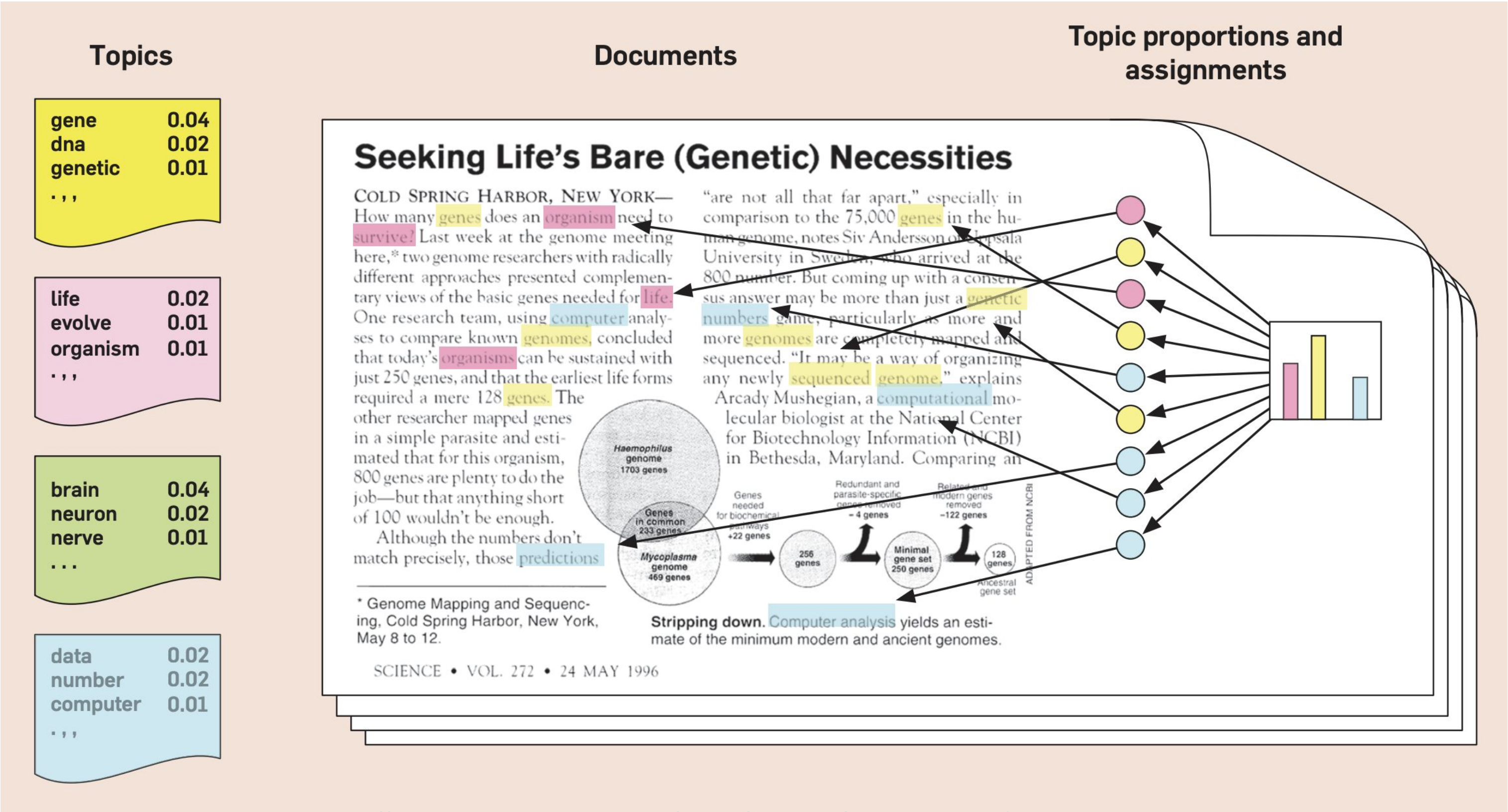
Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left(X_{ij}-(UV)_{ij}\right)^2$$

Subject to

$U_{ij} \geq 0$ **and** $V_{ij} \geq 0$

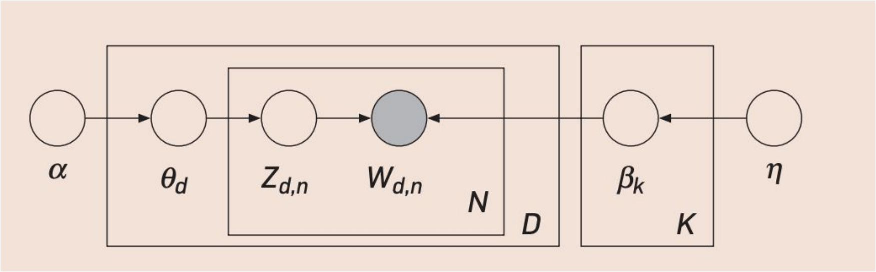
Latent Dirichlet Allocation (LDA)



http://www.cs.columbia.edu/~blei/papers/Blei2012.pdf

The LDA generative process

- α/η = parameters governing the distributions from which theta+beta are drawn
- K = topics,
- D = docs
- N = words
- $\boldsymbol{\theta}$ = document's distribution over topics
- β = word distribution over topics
- Z = the topic assignment of word n in document d



http://www.cs.columbia.edu/~blei/papers/Blei2012.pdf

NMF + LDA implementation

Issues with topic modelling

- Which is better, NMF or LDA?
- How do we measure performance in topic models?
  - Perplexity
  - Reconstruction error
- Ground truth
  - The “true” topics of the documents

NMF

```
Topic 0:
film films good seen does
Topic 1:
movie watch good movies time
Topic 2:
man life story family young
Topic 3:
horror effects special budget gore
Topic 4:
br money music audience thing
```

LDA

```
Topic 0:
film like movie just good
Topic 1:
movie film like just really
Topic 2:
did movie film like funny
Topic 3:
film like just story good
Topic 4:
film man films story like
```

What’s the difference?

Unsupervised learning

Supervised Learning

What’s the difference?

Unsupervised learning

- Goal: To create informative representations
- Can be applied to any dataset
- Performance typically quantifying loss in representation

Supervised Learning

- Goal: To predict an outcome
- Requires input/output pairs (labels!)
- Performance assessed against output

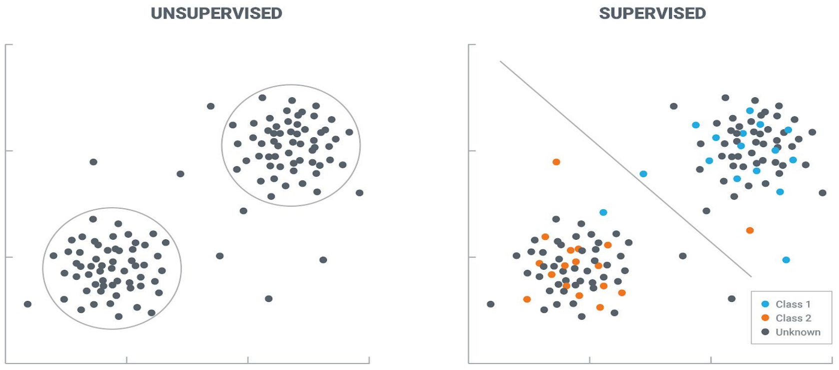
Definitions (Wikipedia)

Unsupervised learning

Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum (

Supervised Learning

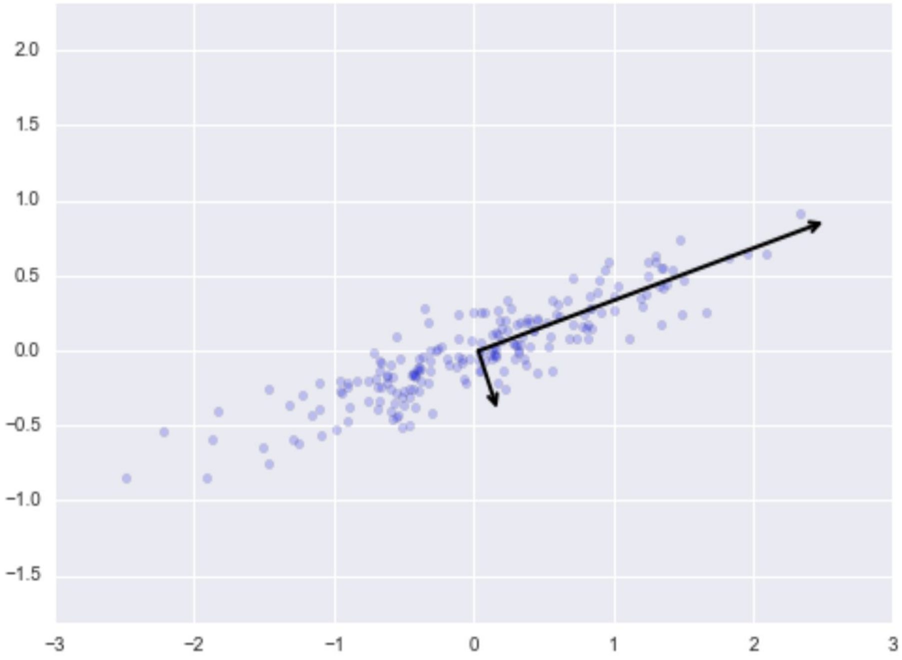
Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.



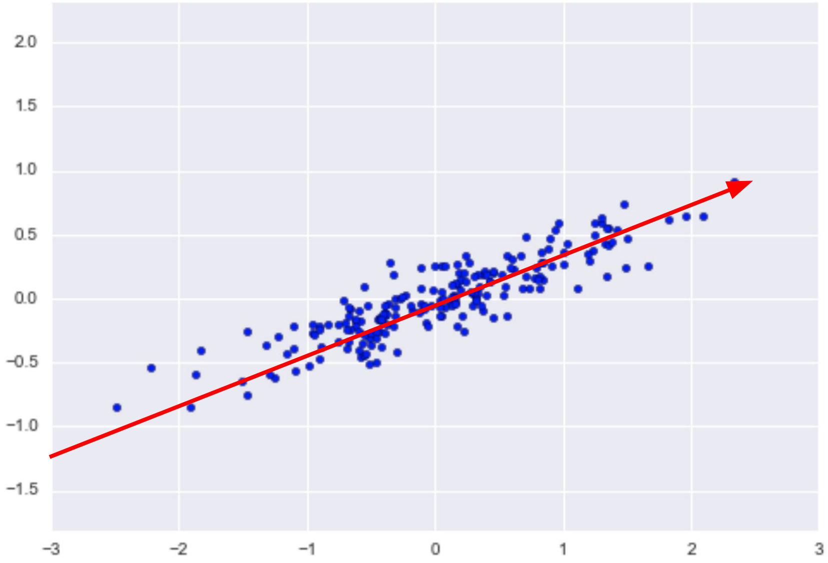
https://lawtomated.com/supervised-vs-unsupervised-learning-which-is-better/

Pattern identification vs regression

Find dimensions with most variation

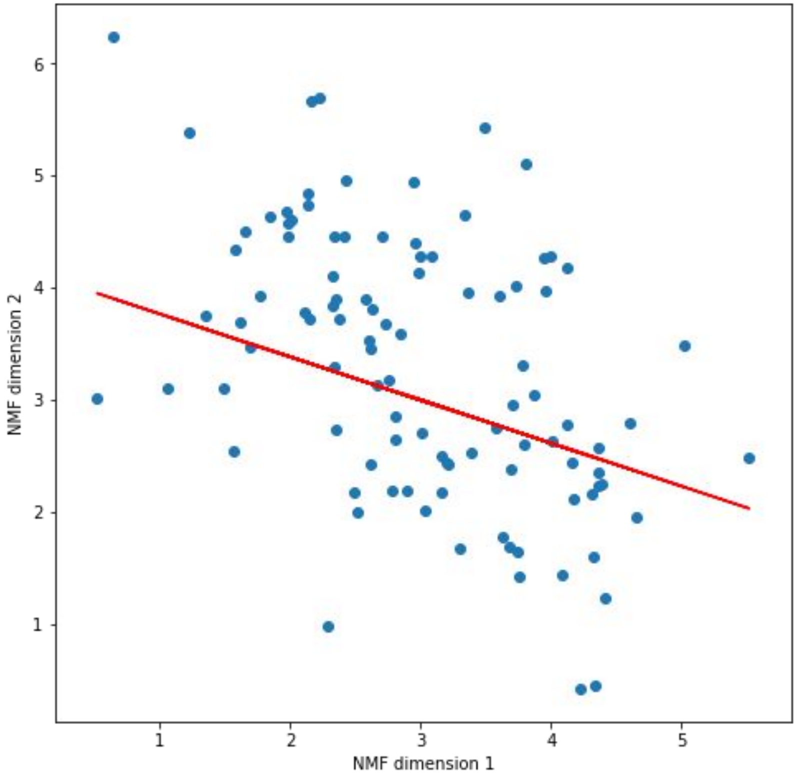


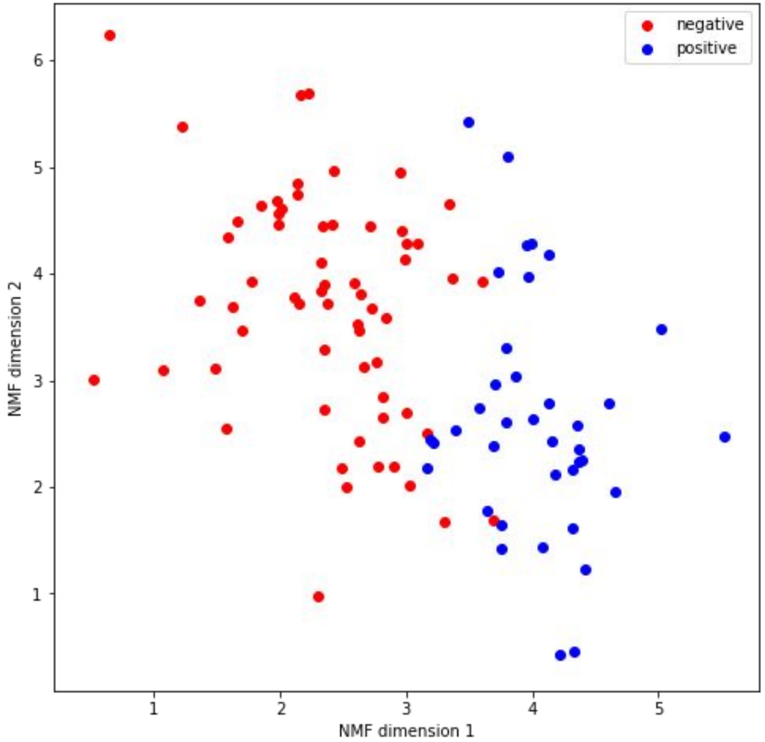
Given X, predict Y



https://jakevdp.github.io/PythonDataScienceHandbook/05.09-principal-component-analysis.html

Regression vs classification

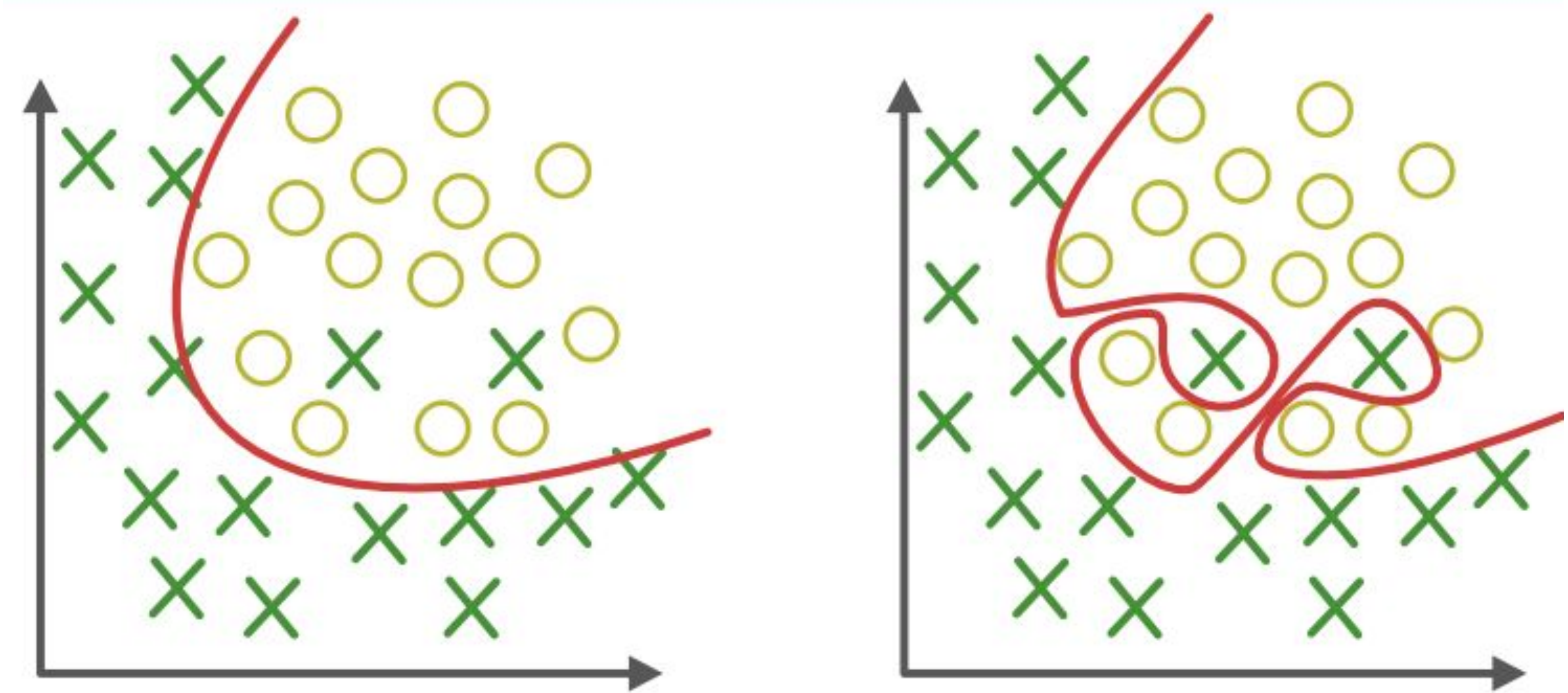




Which model is better?

Accuracy: 93%

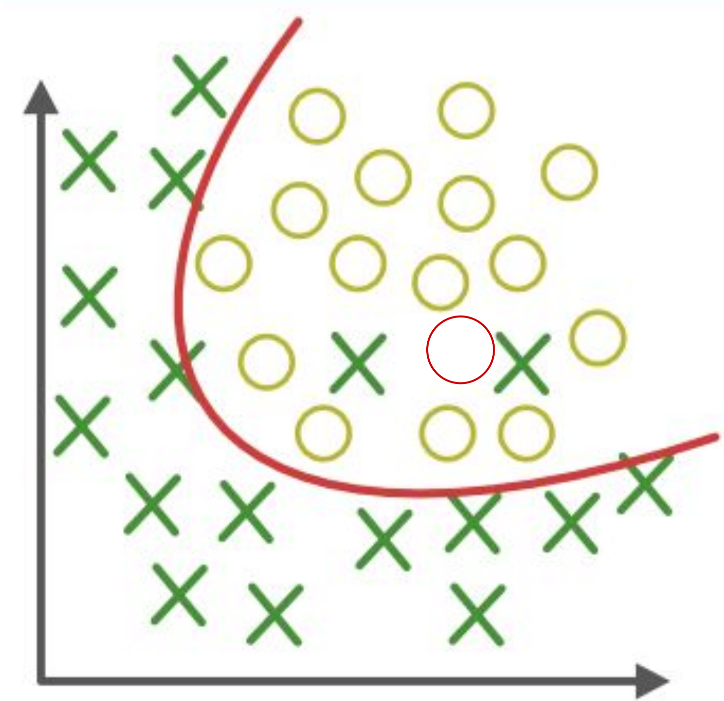
Accuracy: 100%



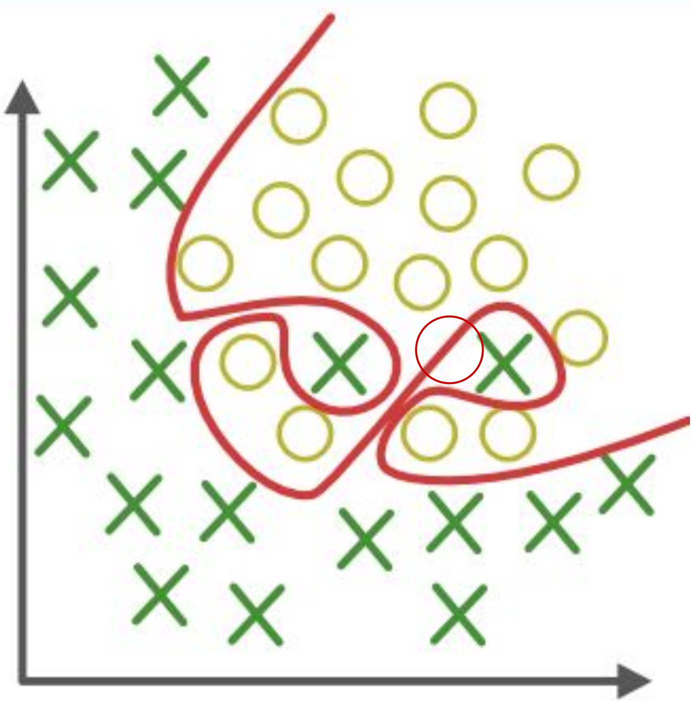
https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

Which model is better?

Accuracy: 94%



Accuracy: 96%

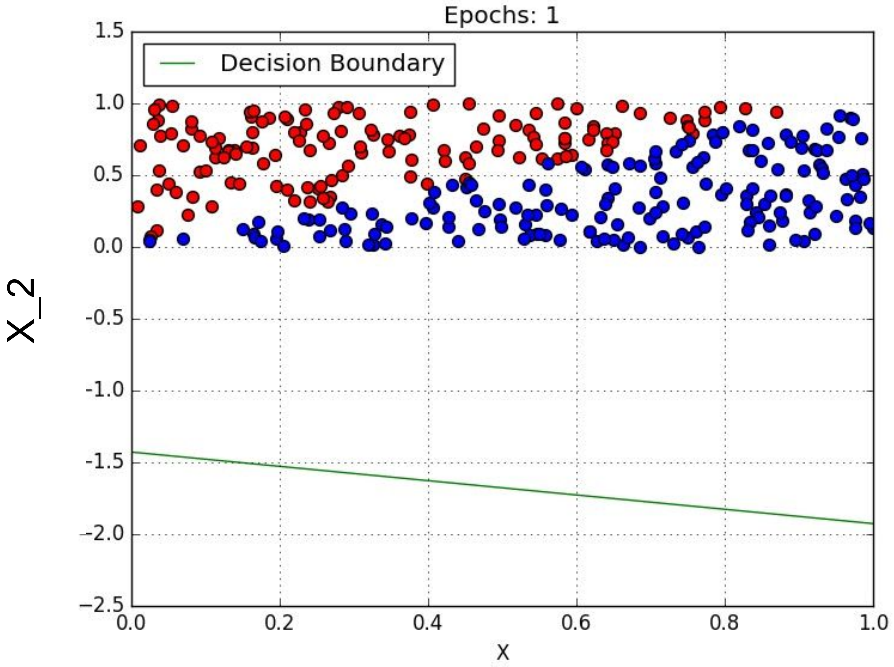


https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

Training, validation and testing

- Training a model: Fitting the model to the data
  - Regression: Predict the value of outcome Y given X
  - Classification: Predict the class of Y given X
- Validation
  - Simulating a situation where you have unseen data
  - Data held out from training, useful for tuning parameters/tracking training
- Testing
  - Dataset for performance measurement
  - Typically used for final assessment
- Model is fit to train, tweaked according to validation, evaluated on test

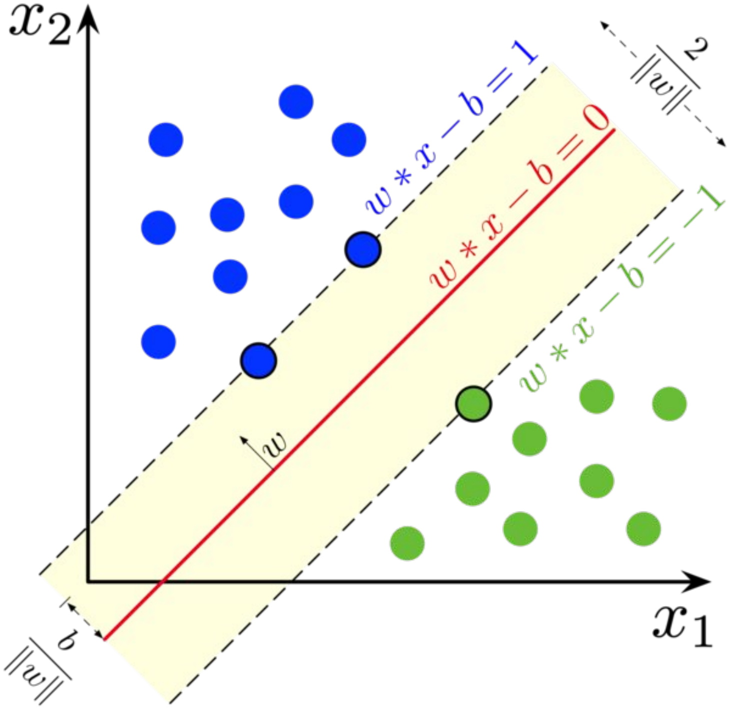
Support Vector Machines - Finding a decision boundary



Blue = Positive
Red = Negative

https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Intuition behind Support Vector Machines



https://en.wikipedia.org/wiki/Support_vector_machine