Natural Language Processing with Deep Learning
Sentiment Analysis with Machine Learning

Navid Rekab-Saz

navid.rekabsaz@jku.at

JOHANNES KEPLER UNIVERSITY LINZ

Institute of Computational Perception

Agenda

- Introduction to Machine Learning
- Sentiment Analysis
- Feature Extraction
- *Breaking* the curse of dimensionality!

Agenda

- **Introduction to Machine Learning**
- Sentiment Analysis
- Feature Extraction
- *Breaking* the curse of dimensionality!

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 – Recap

- Transpose
  - $\boldsymbol{a}$ is in $1\times d$ dimensions $\rightarrow$ $\boldsymbol{a}^{\mathbf{T}}$ is in $d\times 1$ dimensions
  - $\boldsymbol{A}$ is in $e\times d$ dimensions $\rightarrow$ $\boldsymbol{A}^{\mathbf{T}}$ is in $d\times e$ dimensions

- Inverse of the square matrix $\boldsymbol{S}$ is $\boldsymbol{S}^{-\mathbf{1}}$

- Dot product
  - $\boldsymbol{a}\cdot\boldsymbol{b}^{T} = c$

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

    dimensions: $1\times d \cdot d\times e = 1\times e$
  - $\boldsymbol{A}\cdot\boldsymbol{B} = \boldsymbol{C}$

    dimensions: $l\times m \cdot m\times n = l\times n$

Statistical Learning

- Given N observed data points:

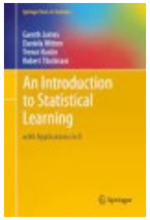
$$\boldsymbol{X} = [\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}]$$

accompanied with output (label) values:

$$\boldsymbol{y} = [y_1, y_2, \dots, y_N]$$

and each data point is defined as a vector with l dimensions (features):

$$\boldsymbol{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_l^{(i)}]$$



Statistical Learning

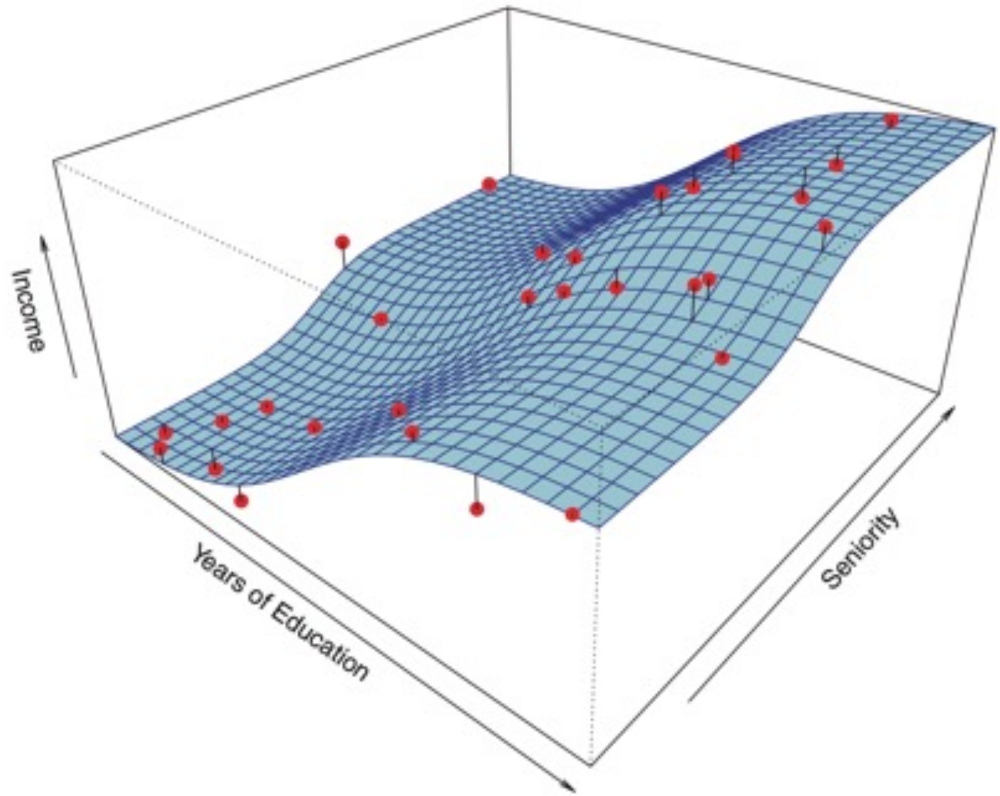
- Statistical learning assumes that there exists a **TRUE function** (f^{TRUE}) that has generated these data:

$$\boldsymbol{y} = f^{TRUE}(\boldsymbol{X}) + \epsilon$$

- f^{TRUE}
  - The *true* but *unknown* function that produces the data
  - A fixed function

- $\epsilon > 0$
  - Called **irreducible error**
  - Rooted in the constrains in gathering data, and measuring and quantifying features

Example f^{TRUE}



f^{TRUE} → blue surface

$\boldsymbol{X}$ → Red points with two features: ***Seniority***, ***Years of Education***

$\boldsymbol{y}$ → Income

ϵ → the differences between the data points and the surface

Machine Learning Model

- A machine learning (ML) model tries to estimate f^{TRUE} by defining function f:

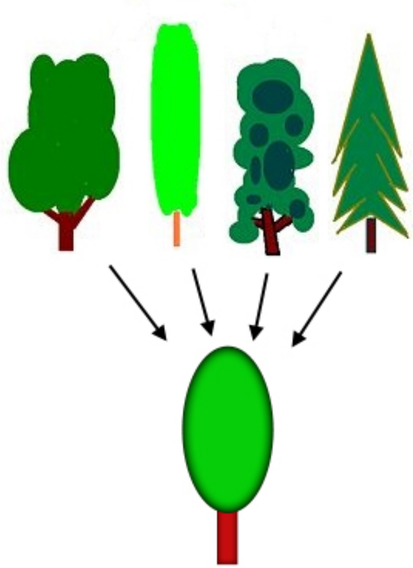
$$\hat{\boldsymbol{y}} = f(\boldsymbol{X})$$

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

- The differences between the values of $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ is **reducible error**
  - Can be reduced by better models, better estimations of f^{TRUE}

Generalization

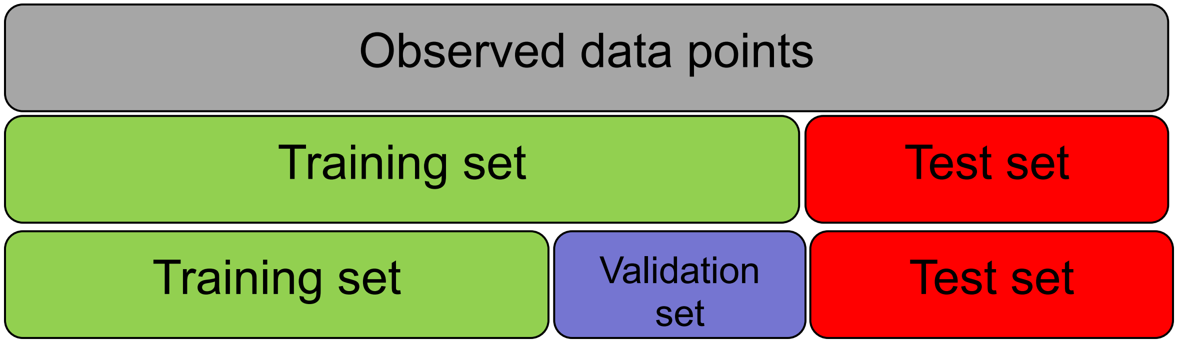
- The aim of machine learning is to create a model using **observed experiences** (training data) that generalizes to the problem domain, namely **performs** well on **unobserved instances** (test data)



link

Learning the model – Splitting dataset

- Data points are splitted 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

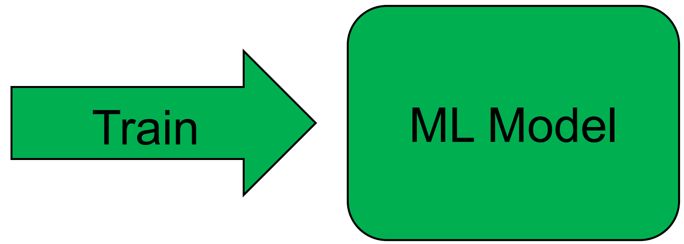
Test Set

					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	?

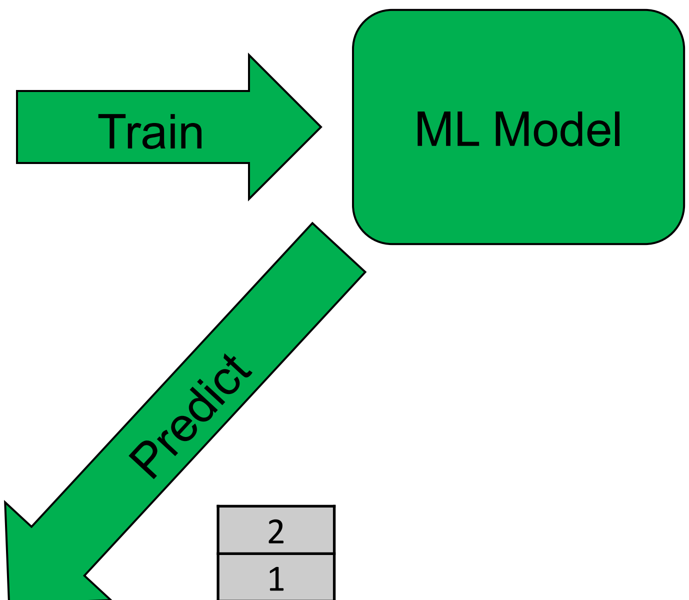
2
1
2
2
1
4

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

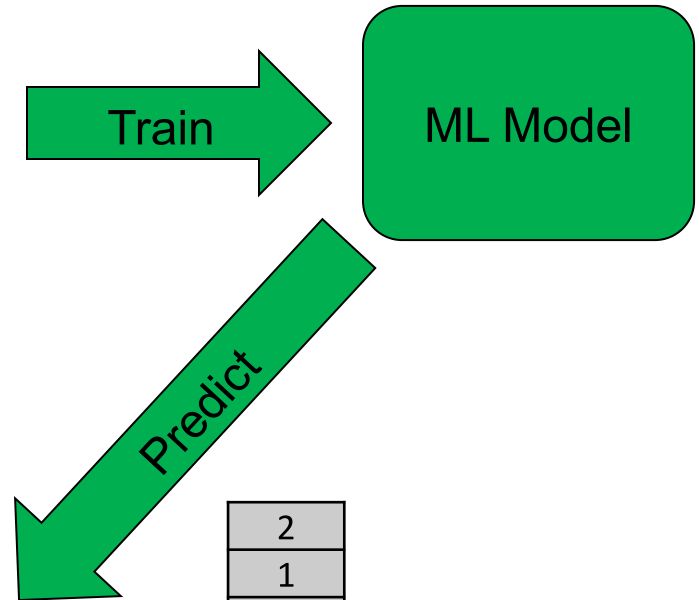
				$\hat{y}$
M	15	T	no	1
M	15	A	yes	1
F	16	T	no	2
F	16	T	no	2
F	16	T	no	3
M	17	T	no	4

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 – Generalization error

Tuning hyper parameters – Model selection

- Decide on the exploration of several sets of the model’s hyper-parameters
- Train a separate model per each set using training set
- Among the trained models, select the best performing one based on the evaluation result on validation set
- Take the selected model and evaluate it on test set → final model performance

ML models

- **Parametric** models
  - The model is defined as a function (or a family of functions) consisting of a set of parameters
  - Functions such as linear regression, logistic regression, naïve Bayes, and neural networks
  - The problem of finding the ML model is reduced to finding the optimum values for the parameters
- **Non-parametric** models
  - There is no assumption about the form of the function
  - The model is directly learned from data
  - ML models such as SVM, k-NN, smoothing spline, gaussian processes

Term of the day!

Inductive bias: all assumptions we consider in defining and creating an ML model. Our prior knowledge about what f^{TRUE} should be.

A sample ML model: Linear Regression

- f is defined as a Linear Regression function:

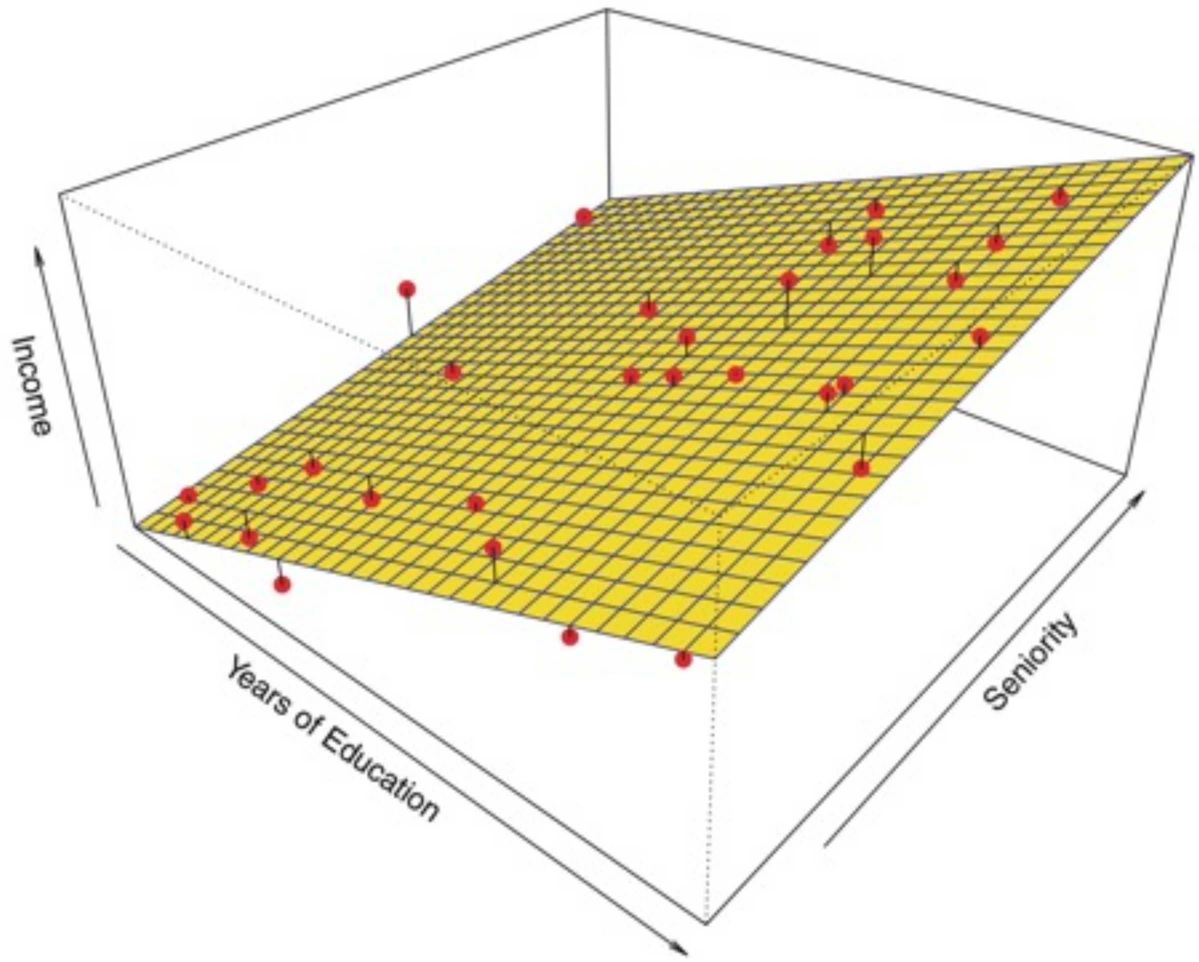
$$y = f(\boldsymbol{x}; \boldsymbol{w}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_l x_l$$

where $\boldsymbol{w} = [w_0, w_1, \ldots, w_l]$ is the set of model parameters

- In the “income” example:

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

A trained Linear Regression model



Loss Function

- Optimization of parameters is done by first defining a loss function
- A loss function measures the discrepancies between the predicted outputs $\hat{\boldsymbol{y}}$ and real ones $\boldsymbol{y}$

- E.g. **Mean Square Error** (MSE) – a common regression loss function:

$$\mathcal{L}(y_i, \hat{y}_i; \boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Loss functions for classification: Next lectures

Good to know! What is Mean Absolute Error and how is it different from MSE?

Optimization

- Next, training data is used to find an *optimum* set of parameters $\boldsymbol{w}^*$ by optimizing the loss function:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} \mathcal{L}\left(y_i, \hat{y}_i; \boldsymbol{w}\right)$$

$$\text{MSE:} \quad \boldsymbol{w}^* = \underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - f\left(x_i; \boldsymbol{w}\right)\right)^2$$

- How to optimize:
  - **Stochastically**, e.g. using Stochastic Gradient Descent (SGD) → next lecture
  - **Analytically**, e.g. in linear regression → Deep Learning book 5.1.4

ML models… cont.

Model Capacity

low ⟷ high

less flexible
less parameters
lower variance
higher bias
prune to underfitting

more flexible
more parameters
higher variance
lower bias
prune to overfitting

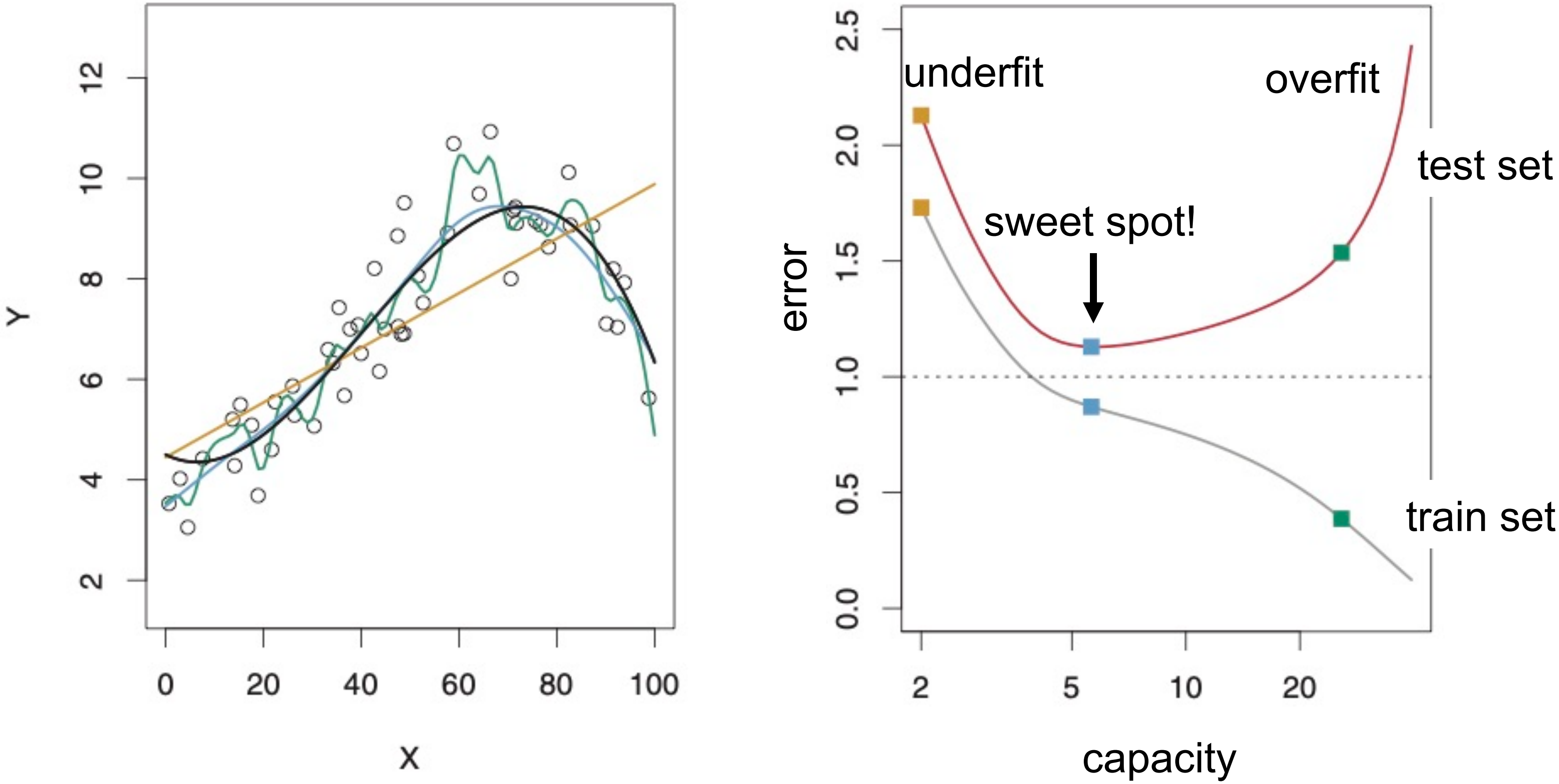
Terms of the day!

(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.

Learning Curve



Models:
black → f^{TRUE}
orange → linear regression
blue and green→ two smoothing spline models

Regularization

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

- E.g. adding the squared L2 norm of parameters to loss function:

$$\mathcal{L}(y_i, \hat{y}_i; \boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \|\boldsymbol{w}\|_2^2$$

$$\|\boldsymbol{w}\|_2 = \sqrt{\sum\nolimits_i w_i^2}$$

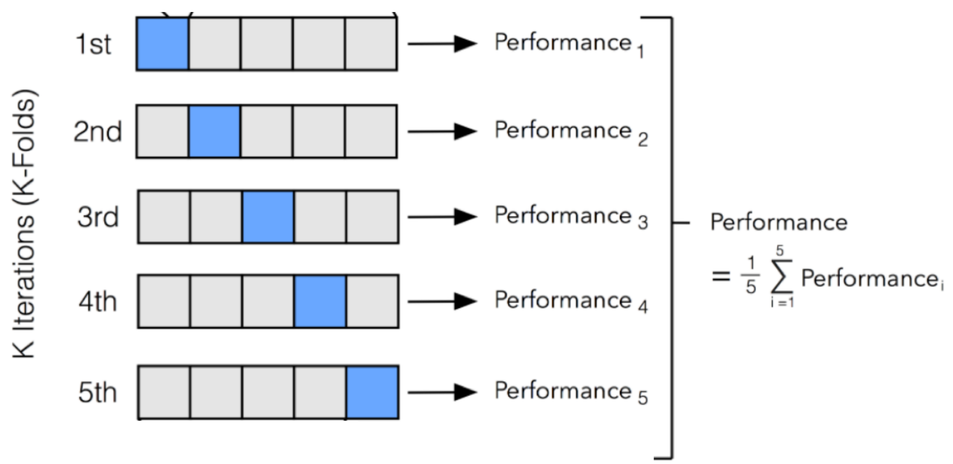
Common Evaluation Metrics

- 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

k-fold Cross Validation

- A rigours evaluation method
  - avoids bias in train/test splitting

- How to
  - Split data into *k* equal-size folds (*k=5 or* 10)
  - Repeat *k* times:
    - Use one left-out fold for test
    - Use the rest of *k-1* folds for training
  - Final performance is the average of the evaluation results of the *k* models



Agenda

- Introduction to Machine Learning
- **Sentiment Analysis**
- Feature Extraction
- *Breaking* the curse of dimensionality!

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]

Text-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!

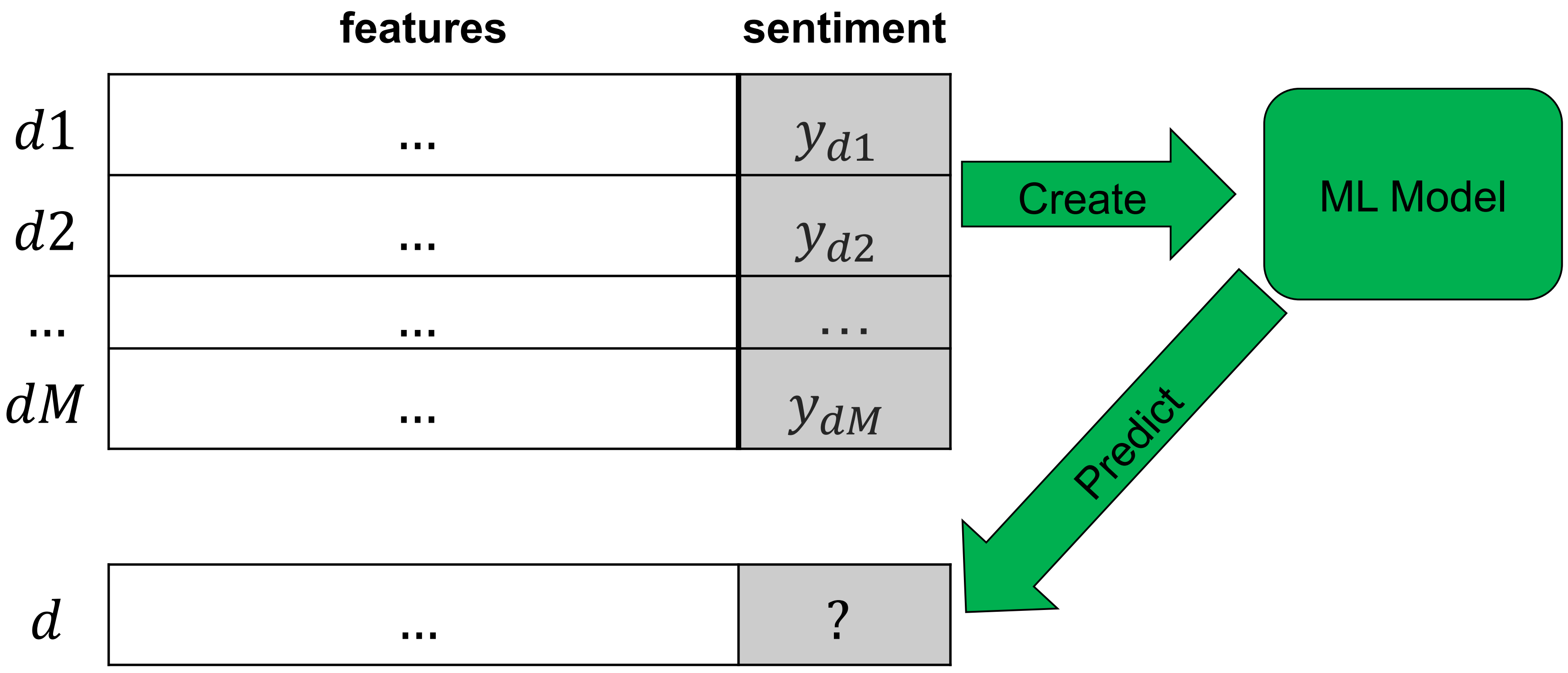
- We approach sentiment prediction with ML

Problem definition

- A dataset consist of M text documents and their sentiments (outputs)

- Possible sentiment values:
  - [-1, 0, 1] → [negative, neutral, positive] (classification problem)
  - Real-valued numbers e.g. stock price (regression problem)

Sentiment Analysis with ML



Agenda

- Introduction to Machine Learning
- Sentiment Analysis
- **Feature Extraction**
- *Breaking* the curse of dimensionality!

Dictionary

- To extract features, first a dictionary with N words (terms) is defined:

$$[t1, t2, \dots, tN]$$

Document-Term Matrix

- The features are based on the terms in the dictionary
  - Bag of Words (BoW) representations of documents
- $x_{t,d}$ is feature value → weight of term t in document d

	$t1$	$t2$	…	tN	**sentiment**
$d1$	$x_{t1,d1}$	$x_{t2,d1}$	…	$x_{tN,d1}$	y_{d1}
$d2$	$x_{t1,d2}$	$x_{t2,d2}$	…	$x_{tN,d2}$	y_{d2}
…	…	…	…	…	…
dM	$x_{t1,dM}$	$x_{t2,dM}$	…	$x_{tN,dM}$	y_{dM}

Term Weightings

- A term weighting method measures the importance of a term in a document

- One common method is to count the number of occurrences of a term in a document $\Longrightarrow$ **term count**:
$$x_{t,d} = \mathrm{tc}_{t,d} = \#\ \text{of occurrences of}\ t\ \text{in}\ d$$

- Using logarithm to dampening raw counts is shown to be more effective $\Longrightarrow$ **term frequency**:
$$x_{t,d} = \mathrm{tf}_{t,d} = \log(1 + \mathrm{tc}_{t,d})$$

On informativeness of less frequent terms

- Terms that do not appear often usually carry more information in comparison with highly frequent ones
  - e.g., ***JKU*** in a large news corpora

- Inverse document frequency (idf) is a well-known method to measure how often words appear in a collection:

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

  - df_t is the document frequency of t, namely the number of documents that contain term t
  - Higher idf_t means that the term appears less often in the collection, and is therefore more informative (important)
    - e.g., ***JKU*** has high idf, while ***the*** has very low idf

Term weightings

- $\text{tf-idf}_{t,d}$ term weighting is the product of $\text{tf}_{t,d}$ and idf_t

$$x_{t,d} = \text{tf-idf}_{t,d} = \log(1 + \text{tc}_{t,d}) \times \log({}^{M}/_{\text{df}_t})$$

increases with the number of occurrences within a document

increases with the rarity of the term in the collection

- A well-known term weighting method!

Wrap-up!

- Use any term weightings to create document-term matrix

	$t1$	$t2$	...	tN	**sentiment**
$d1$	$x_{t1,d1}$	$x_{t2,d1}$	...	$x_{tN,d1}$	y_{d1}
$d2$	$x_{t1,d2}$	$x_{t2,d2}$	...	$x_{tN,d2}$	y_{d2}
...	...	...	...	...	$\dots$
dM	$x_{t1,dM}$	$x_{t2,dM}$	...	$x_{tN,dM}$	y_{dM}

- The rest is standard machine learning!
  - Model training
  - Hyper-parameter tuning and model selection
  - Evaluation

Agenda

- Introduction to Machine Learning
- Sentiment Analysis
- Feature Extraction
- ***Breaking* the curse of dimensionality!**

Supervised Sentiment Analysis

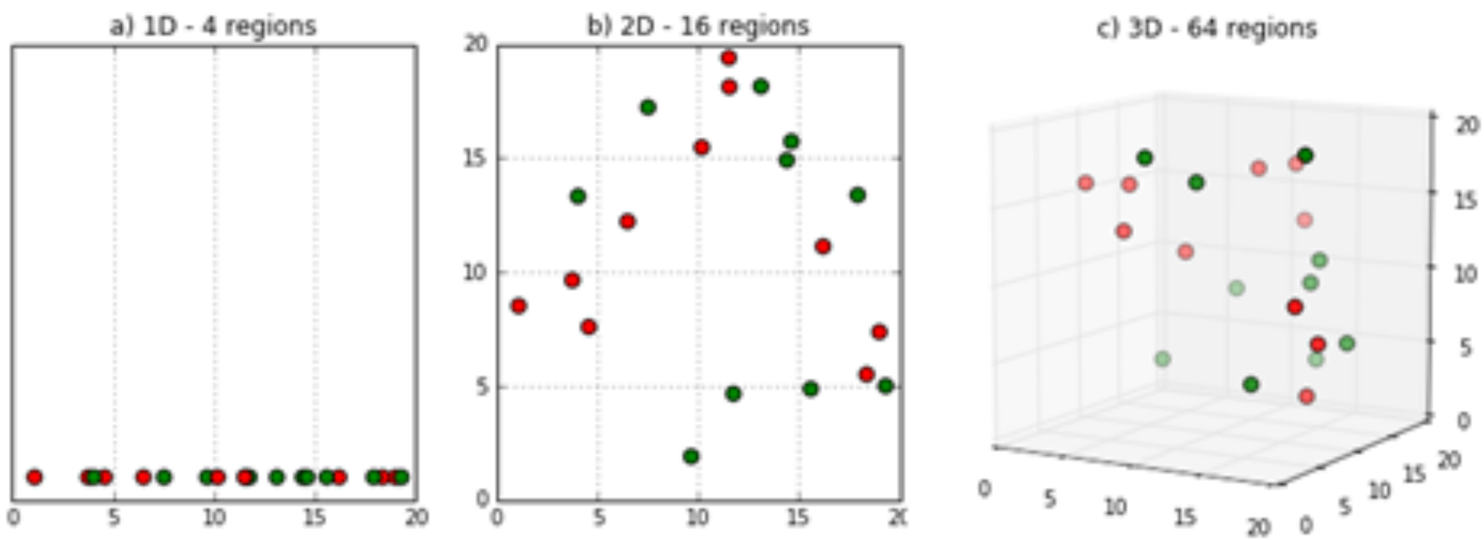
	$t1$	$t2$	…	tN	**sentiment**
$d1$	$x_{t1,d1}$	$x_{t2,d1}$	…	$x_{tN,d1}$	y_{d1}
$d2$	$x_{t1,d2}$	$x_{t2,d2}$	…	$x_{tN,d2}$	y_{d2}
…	…	…	…	…	…
dM	$x_{t1,dM}$	$x_{t2,dM}$	…	$x_{tN,dM}$	y_{dM}

$N \sim [20K - 500K]$
$M \sim [10K - 100K]$

- The feature vectors are
  - sparse (a lot zeros)
  - in a very high dimension

Curse of dimensionality

- Curse of dimensionality happens when the amount of data does not suffice to support the sparsity in dimensionality
- It causes
  - Data sparsity
  - Issues in measuring "closeness"



link

Curse of dimensionality

- Why low-dimensional vectors?
  - Easier to store and load (efficiency)
  - More efficient when used as features in ML models
  - Better generalization since the noise in data is reduced
  - Able to capture higher-order relations:
    - Synonymy like *car* and *automobile* can be merged into same dimensions
    - Polysomy like *bank (financial institution)* and *bank (bank of river)* can be separated into different dimensions

Feature (Dimensionality) reduction

- Feature selection
  - keep some important features and get rid of the rest!

- Dimensionality reduction
  - project data from high to a low dimensional space

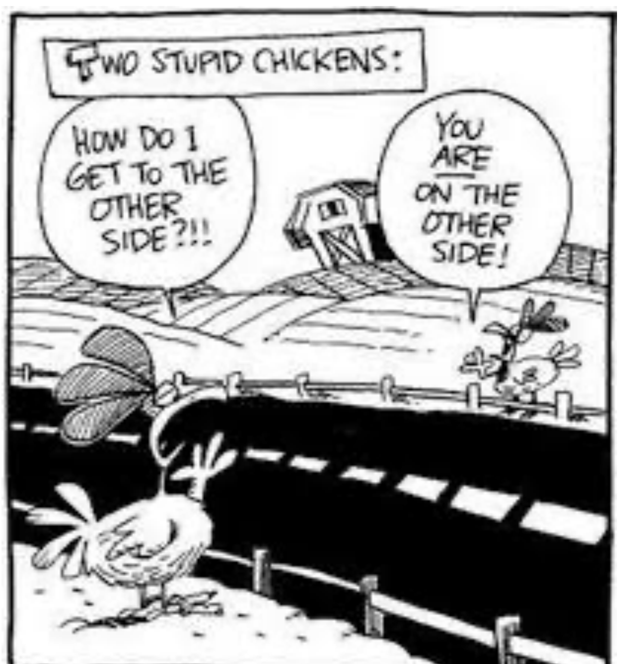
Feature selection

- During pre-processing
  - Remove stop words or very common words
    - $\mathrm{tf{-}idf}$ do it in a "soft" way *why?*
  - Remove very rare words
    - Usually done when creating dictionary
  - Stemming & lemmatization
- Features definition
  - Use only the words in a domain-specific lexicon as features
- Post-processing
  - Keep important features using some informativeness measures
  - Subset selection

Dimensionality reduction with LSA

- Latent Semantic Analysis (LSA)
  - A common method in Information Retrieval to capture semantics
  - Based on Singular Value Decomposition (SVD)

Semantics matters!



Singular Value Decomposition

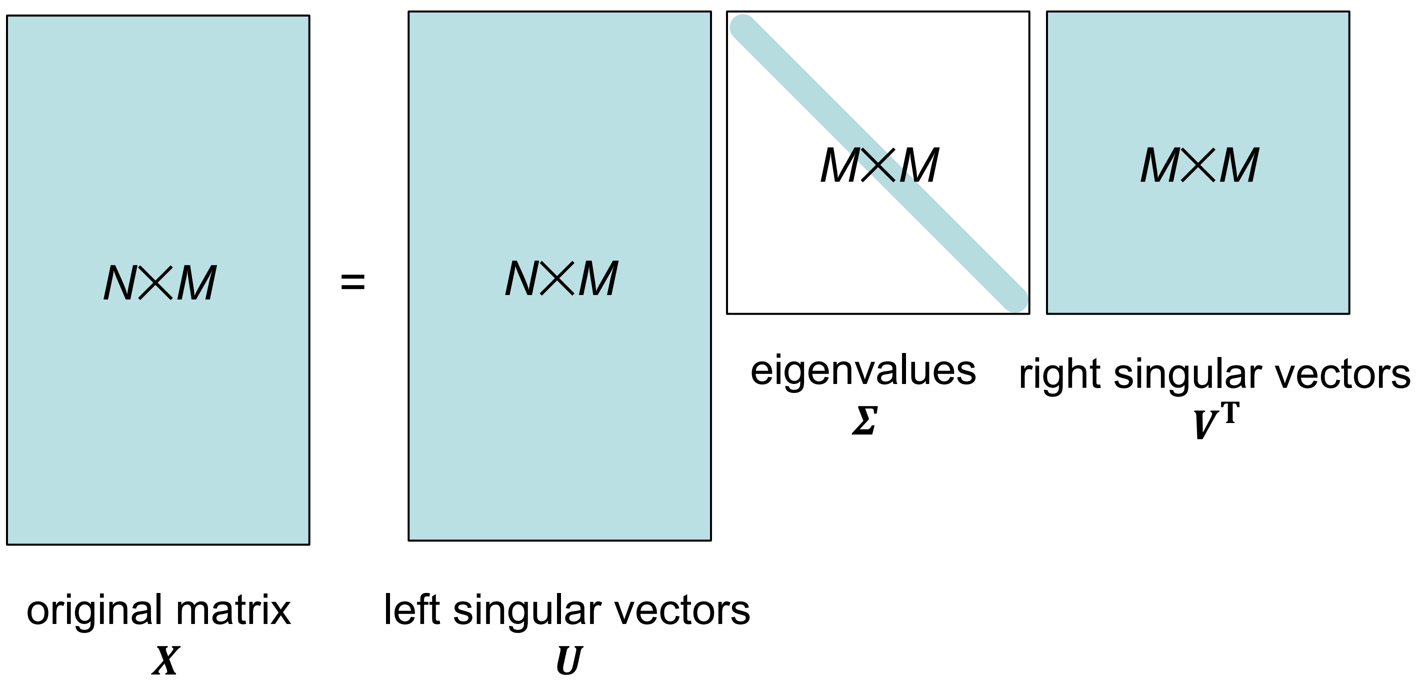
- An $N \times M$ matrix $\boldsymbol{X}$ can be factorized to three matrices:

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

- $\boldsymbol{U}$ (left singular vectors) is an $N \times M$ unitary matrix
- $\boldsymbol{\Sigma}$ is an $M \times M$ diagonal matrix, diagonal entries
  - are eigenvalues,
  - 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

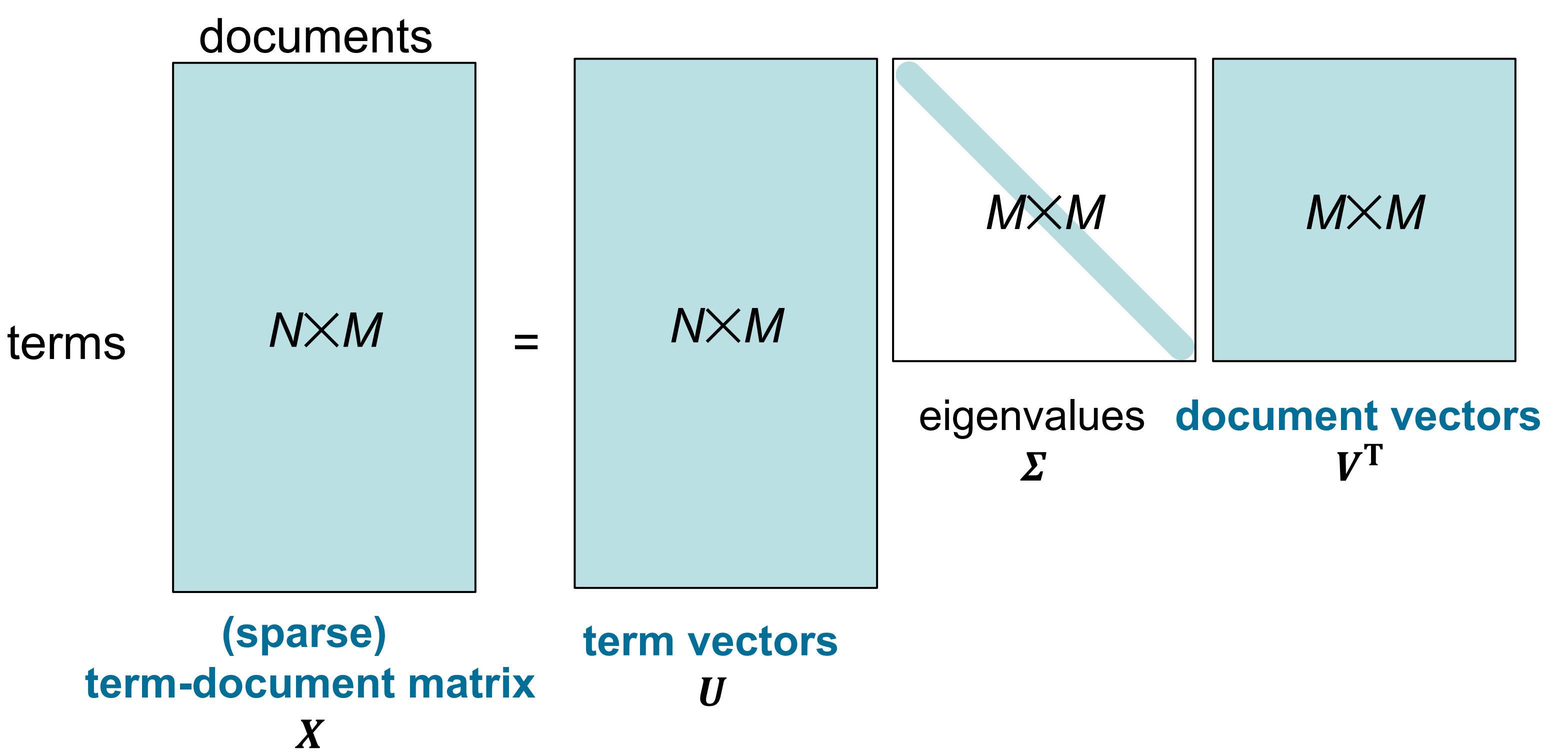


Latent Semantic Indexing

At Training Time

- Step 1: apply SVD on the **term-document** matrix of training data

☞ Not the document-term matrix! Although it is also possible to start with the document-term matrix, we follow the typical definition!



Latent Semantic Indexing

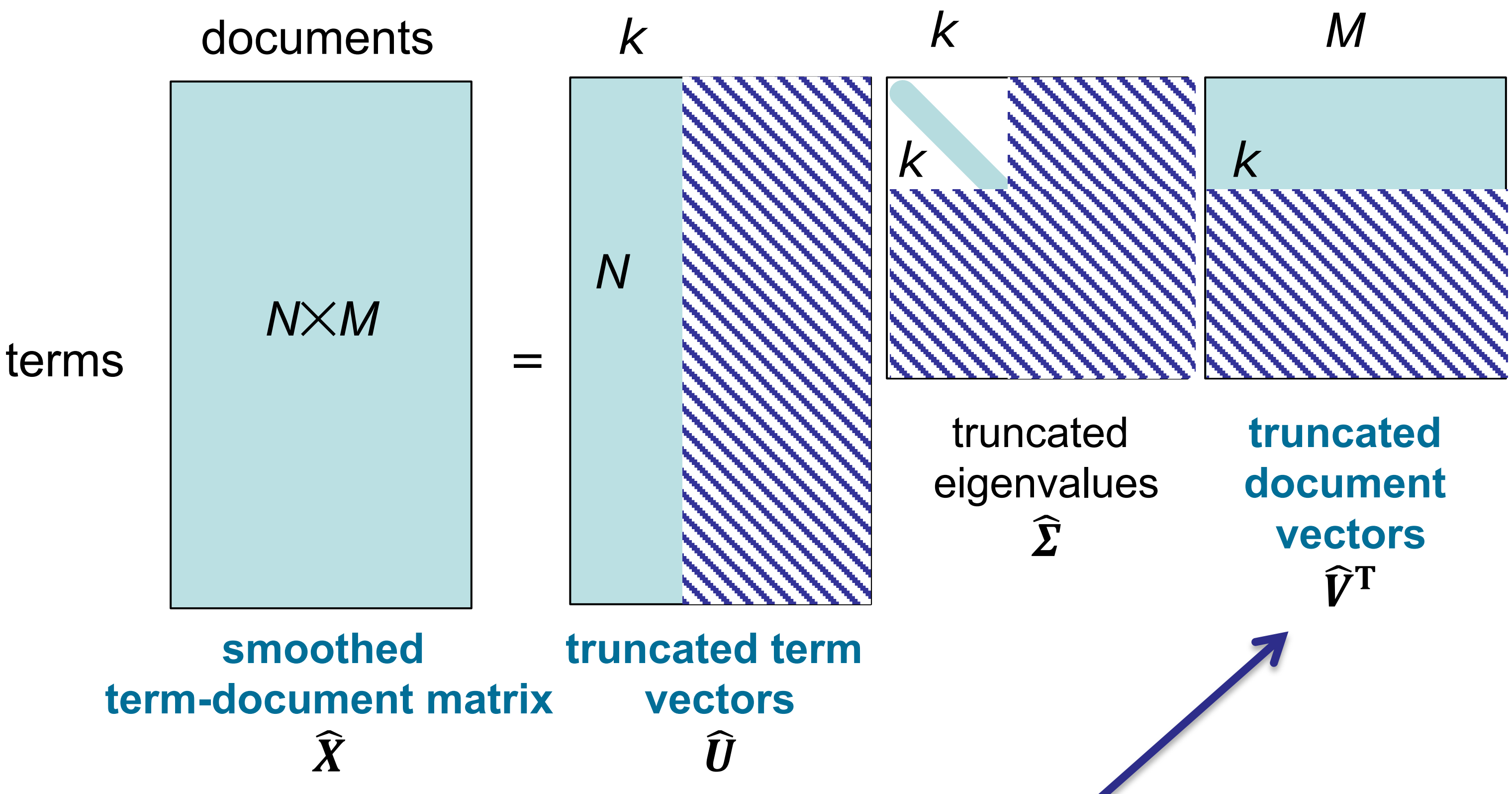
At Training Time

- Step 2: keep only top *k* eigenvalues in $\boldsymbol{\Sigma}$ and set the rest to zero, called $\widehat{\boldsymbol{\Sigma}}$
- Truncate the $\boldsymbol{U}$ and $\boldsymbol{V}^{\mathrm{T}}$ matrices based on the changes in $\boldsymbol{\Sigma}$, called $\widehat{\boldsymbol{U}}$ and $\widehat{\boldsymbol{V}}^{\mathrm{T}}$ respectively
- If we multiply the truncated matrices, we have the rank *k* least-squares approximation to the original matrix

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

Latent Semantic Indexing

At Training Time



- $\hat{\boldsymbol{V}}$ matrix is the dense low-dimensional document vectors

Latent Semantic Indexing

At Inference Time (Validation or 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

- done through this calculation:

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

- Exercise: calculate if the chain of dot products from $\boldsymbol{d}$ to $\widehat{\boldsymbol{d}}$ results to the correct dimension