Winter semester 2021/22

344.175 VL: Natural Language Processing

Sentiment Analysis with Machine Learning

Navid Rekab-saz

navid.rekabsaz@jku.at

JOHANNES KEPLER UNIVERSITY LINZ

Institute of Computational Perception

Agenda

- Machine Learning – a quick tour
- Sentiment Analysis with Bag of Words
- Low-dimensional representations with SVD

Notation

- $a \longrightarrow$ a value or a scalar

- $\boldsymbol{b} \longrightarrow$ an array or a vector
  - i^{th} element of $\boldsymbol{b}$ is the scalar b_i

- $\boldsymbol{C} \longrightarrow$ a set of arrays or a matrix
  - i^{th} vector of $\boldsymbol{C}$ is $\boldsymbol{c}_i$
  - j^{th} element of the i^{th} vector of $\boldsymbol{C}$ is the scalar $c_{i,j}$

Linear Algebra – Transpose

- $\boldsymbol{a}$ is in $1 \times d$ dimensions $\rightarrow$ $\boldsymbol{a}^{\mathrm{T}}$ is in $d \times 1$ dimensions

- $\boldsymbol{A}$ is in $e \times d$ dimensions $\rightarrow$ $\boldsymbol{A}^{\mathrm{T}}$ is in $d \times e$ dimensions

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

Linear Algebra – Dot product

- $\boldsymbol{a} \cdot \boldsymbol{b}^T = c$
  - dimensions: $1 \times d \cdot d \times 1 = 1$

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix} = 5$$

- $\boldsymbol{a} \cdot \boldsymbol{B} = \boldsymbol{c}$
  - dimensions: $1 \times d \cdot d \times e = 1 \times e$

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 0 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 5 & 2 \end{bmatrix}$$

- $\boldsymbol{A} \cdot \boldsymbol{B} = \boldsymbol{C}$
  - dimensions: $l \times m \cdot m \times n = l \times n$

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & 1 \\ 0 & 0 & 5 \\ 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 0 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 5 & 2 \\ 3 & 2 \\ 5 & -5 \\ 8 & 13 \end{bmatrix}$$

Agenda

- **Machine Learning – a quick tour**
- Sentiment Analysis with Bag of Words
- Low-dimensional representations with SVD

Statistical Learning

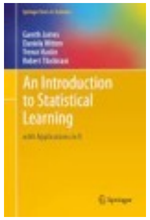
Problem definition

- N observed data points, where each data point $\boldsymbol{x}_i$ is accompanied with a label y_i

$$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)\}$$

- Each data point is a vector with L dimensions (features):

$$\boldsymbol{x}_i = \left[x_{i,1}, x_{i,2}, \dots, x_{i,L}\right]$$



Statistical Learning

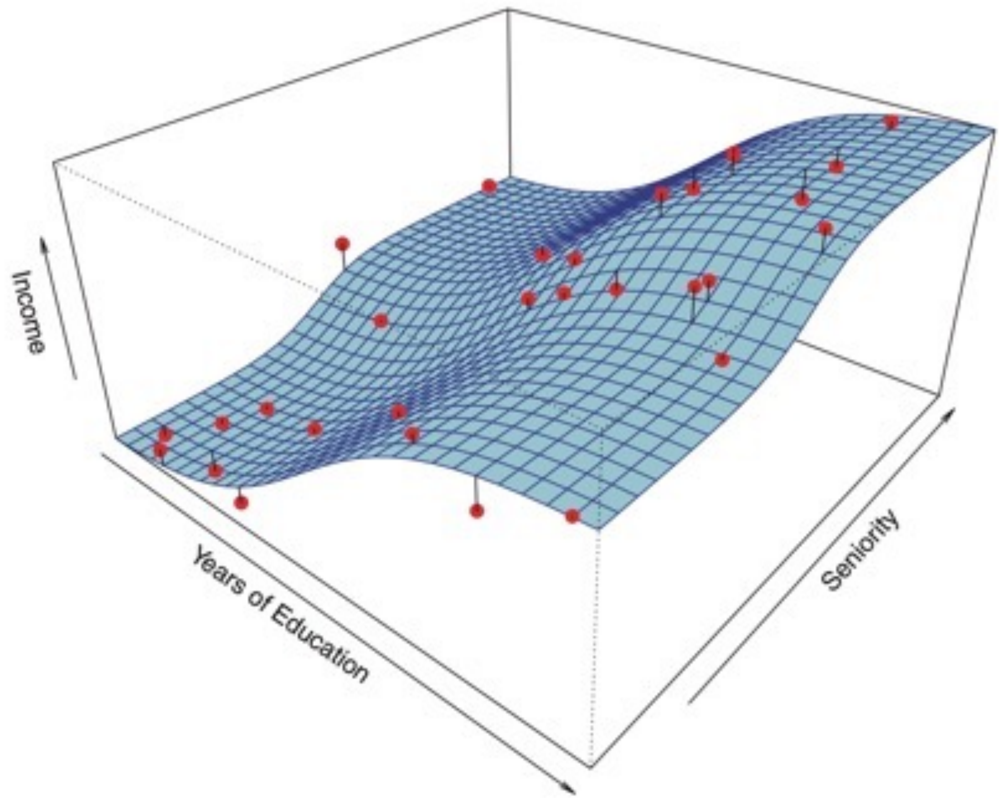
Assumption:

- The data is generated by a **TRUE but unknown function** (f^{TRUE}) such that:

$$y_i = f^{TRUE}(\boldsymbol{x}_i) + \epsilon_i$$

- $\epsilon_i > 0$
  - Called **irreducible error**
  - The error is caused by the constrains in gathering data, and measuring features
  - It means that in data-oriented approaches, there always exists some error that can't be reduced

Example f^{TRUE}



f^{TRUE} → blue surface, the true but unknown function

$\boldsymbol{x}_i$→ each red point with two features: ***Seniority*** & ***Years of Education***

y_i→ **Income** for each data point

ϵ_i→ the distance between the measured **income** of each data point and the surface

Machine Learning Model

- In machine learning, we aim to estimate f^{TRUE} by solely looking at data points
- A machine learning model defines function f:

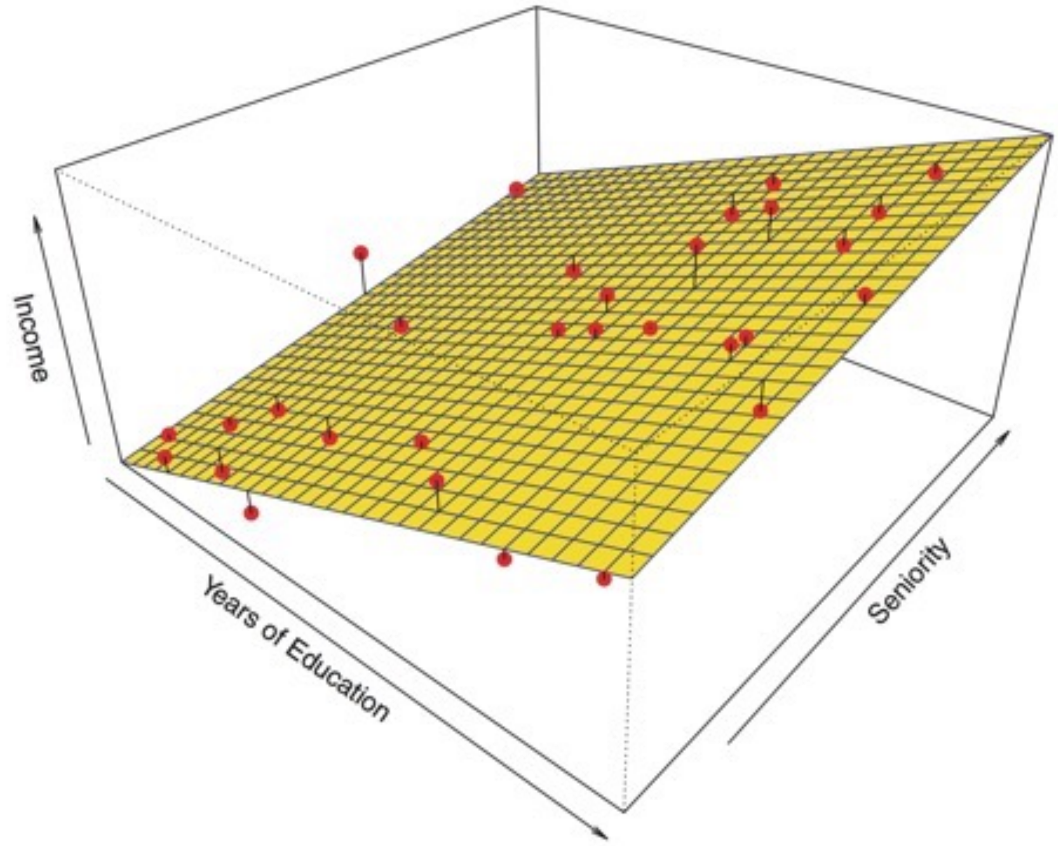
$$\widehat{\boldsymbol{y}} = f(\boldsymbol{X})$$

such that $\widehat{\boldsymbol{y}}$ (predicted outputs) be close to $\boldsymbol{y}$ (real outputs).

- The difference between $\widehat{\boldsymbol{y}}$ and $\boldsymbol{y}$ is **reducible error**
  - Can be reduced by better models

- In statistical/machine/deep learning, we approach reducible error

A machine learning model

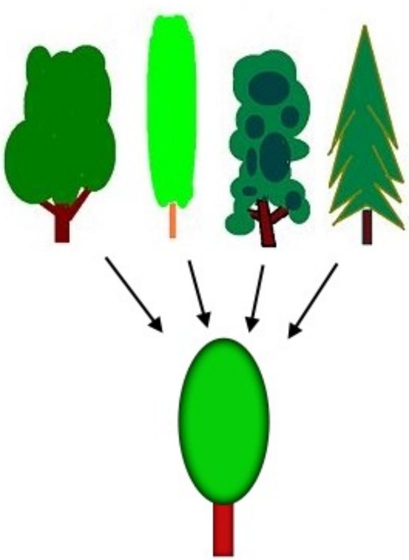


f → yellow surface, machine learning model (here a linear regression)

Prediction error → the distance between the data points and the surface

Generalization

- The aim of machine learning is to create a model using **observed experiences** (training data) that generalizes to the problem domain, namely **performs** well on **unobserved instances** (test data)

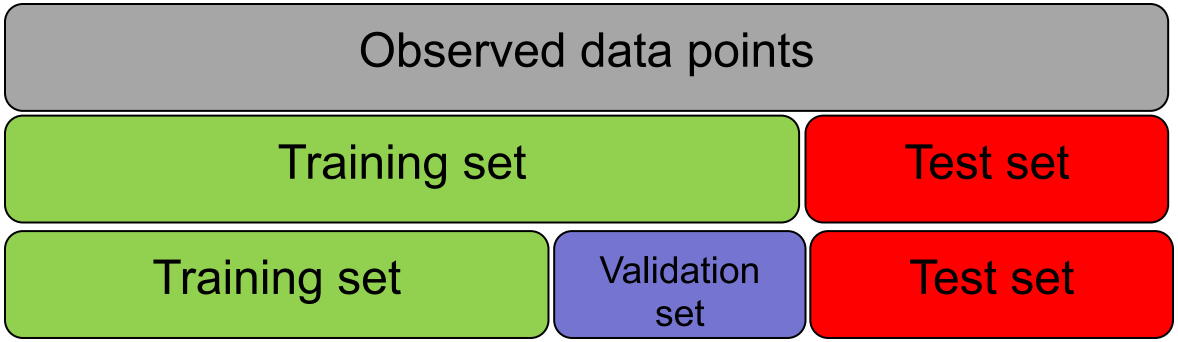


link

Learning the model – Splitting dataset

- To conduct machine learning experiments, we split the data into:
  - **Training set**: for training the model
  - **Validation set**: for tuning model's hyper-parameters
  - **Test set**: for evaluating model's performance

- Common train – validation – test splitting sizes
  - 60%, 20%, 20%
  - 70%, 15%, 15%
  - 80%, 10%, 10%



Learning the model

Features / Variables (X)

Labels / Output Variable (Y)

Dataset

sex	age	Pstatus	romantic	Walc
F	18	A	no	1
F	17	T	no	1
F	15	T	no	3
F	15	T	yes	1
F	16	T	no	2
M	16	T	no	2
M	16	T	no	1
F	17	A	no	1
M	15	A	no	1
M	15	T	no	1
F	15	T	no	2
F	15	T	no	1
M	15	T	no	3
M	15	T	no	2
M	15	A	yes	1
F	16	T	no	2
F	16	T	no	2
F	16	T	no	1
M	17	T	no	4

Pstatus: parent's cohabitation status ('T' - living together 'A' - apart)
Romantic: with a romantic relationship
Walc: weekend alcohol consumption (from 1 - very low to 5 - very high)

http://archive.ics.uci.edu/ml/datasets/STUDENT+ALCOHOL+CONSUMPTION#

Learning the model

Train Set

sex	age	Pstatus	romantic	Walc
F	18	A	no	1
F	17	T	no	1
F	15	T	no	3
F	15	T	yes	1
F	16	T	no	2
M	16	T	no	2
M	16	T	no	1
F	17	A	no	1
M	15	A	no	1
M	15	T	no	1
F	15	T	no	2
F	15	T	no	1
M	15	T	no	3

Test Set

M	15	T	no	2
M	15	A	yes	1
F	16	T	no	2
F	16	T	no	2
F	16	T	no	1
M	17	T	no	4

Learning the model

Train Set

sex	age	Pstatus	romantic	Walc
F	18	A	no	1
F	17	T	no	1
F	15	T	no	3
F	15	T	yes	1
F	16	T	no	2
M	16	T	no	2
M	16	T	no	1
F	17	A	no	1
M	15	A	no	1
M	15	T	no	1
F	15	T	no	2
F	15	T	no	1
M	15	T	no	3

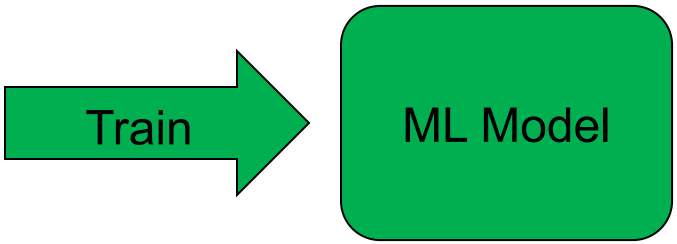
Test Set

					y
M	15	T	no	?	2
M	15	A	yes	?	1
F	16	T	no	?	2
F	16	T	no	?	2
F	16	T	no	?	1
M	17	T	no	?	4

Learning the model

Train Set

sex	age	Pstatus	romantic	Walc
F	18	A	no	1
F	17	T	no	1
F	15	T	no	3
F	15	T	yes	1
F	16	T	no	2
M	16	T	no	2
M	16	T	no	1
F	17	A	no	1
M	15	A	no	1
M	15	T	no	1
F	15	T	no	2
F	15	T	no	1
M	15	T	no	3



Test Set

M	15	T	no	?
M	15	A	yes	?
F	16	T	no	?
F	16	T	no	?
F	16	T	no	?
M	17	T	no	?

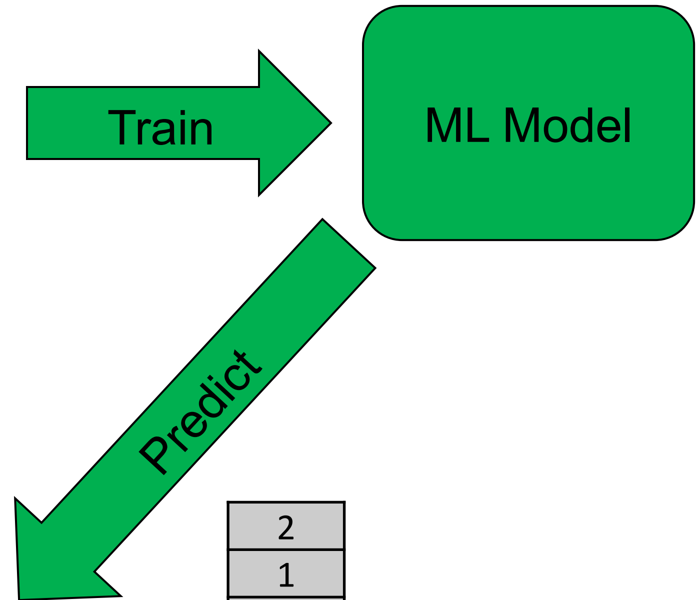
2
1
2
2
1
4

y

Learning the model

Train Set

sex	age	Pstatus	romantic	Walc
F	18	A	no	1
F	17	T	no	1
F	15	T	no	3
F	15	T	yes	1
F	16	T	no	2
M	16	T	no	2
M	16	T	no	1
F	17	A	no	1
M	15	A	no	1
M	15	T	no	1
F	15	T	no	2
F	15	T	no	1
M	15	T	no	3



Test Set

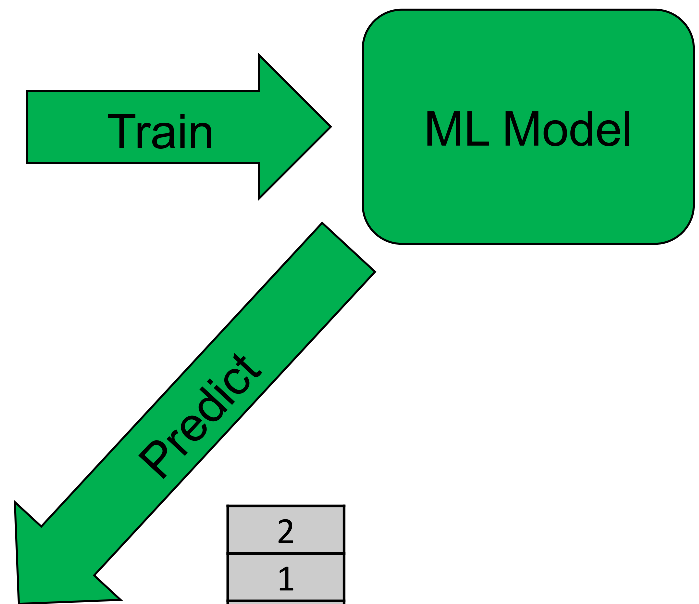
				$\hat{y}$
M	15	T	no	1
M	15	A	yes	1
F	16	T	no	2
F	16	T	no	2
F	16	T	no	3
M	17	T	no	4

y
2
1
2
2
1
4

Learning the model

Train Set

sex	age	Pstatus	romantic	Walc
F	18	A	no	1
F	17	T	no	1
F	15	T	no	3
F	15	T	yes	1
F	16	T	no	2
M	16	T	no	2
M	16	T	no	1
F	17	A	no	1
M	15	A	no	1
M	15	T	no	1
F	15	T	no	2
F	15	T	no	1
M	15	T	no	3



Test Set

				$\hat{y}$
M	15	T	no	1
M	15	A	yes	1
F	16	T	no	2
F	16	T	no	2
F	16	T	no	3
M	17	T	no	4

y
2
1
2
2
1
4

Evaluation → Prediction error

Elements of machine learning

- Various models:
  - Linear Regression / Logistic Regression
  - Support Vector Machines (SVM)
  - Decision Tree / Random Forest
  - Neural Networks (Multi-layer Perceptron)
  - Etc.

- Model parameters
  - Each model has a set of variables which define the output of the model

- Model hyperparameters
  - A set of specifications of each model, set before training
    - E.g., the kernel shape of SVM, or regularization weight in logistic regression

A sample ML model: Linear Regression

- In Linear Regression, f is defined as:

$$y = f(\boldsymbol{x}; \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_l x_l$$

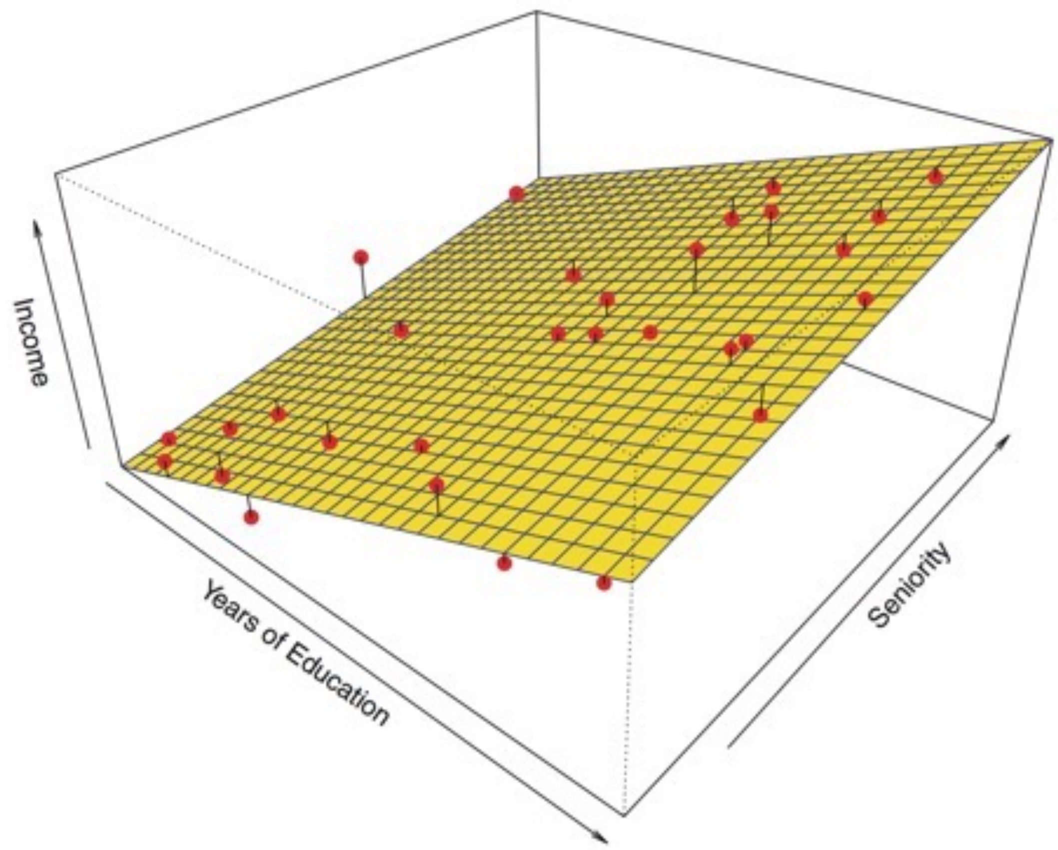
where $\boldsymbol{w} = [w_0, w_1, \ldots, w_l]$ are model parameters

- In the "income" example:

$$income = f(\boldsymbol{x}; \boldsymbol{w}) = w_0 + w_1 \times education + w_2 \times seniority$$

Model parameters are shown in red

A sample ML model: Linear Regression



$$income = f(\boldsymbol{x}; \boldsymbol{w}) = w_0 + w_1 \times education + w_2 \times seniority$$

Which ML model?

Model Capacity

low

high

less flexible
less parameters
lower variance
higher bias
prone to underfitting

more flexible
more parameters
higher variance
lower bias
prone to overfitting

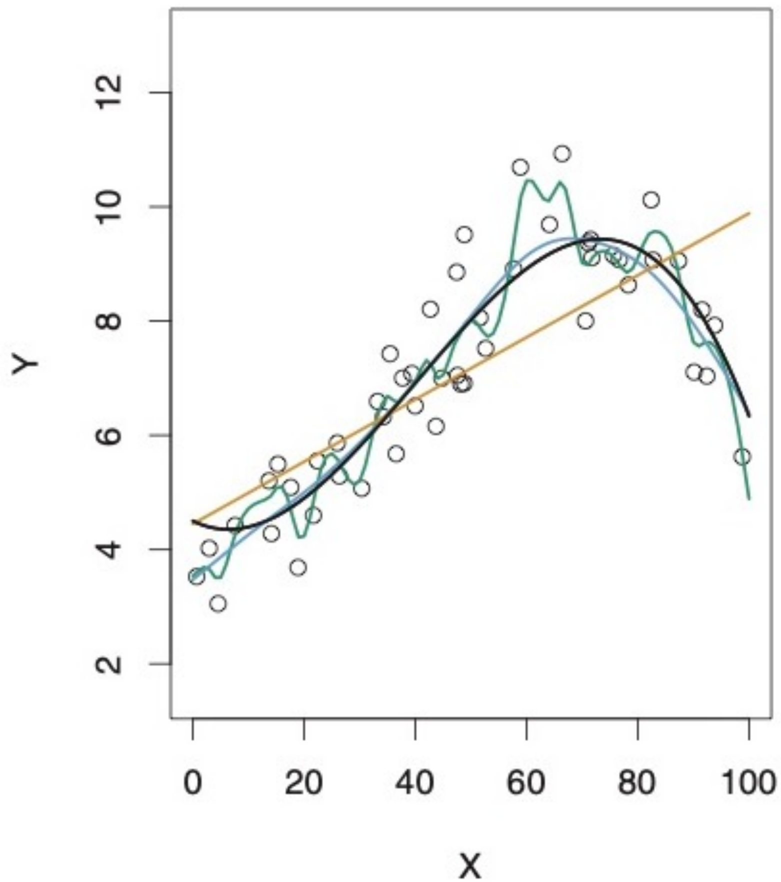
Terms of the day!

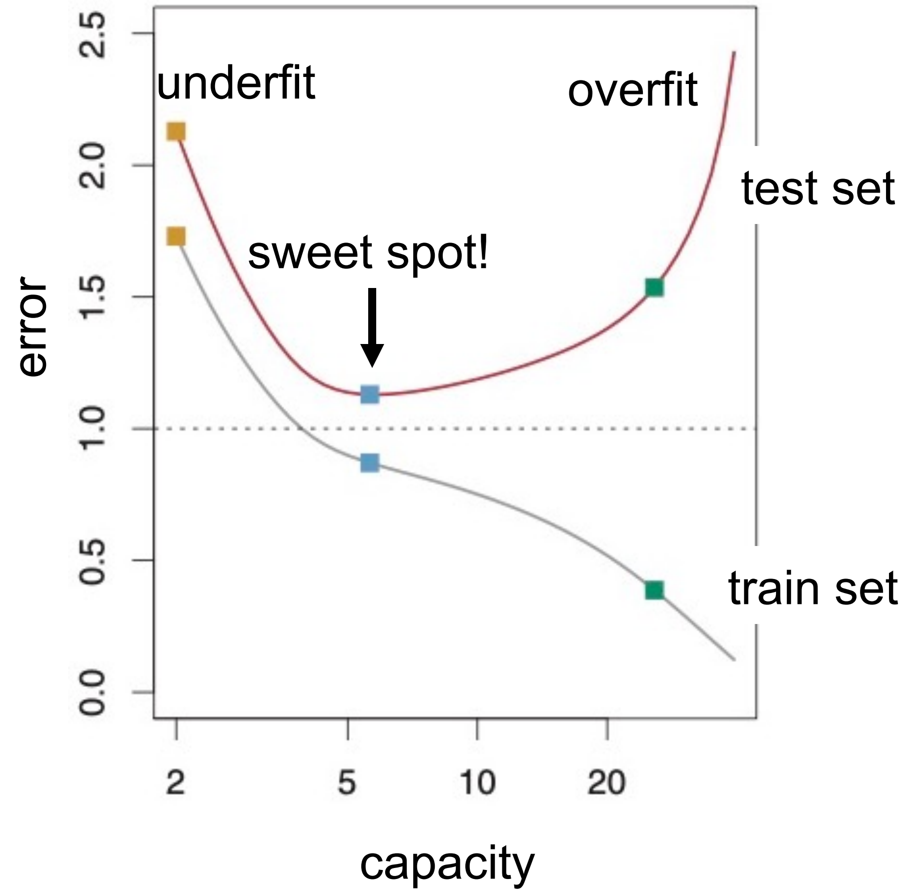
(Statistical) Bias indicates the amount of assumptions, taken to define a model. Higher bias means more assumptions and less flexibility, as in linear regression.

Variance: in what extent the estimated parameters of a model vary when the values of data points change (are resampled).

Overfitting: When the model exactly fits to training data, namely when it also captures the noise in data.

Which ML model?





<u>Models:</u>
black → f^{TRUE}
orange → linear regression (higher bias, less variance)
blue and green→ two smoothing spline models (less bias, higher variance)

Elements of machine learning – cont.

- Evaluation
  - Measuring model's performance by comparing predictions with actual labels

Classification

- **Accuracy** $\frac{\#\ of\ correct\ predictions}{\#\ of\ samples}$
- Precision $\frac{TP}{TP+FP}$
- Recall $\frac{TP}{TP+FN}$
- F-measure $\frac{2 * precision * recall}{precision + recall}$

Regression

- MSE
- R-squared

Elements of machine learning – cont.

- Loss function
  - A function that measures the discrepancies between the predicted outputs $\hat{\boldsymbol{y}}$ and the actual labels $\boldsymbol{y}$
  - The model uses loss function to find an *optimum* set of parameters that reduce the loss

- Regularization
  - A regularization method introduces additional information (assumptions) to **avoid overfitting** by **decreasing variance**

Model selection & Hyper parameter tunning

- Among various hyperparameter values, which model should we use?

Model selection:

- Decide on several sets of hyper-parameters
  - E.g., various kernel shapes of SVM, or a range of values for the regularization weight in logistic regression
- For each set of hyper-parameters, train a model using training set
- Evaluate all the trained models on the validation set, and select the best performing one
- Evaluate the selected model on test set → task performance

Agenda

- Machine Learning – a quick tour
- **Sentiment Analysis with Bag of Words**
- Low-dimensional representations with SVD

Sentiment Analysis

HP Officejet 6500A Plus e-All-in-One Color Ink-jet - Fax / copier / printer / scanner
$89 online, **$100** nearby ★★★★☆ 377 reviews
September 2010 - Printer - HP - Inkjet - Office - Copier - Color - Scanner - Fax - 250 sh

Reviews

Summary - Based on 377 reviews

1 star	2	3	4 stars	5 stars

What people are saying

ease of use	"This was very easy to setup to four computers."
value	"Appreciate good quality at a fair price."
setup	"Overall pretty easy setup."
customer service	"I DO like honest tech support people."
size	"Pretty Paper weight."
mode	"Photos were fair on the high quality mode."
colors	"Full color prints came out with great quality."

A *tough* Example!

"*This past Saturday, I bought a <u>Nokia</u> phone and my girlfriend bought a <u>Motorola</u> phone with Bluetooth. We called each other when we got home. The voice on my phone was clear, better than my previous Samsung phone. The battery life was however short. My girlfriend was quite happy with her phone. I wanted a phone with good sound quality. So my purchase was a real disappointment. I returned the phone yesterday.*"

[1]

Document-level Sentiment Analysis

- Text- or document-level sentiment analysis assumes that whole the text expresses one sentiment about one opinion target
  - Not like the previous example!

Approaches

- Unsupervised methods: use a lexicon of sentiment words to output a sentiment score

- Supervised methods: learn to predict sentiment scores

Unsupervised Sentiment Analysis

- Some sentiment lexicons:
  - SentiWordNet
    - Each term has a positive and a negative score
  - Financial sentiment dictionary
    - Groups of negative/positive/uncertain words
  - …

Terms	Neg_score	Pos_score
able	0	0.125
unable	0.75	0
emerging	0	0

Group	Sample terms
Negative	discontinued, penalties, misconduct
Positive	achieve, efficient, profitable
Uncertainty	approximate, fluctuate, uncertain, variability

E.g., one way using SentiWordNet would be:

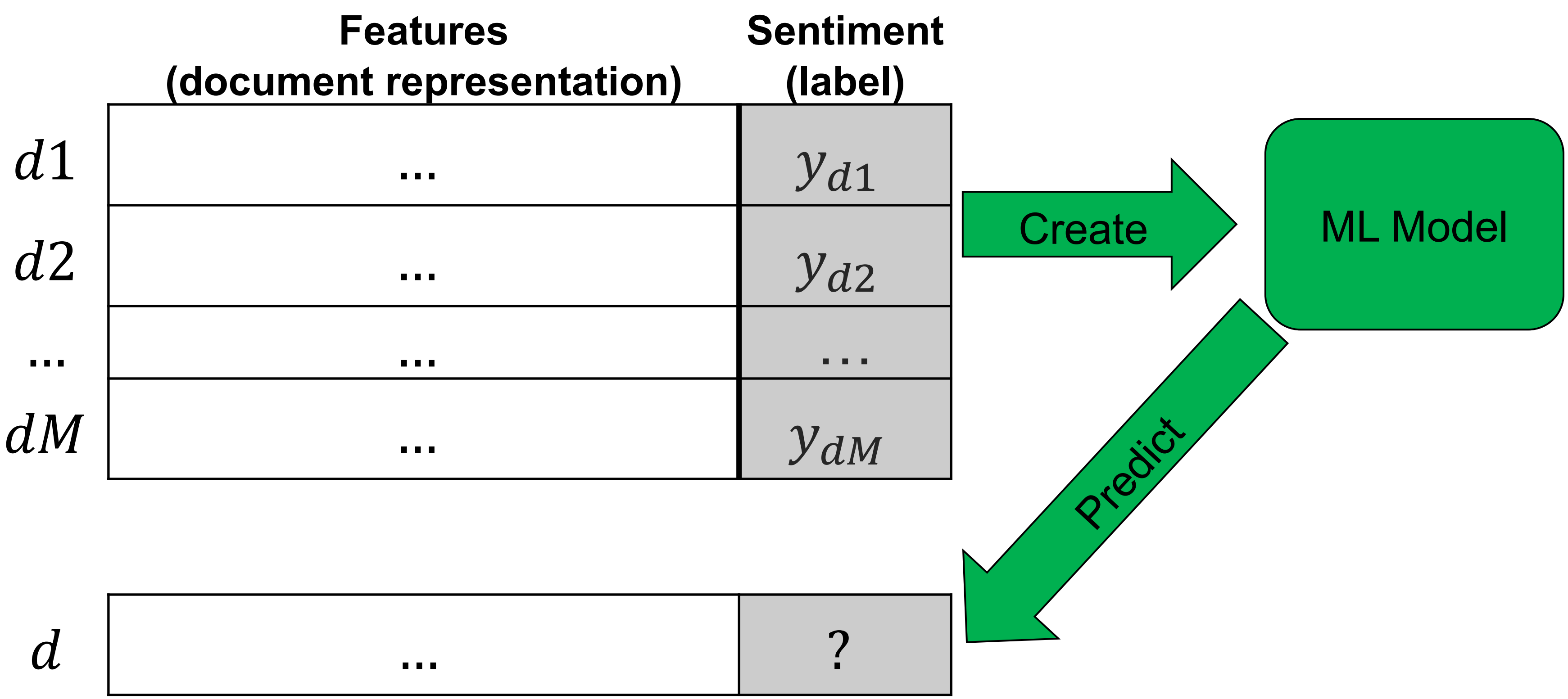
- The positive sentiment of the document d is the sum of the positive scores of the terms in the lexicon (V^{lex}), which appear in d:

$$pos\_sentiment(d) = \sum_{v \in \{V^{lex} \cap d\}} pos\_score(v)$$

Some resources
SentiWordNet: https://github.com/aesuli/SentiWordNet
Bing Liu's opinion lexicon: https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
Loughran-McDonold financial sentiment dictionary: https://sraf.nd.edu/textual-analysis/resources/

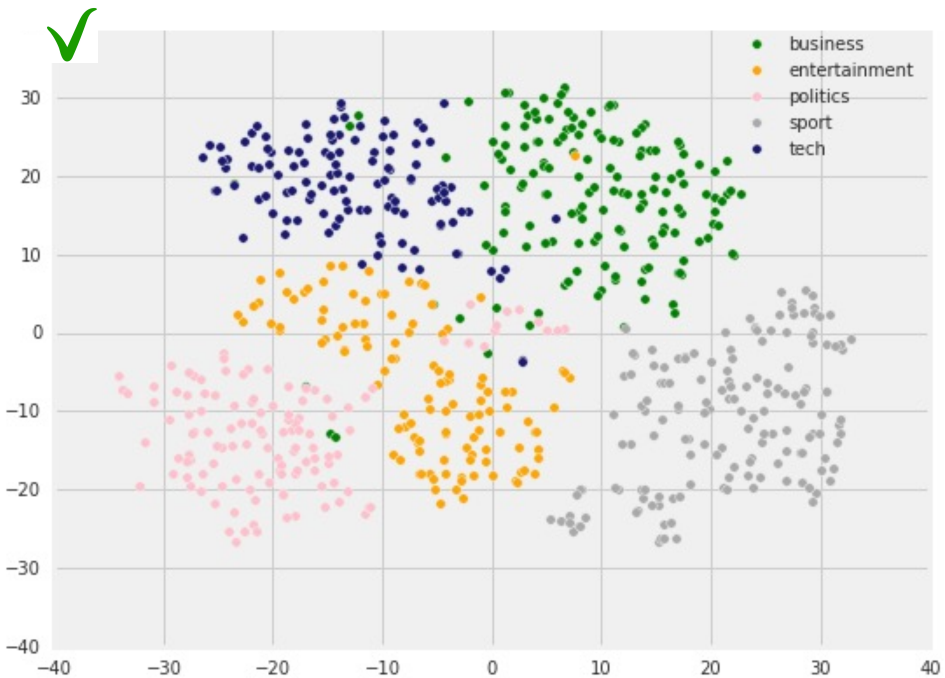
Supervised Sentiment Analysis

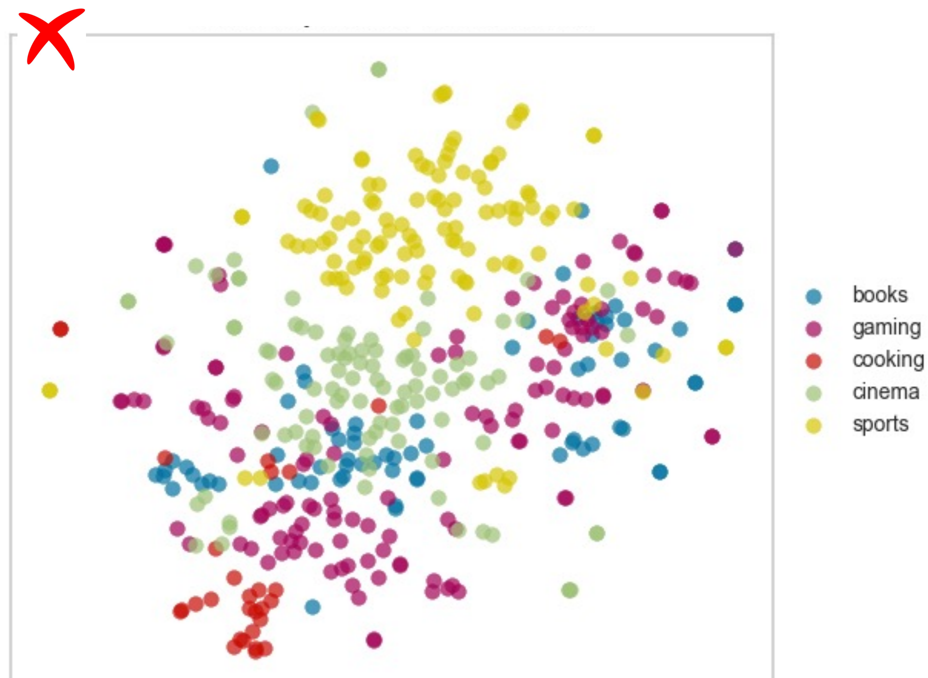
- A dataset consist of M documents and their assigned sentiments
- Possible sentiment values:
  - [-1, 0, 1] → [negative, neutral, positive] (classification problem)
  - Real-valued numbers e.g. stock price (regression problem)



Document representation is central!

- The key to effective classification is in *good* document representations!
  - Feature extraction in classical ML
  - Representation learning in deep learning
- If representations are *well-separated* and *well-generalized*, any ML model with enough capacity can effectively classify the test data.



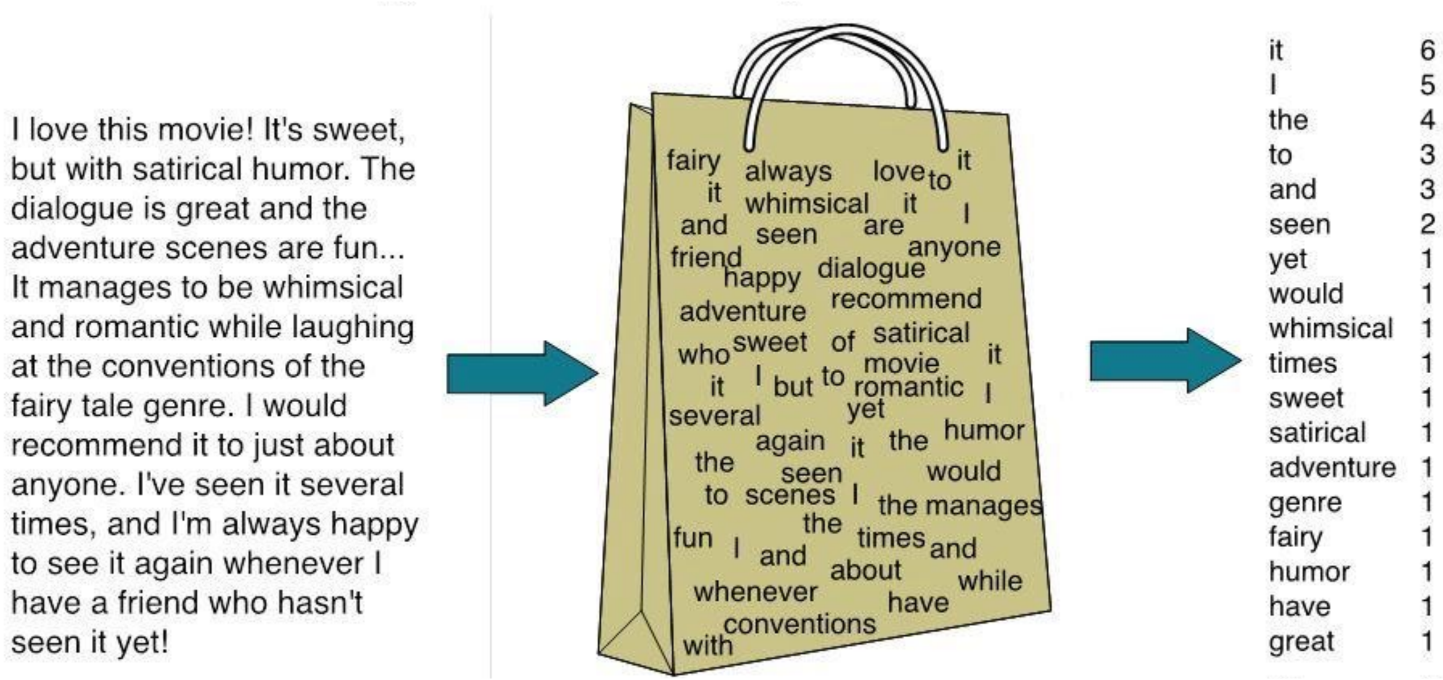


Two sample document representation sets projected to two-dimensional spaces

source source

Towards creating a document representation
Bag-of-words (BoW) Approach

- In BoW, the order and position of words is ignored and only the number occurrences of words are considered



- What are the possible limitations of BoW approaches?

Source: https://www.programmersought.com/article/4304366575/

Towards BoW document representation

Creating Dictionary

- We first create a dictionary $\mathbb{V}$, consisting of N words (terms):

$$\mathbb{V} = \{v1, v2, \dots, vN\}$$

- Dictionary should be created only using training data
- Dictionary need to be preprocessed, e.g., by:
  - keeping only top-*N* most frequent words
  - removing any word with a lower frequency than a threshold
- The tokens that do not appear in the dictionary are called Out-Of-Vocabulary (OOV)
- OOVs also need to be separately handled, e.g., by:
  - replacing them with a special token <oov>, and adding <oov> to the dictionary
  - Ignoring them completely in the processing (removing them from the text)

Document-Word matrix

- Featurization is done in a document-word matrix
- Document are data points (rows)
- Words in the dictionary are features (dimensions, columns)
- $x_{v,d}$ is the feature value of word v in document d
- Each value $x_{v,d}$ is calculated using a word (term) weighting model

	$v1$	$v2$	...	vN	**sentiment (label)**
$d1$	$x_{v1,d1}$	$x_{v2,d1}$	...	$x_{vN,d1}$	y_{d1}
$d2$	$x_{v1,d2}$	$x_{v2,d2}$	...	$x_{vN,d2}$	y_{d2}
...	...	...	...	...	$\dots$
dM	$x_{v1,dM}$	$x_{v2,dM}$	...	$x_{vN,dM}$	y_{dM}

Weighting models
Term count & term frequency

(1) Term (word) count:

- One common word weighting approach is to simply count the number of occurrences of a word in a document,
  - Example: number of times ***JKU*** appears in each news document.

$$\mathrm{tc}_{v,d} = \#\ \text{of occurrences of word}\ v\ \text{in}\ d$$

(2) Term (word) frequency:

- The Importance of a word (probably) does not one-to-one increase with the number of occurrences
- Based on experimental results, logarithm is commonly used to dampen raw counts, resulting in …

$$\mathrm{tf}_{v,d} = \log(1 + \mathrm{tc}_{v,d})$$

On informativeness of less frequent words

- Words that do not appear often, usually carry more information in comparison with highly frequent ones
  - E.g., ***JKU*** in a large news should be more informative than the word ***university*** that appears more often.

- Inverse document frequency (idf) measures the importance of the words according to whole the collection of documents:

$$\mathrm{idf}_v = \log\left(\frac{M}{\mathrm{df}_v + 1}\right)$$

  - M is the number of documents in the collection
  - df_v (document frequency of v) is the number of documents that contain word v

- Higher idf_v means that the word appears less often in the collection, and is therefore more informative (more important)
  - ***JKU*** has a higher idf than ***university***, and ***the*** gets a very low idf.

Weighting models
Term frequency-Inverse document frequency

- **(3) $\mathbf{tf{-}idf}$** weighting model is the product of $\mathrm{tf}_{v,d}$ to idf_v

$$x_{v,d} = \mathrm{tf{-}idf}_{v,d} = \log\left(1 + \mathrm{tc}_{v,d}\right) \times \log\left(\frac{M}{\mathrm{df}_v + 1}\right)$$

tf increases with the number of occurrences <u>within the document</u>

idf increases with the rarity of the word in <u>whole the collection of documents</u>

- A well-known word weighting method!

Putting all together!

- Use a word weighting to prepare document-word matrix

	$v1$	$v2$	...	vN	**sentiment (label)**
$d1$	$x_{v1,d1}$	$x_{v2,d1}$	...	$x_{vN,d1}$	y_{d1}
$d2$	$x_{v1,d2}$	$x_{v2,d2}$	...	$x_{vN,d2}$	y_{d2}
...	...	...	...	...	...
dM	$x_{v1,dM}$	$x_{v2,dM}$	...	$x_{vN,dM}$	y_{dM}

- The rest is applying a standard machine learning pipeline:
  - Use the training set to train models with different hyper-parameters
  - Evaluate the models on the validation set and select the best model
  - Report the test set performance of the best model

Agenda

- Machine Learning – a quick tour
- Sentiment Analysis with Bag of Words
- **Low-dimensional representations with SVD**

Sentiment Analysis with BoW

	$v1$	$v2$	…	vN	**sentiment (label)**
$d1$	$x_{v1,d1}$	$x_{v2,d1}$	…	$x_{vN,d1}$	y_{d1}
$d2$	$x_{v1,d2}$	$x_{v2,d2}$	…	$x_{vN,d2}$	y_{d2}
…	…	…	…	…	…
dM	$x_{v1,dM}$	$x_{v2,dM}$	…	$x_{vN,dM}$	y_{dM}

# of data points:

$$M \sim [10K - 100K]$$

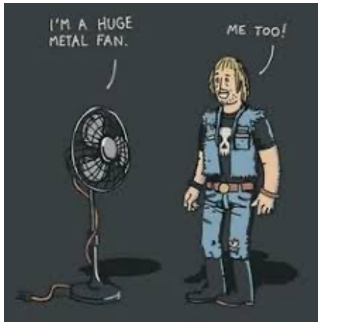
# of features (dimensions):

$$N \sim [20K - 500K]$$

- BoW document representations are
  - sparse (a lot zeros) and …
  - typically in a very high dimension

Why low-dimensional vectors?

- Easier to store and load
- More efficient when used as features in ML models
- Better generalization due to the reduction of noise in data
- Able to capture higher-order relations:
  - Synonyms like *car* and *automobile* might be merged into the same dimensions



  - Polysemies like *bank (financial institution)* and *bank (bank of river)* might be separated into different dimensions

Polysemy is the capacity for a word or phrase to have multiple related meanings.

How to reduce features (dimensions)?

- Feature selection
  - Keep some important features and get rid of the rest!
- Dimensionality reduction
  - Project data from high to a low dimensional space

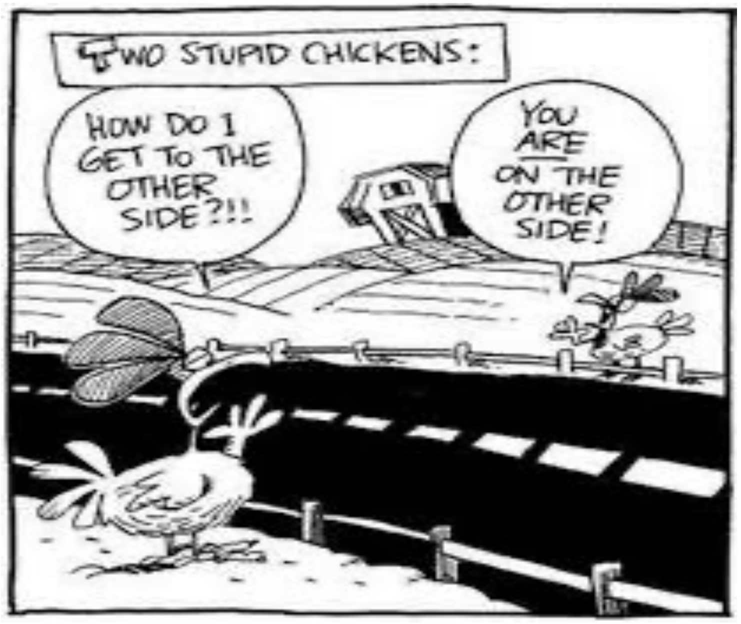
Feature selection

- During preprocessing
  - Remove stop words or very common words
    - tf–idf do it in a "soft" way, *why?*
  - Remove very rare words
    - Usually done as dictionary is created
  - Stemming & lemmatization
- Explicitly specifying features
  - E.g., by limiting the dictionary (and therefore features) to only the words of a domain-specific lexicon
- Post-processing
  - Keep important features using some informativeness measures
  - Subset selection

Dimensionality reduction with LSA

- Latent Semantic Analysis (LSA)
  - A common method to create semantic vectors
  - Based on Singular Value Decomposition (SVD)

Semantics matters!



Singular Value Decomposition

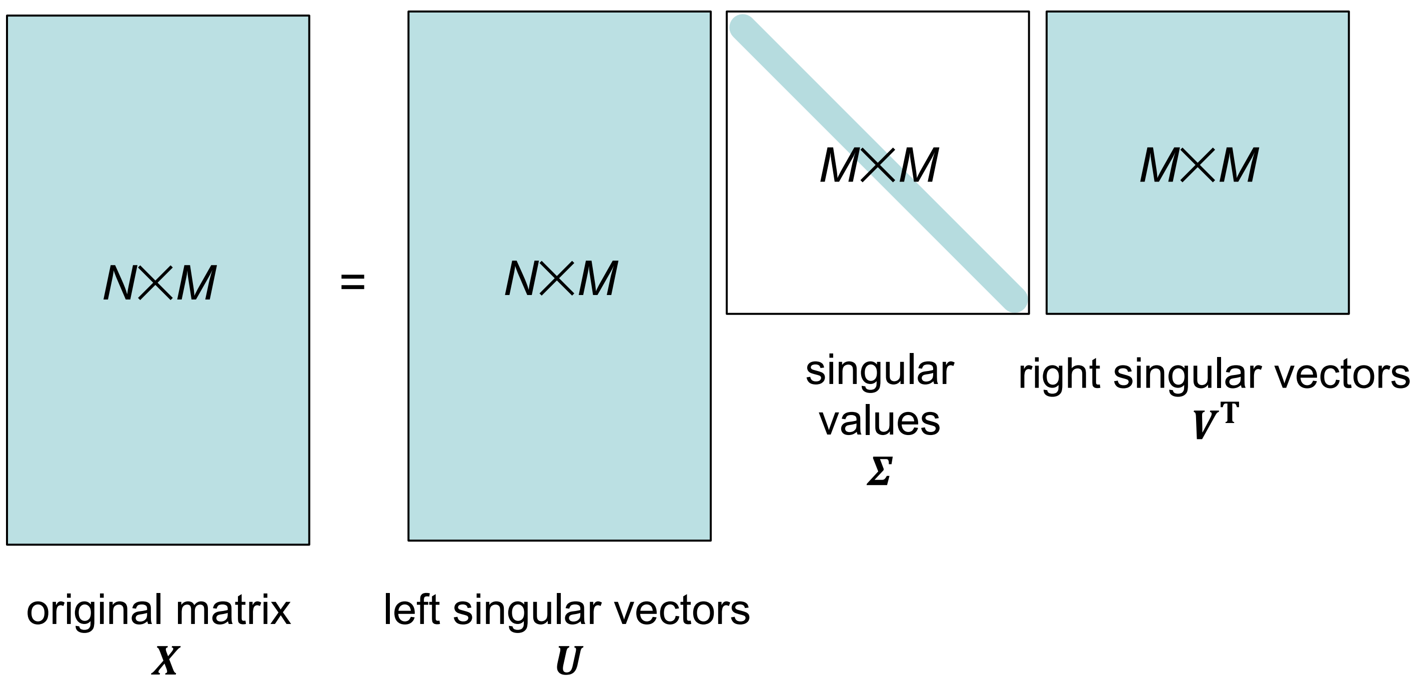
- An $N \times M$ matrix $\boldsymbol{X}$ can be factorized to three matrices:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$$

- $\boldsymbol{U}$ left singular vectors is an $N \times M$ unitary matrix
- $\boldsymbol{\Sigma}$ is an $M \times M$ diagonal matrix, diagonal entries
  - are singular values,
  - show the importance of corresponding M dimensions in $\boldsymbol{X}$
  - are all positive and sorted from large to small values
- $\boldsymbol{V}^{\mathrm{T}}$ right singular vectors is an $M \times M$ unitary matrix

* The definition of SVD is simplified. Refer to https://en.wikipedia.org/wiki/Singular_value_decomposition for the exact definition

Singular Value Decomposition



Latent Semantic Analysis – training

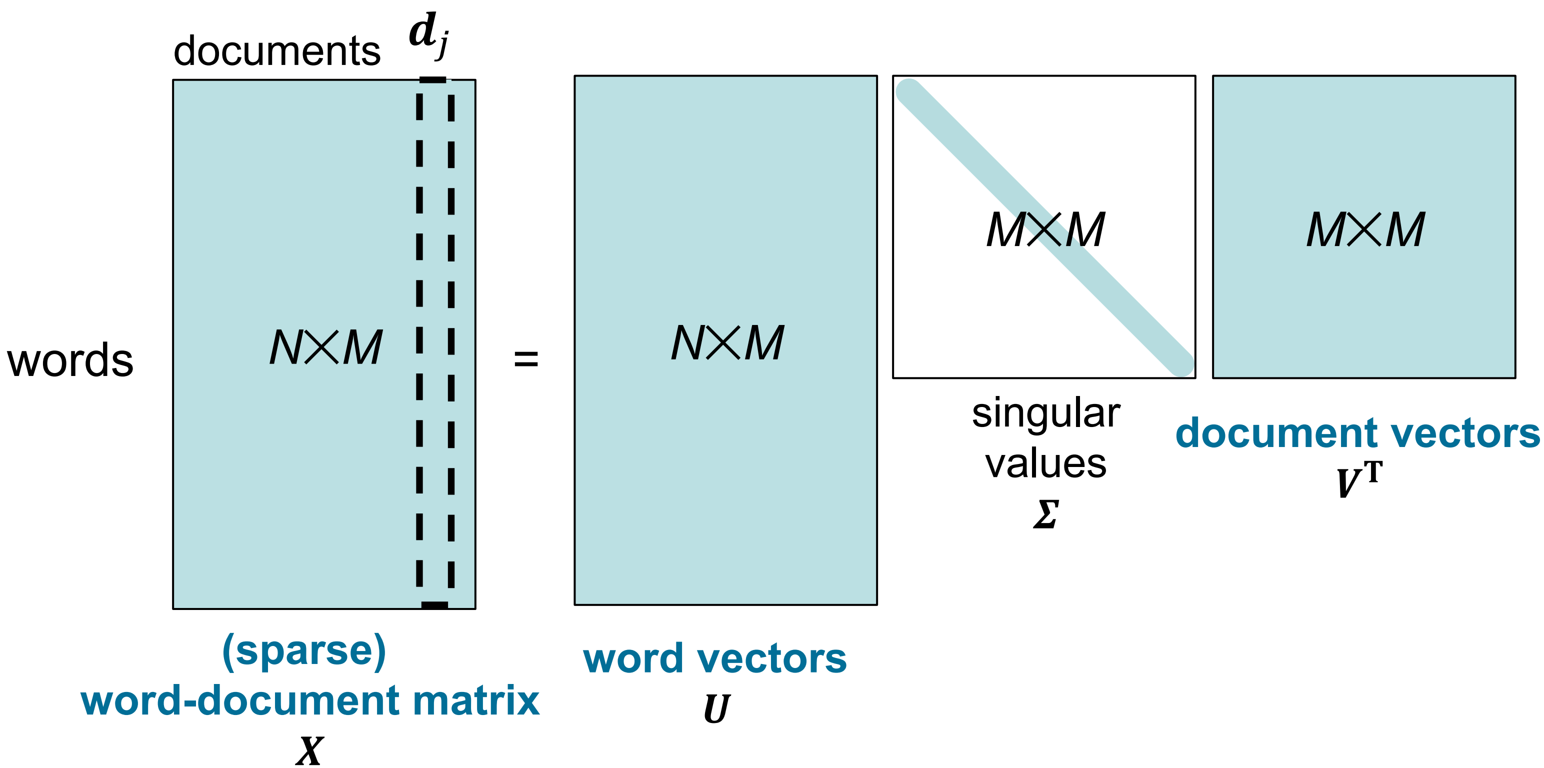
Training Time

Step 1

- Prepare the **word-document** matrix using training data

☞ Note that here we are using word-document matrix and **not** the document-word matrix (as we talked before)! While it is also technically possible to start with the document-word matrix, we follow here the common definition of LSA.

- Apply SVD to the matrix



Latent Semantic Analysis – training

Training Time

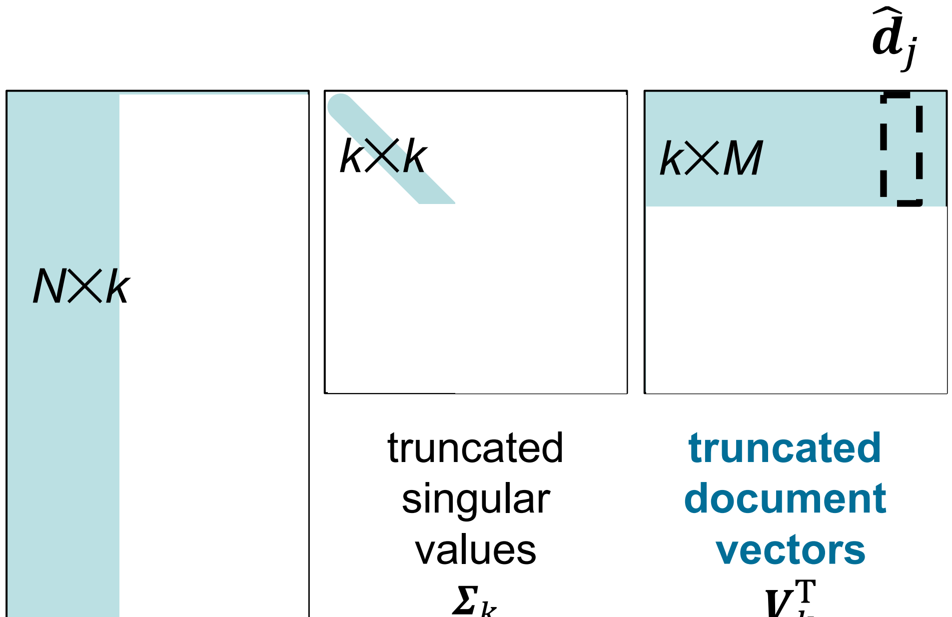
Step 2

- Keep only top k singular values in $\boldsymbol{\Sigma}$ and set the rest to zero, called $\boldsymbol{\Sigma}_k$
- Truncate the $\boldsymbol{U}$ and $\boldsymbol{V}^{\mathrm{T}}$ matrices, resulting in $\boldsymbol{U}_k$ and $\boldsymbol{V}_k^{\mathrm{T}}$

- Columns in $\boldsymbol{V}_k^{\mathrm{T}}$ are the new low-dimensional document representations
  - Vectors of $\boldsymbol{V}_k^{\mathrm{T}}$ are used to train ML models

Latent Semantic Analysis – training

Training Time



truncated word vectors
$\boldsymbol{U}_k$

truncated singular values
$\boldsymbol{\Sigma}_k$

truncated document vectors
$\boldsymbol{V}_k^{\mathrm{T}}$

- $\boldsymbol{V}_k^{\mathrm{T}}$ is the matrix of dense low-dimensional document vectors
  - Used for training the ML models

Latent Semantic Analysis – example

Training Time

- A word-document matrix with 3 documents and 7 words
- Apply SVD to the matrix

$$\boldsymbol{X} = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 5 \\ 4 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} 0.04 & 0.35 & -0.91 \\ 0.21 & 0.16 & 0.20 \\ 0.94 & -0.17 & -0.04 \\ 0.12 & 0.88 & 0.27 \\ 0.18 & -0.03 & 0.00 \\ 0.02 & 0.20 & 0.20 \end{bmatrix}}_{\boldsymbol{U}} \cdot \underbrace{\begin{bmatrix} 5.21 & 0 & 0 \\ 0 & 4.59 & 0 \\ 0 & 0 & 1.66 \end{bmatrix}}_{\boldsymbol{\Sigma}} \cdot \underbrace{\begin{bmatrix} 0.15 & 0.04 & 0.98 \\ -0.92 & -0.34 & 0.15 \\ -0.34 & 0.93 & 0.01 \end{bmatrix}}_{\boldsymbol{V}^{\mathrm{T}}}$$

Latent Semantic Analysis – example

Training Time

$$\begin{bmatrix} 0.04 & 0.35 & -0.91 \\ 0.21 & 0.16 & 0.20 \\ 0.94 & -0.17 & -0.04 \\ 0.12 & 0.88 & 0.27 \\ 0.18 & -0.03 & 0.00 \\ 0.02 & 0.20 & 0.20 \end{bmatrix} \cdot \begin{bmatrix} 5.21 & 0 & 0 \\ 0 & 4.59 & 0 \\ 0 & 0 & 1.66 \end{bmatrix} \cdot \begin{bmatrix} 0.15 & 0.04 & 0.98 \\ -0.92 & -0.34 & 0.15 \\ -0.34 & 0.93 & 0.01 \end{bmatrix}$$

$$\boldsymbol{U} \qquad \boldsymbol{\Sigma} \qquad \boldsymbol{V}^{\mathrm{T}}$$

- Keep only the top $k = 2$ singular values:

New k -dimensional document representations

$$\begin{bmatrix} 0.04 & 0.35 \\ 0.21 & 0.16 \\ 0.94 & -0.17 \\ 0.12 & 0.88 \\ 0.18 & -0.03 \\ 0.02 & 0.20 \end{bmatrix} \cdot \begin{bmatrix} 5.21 & 0 \\ 0 & 4.59 \end{bmatrix} \cdot \begin{array}{c} \begin{array}{ccc} \hat{\boldsymbol{d}}_1 & \hat{\boldsymbol{d}}_2 & \hat{\boldsymbol{d}}_3 \\ \downarrow & \downarrow & \downarrow \end{array} \\ \begin{bmatrix} 0.15 & 0.04 & 0.98 \\ -0.92 & -0.34 & 0.15 \end{bmatrix} \end{array}$$

$$\boldsymbol{U}_k \qquad \boldsymbol{\Sigma}_k \qquad \boldsymbol{V}_k^{\mathrm{T}}$$

Latent Semantic Analysis – inference

Inference Time (Validation/Test)

- Given a high-dimensional document vector $\boldsymbol{d}^*$ in $N \times 1$ dimensions, we want to project it to the low-dimensional space, resulting in a new vector $\widehat{\boldsymbol{d}^*}$ with $k \times 1$ dimensions

- This is achieved through this calculation:

$$\widehat{\boldsymbol{d}^*} = \boldsymbol{\Sigma}_k^{-1} \boldsymbol{U}_k^{\mathrm{T}} \boldsymbol{d}^*$$

Exercise: examine this formula by calculating if the chain of dot products starting from $\boldsymbol{d}^*$ ends up to the correct dimension of $\widehat{\boldsymbol{d}^*}$

Latent Semantic Analysis – inference example

Inference Time (Validation/Test)

- Example: high-dimensional document $\boldsymbol{d}^*$

$$\boldsymbol{d}^* = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 3 \\ 1 \\ 0 \end{bmatrix}$$

- And the matrices calculated at train time:

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} 5.21 & 0 \\ 0 & 4.59 \end{bmatrix} \qquad \boldsymbol{\Sigma}_k^{-1} = \begin{bmatrix} 0.19 & 0 \\ 0 & 0.21 \end{bmatrix} \qquad \boldsymbol{U}_k = \begin{bmatrix} 0.04 & 0.35 \\ 0.21 & 0.16 \\ 0.94 & -0.17 \\ 0.12 & 0.88 \\ 0.18 & -0.03 \\ 0.02 & 0.20 \end{bmatrix}$$

$$\widehat{\boldsymbol{d}^*} = \boldsymbol{\Sigma}_k^{-1} \boldsymbol{U}_k^{\mathrm{T}} \boldsymbol{d}^* = \begin{bmatrix} 0.11 \\ 0.62 \end{bmatrix}$$