

9. Classification, Clustering, and Learning to Rank

Prof. Dr. Goran Glavaš

Center for AI and Data Science (CAIDAS)
Fakultät für Mathematik und Informatik
Universität Würzburg



Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International

After this lecture, you'll...

2

- Know the very basics of machine learning
- Understand supervised text classification
- Know some methods for (unsupervised) text clustering
- Understand how to combine different ranking functions (and other features) in a supervised IR setting – learning to rank

Outline

3

- [Recap of Lecture #8](#)
- Primer on Machine Learning
- Text Classification
- Text Clustering
- Learning to Rank

Recap of the previous lecture

4

- Latent and Semantic Retrieval
 - **Q:** Why is term matching sometimes not good enough for retrieval?
 - **Q:** When should you use term-based IR models and when semantic/latent ones?
- Latent Semantic Indexing
 - **Q:** What Latent Semantic Indexing (LSI)?
 - **Q:** What is Singular Value Decomposition and how are latent topics represented?
 - **Q:** How do we obtain latent representations of documents and terms? How to transform the query into latent space?
- Latent Dirichlet Allocation
 - **Q:** What is LDA and how are latent topics represented in this probabilistic setting?
 - **Q:** What is the generative story that LDA assumes?
- Word embeddings for IR
 - **Q:** How are word embedding models different from latent topic models?
 - **Q:** How does CBOW model learn word embeddings?
 - **Q:** How to exploit word embeddings for an IR model?

LSI – Singular Value Decomposition

5

- Given a matrix **A** (with non-negative elements), the **Singular Value Decomposition** finds **orthogonal** matrices **U** and **V** and a rectangular diagonal matrix **Σ** such that:

$$A = U\Sigma V^T$$

- Matrix **U** is of dimensions **M x M**
- Matrix **V** is of dimensions **N x N**
- Matrix **Σ** is of dimensions **M x N**
- U and V are orthogonal: **U^TU = I**, **V^TV = I**
- Values of the diagonal matrix **Σ** are singular values of the original matrix **A**
- Let **r** be the rank of matrix **A**

LSI reduction – example

6

- This leaves us with the **best possible** approximation of rank \mathbf{A}_K ($K = 2$ in our example) of the original term-document occurrence matrix \mathbf{A}

Dense vectors of terms

\mathbf{U}_K

<i>president</i>	-0.43	0.13
<i>minister</i>	-0.53	0.25
<i>speech</i>	-0.58	0.33
<i>law</i>	-0.12	-0.05
<i>ball</i>	-0.22	-0.51
<i>score</i>	-0.26	-0.62
<i>player</i>	-0.22	-0.40
<i>run</i>	-0.03	-0.06
<i>person</i>	-0.11	-0.03
<i>piano</i>	-0.10	-0.02
<i>mouse</i>	-0.09	-0.08

Dense vectors of documents

d_1	d_2	d_3	d_4	d_5	d_6
-4.66	-4.37	-2.71	-2.37	-1.51	-1.65
2.01	2.12	0.49	-4.23	-2.93	-3.35

$\Sigma_K \mathbf{V}_K^T$

- \mathbf{A}_K has the same dimensions as original \mathbf{A} ($M \times N$)
- \mathbf{U}_K is of size $M \times K$, and $\Sigma_K \mathbf{V}_K^T$ of size $K \times N$

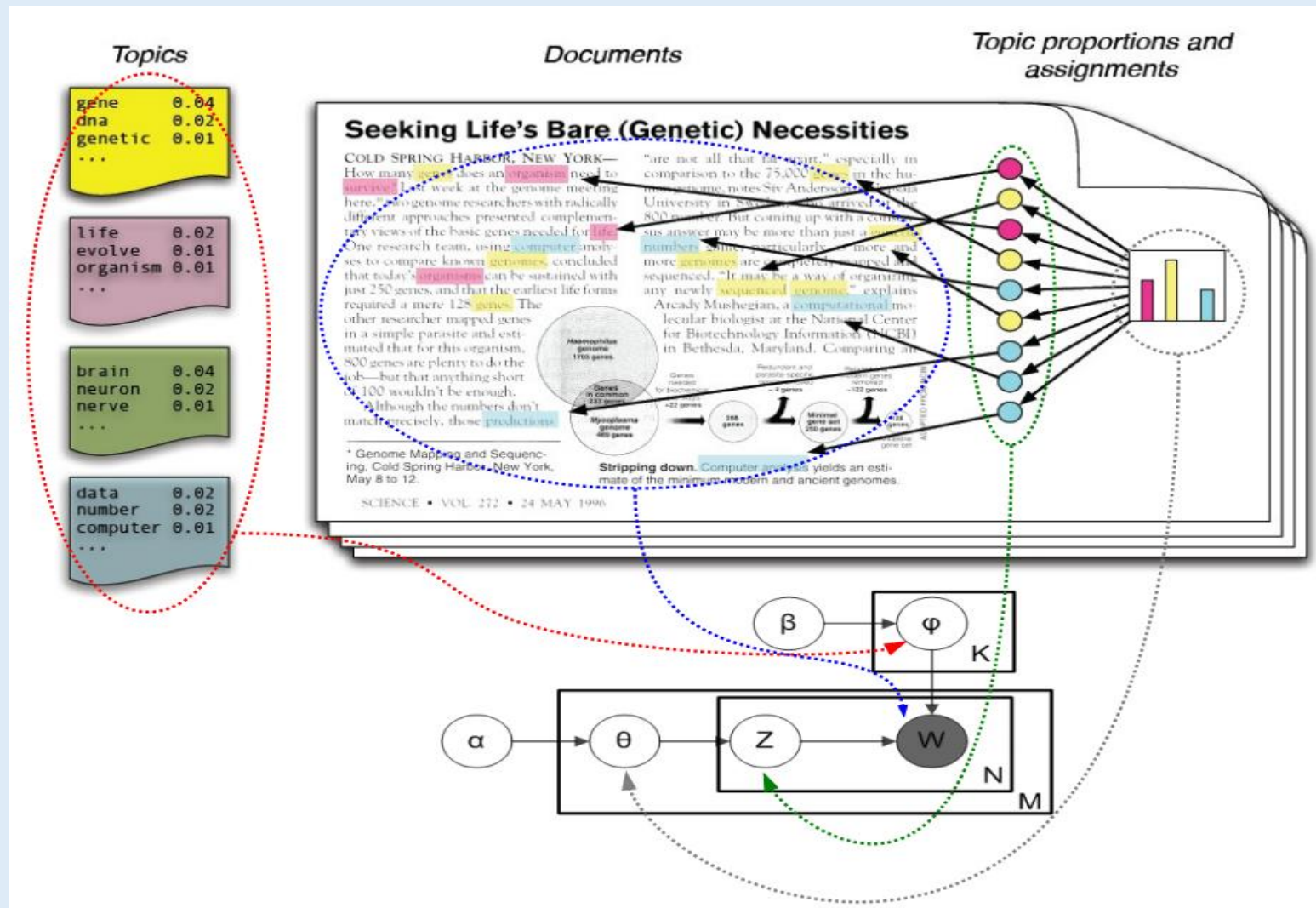
LDA – Generative View

7

1. For each topic k ($k = 1, \dots, K$):
 - Draw parameters of a multinomial distribution φ_k (over terms) for topic k from a Dirichlet distribution $Dir_N(\beta)$
2. For each document d in the collection:
 - Draw parameters of a multinomial distribution of topics for the document d , θ_d , from a Dirichlet distribution $Dir_K(\alpha)$
 - For each term position w_{dn} in the document d :
 - a) Draw a topic assignment (i.e., a concrete multinomial distribution over terms) z_{dn} from $Mult_K(\theta_d)$
 - b) Draw a concrete term w_{dn} from the multinomial distribution over terms of the topic z_{dn} (drawn in a)), $Mult_N(\varphi_{z_{dn}})$

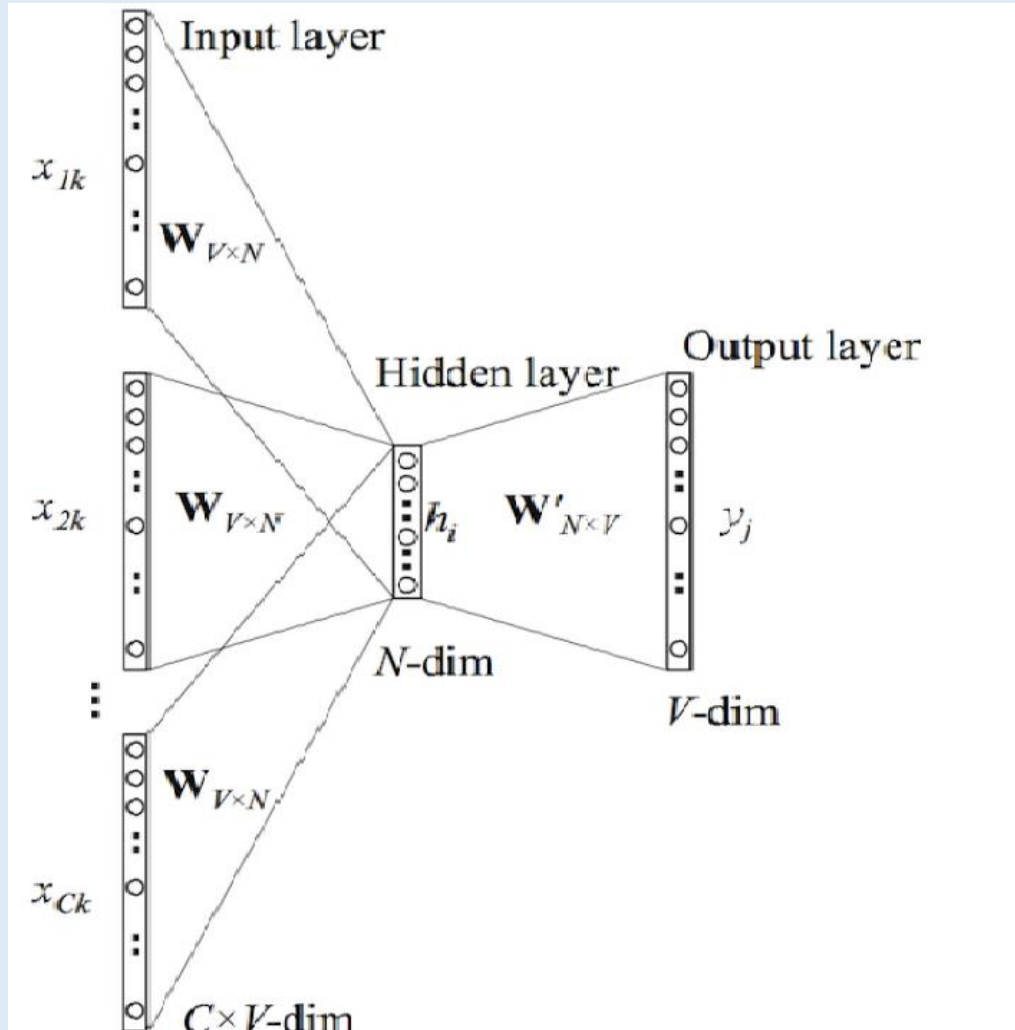
LDA – Generative View

8



Continuous Bag-of-Words (CBOW)

9



- Context consists of C words, with corresponding one-hot vectors
 - $x_{1k}, x_{2k}, \dots, x_{Ck}$
- One-hot vectors transformed to dense vectors using input matrix \mathbf{W} ($V \times N$)
- Dense context vector h is obtained as:

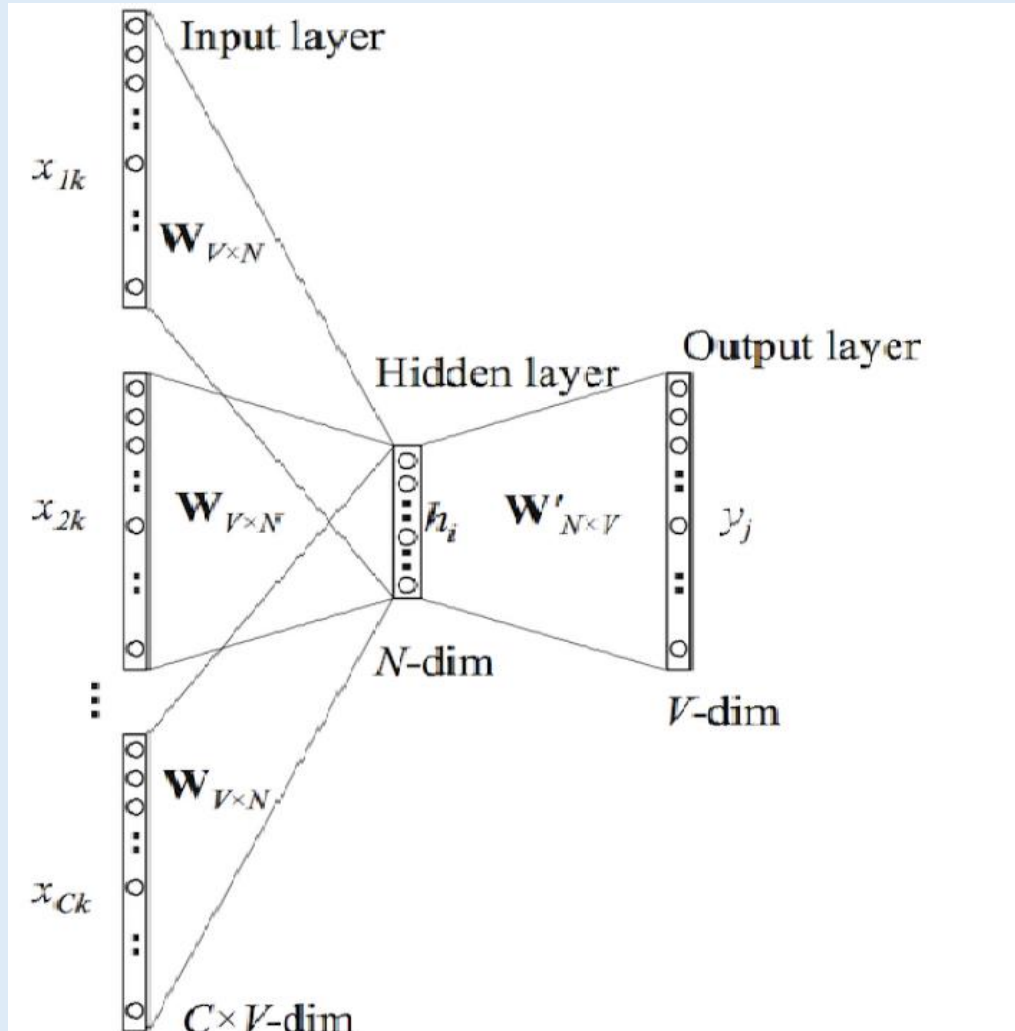
$$h = \frac{1}{C} \mathbf{W} \left(\sum_{i=1}^C x_{ik} \right)$$

- Dense context vector h is then multiplied with the output matrix \mathbf{W}' ($N \times V$)

$$y_k = \text{softmax}(h^T \mathbf{W}')$$

Continuous Bag-of-Words (CBOW)

10



- Output vector y needs to be **as similar as possible** to one-hot vector of **center word**
- Parameters of the model are elements of \mathbf{W} and \mathbf{W}'
 - Each row of \mathbf{W} is the **dense context vector** of one vocabulary word
 - Each column of \mathbf{W}' is the **dense center vector** of one vocabulary word
- Dense representation (**embedding**) of the i -th vocabulary term is concatenation of
 1. i -th row of \mathbf{W} and
 2. i -th column of \mathbf{W}'

Outline

11

- Recap of Lecture #8
- [Primer on Machine Learning](#)
- Text Classification
- Text Clustering
- Learning to Rank

Why machine learning?

12

- For many IR and NLP tasks, it is **difficult** to come up with an **explicit** (i.e., **rule-based**) algorithm that solves the task **efficiently**
- For example
 - **POS tagging** – difficult to devise the closed set of rules that infer the POS tag of the words from the word's context
 - **Sentiment analysis** – complete set of rules that determine the sentiment of a review?
 - **Named entity recognition** – a manually defined finite state automaton that recognizes the sequences of words that form named entities?
 - **Semantic textual similarity** – measure the word overlap and manually determine the threshold according to which two texts are considered similar?

Why machine learning?

13

- The **problems** with devising **rule-based systems** for complex tasks are numerous:
 1. We simply **need to many rules** to cover all the cases
 2. There are **many exceptions** (including exceptions to exceptions!) to be handled
 3. We **need expert knowledge** (i.e., an expert to handcraft the rules)
 4. Rules can be **difficult** to
 - Design – rules interact in unpredictable ways
 - Maintain – adding new rules can easily break everything
 - Adopt to new domains – we need to significantly modify/add rules
- IR and NLP tasks are often **inherently subjective** (e.g., relevance of a document for the query)
 - It is **difficult to model subjectivity** with rules

Why machine learning?

14

- It is often **easier to manually label** some concept than to design an explicit algorithm that captures the concept automatically
- Labeling typically **does not require** too much expert knowledge
- We don't care how complex or subjective the task is
 - We let the data „**speak for itself**” and machine learning algorithm to do the work
- If we're lucky, the labeled data might be already **readily available** (e.g., reviews with assigned ratings)

Machine learning basics

15

- **Supervised machine learning**

- We have labeled data as input
- Supervised ML algorithms learn the mapping between input representations and output labels
- **Classification**: output is a discrete label (no ordering between the labels)
- **Regression**: output is an integer or real value (obviously, there is ordering)

- **Unsupervised machine learning**

- We have no labels (i.e., we have unlabeled data) at input
- **Clustering**: grouping instances by the similarity of their representations
- **Outlier detection**: recognizing instances that are very dissimilar from all other instances in the dataset

Supervised machine learning

16

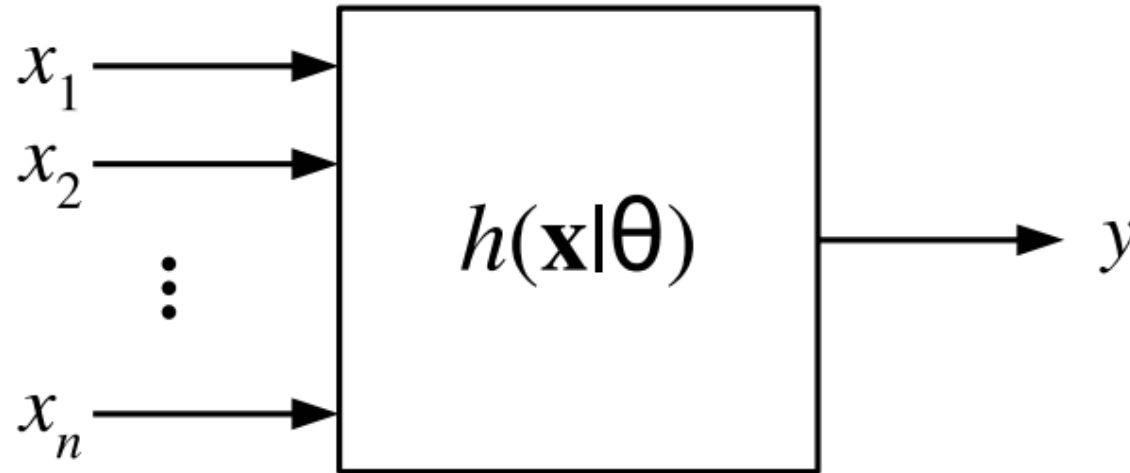
- **Supervised machine learning** models „**learn**” the mapping between input values and output values
- A single input to the classifier is called an **instance** or **example** (denoted „**x**”)
 - An instance is represented as an n-dimensional feature vector

$$\mathbf{x} = (x_1, x_2, \dots, x_n)$$

- The desired output is called the **target label** (or just label, denoted **y**)
- A classifier h maps an instance \mathbf{x} to a label y – $h : \mathbf{x} \rightarrow y$
- „**Learning**” – model has **parameters** θ (denoted $h(\mathbf{x} | \theta)$) whose values are **optimized** to maximize the prediction accuracy of the **output labels**, given instance

Supervised classification

17



- Types of classifiers in IR/NLP:
 - **Binary classification**: just two output labels (yes/no, 0/1)
 - **Multi-class classification**: each instance has one of K labels
 - **Multi-label classification**: an instance can have more than one label at once
 - **Sequence labeling**: input is a sequence of instances and the output is the sequence of labels

Supervised classification

18

- **Training** (or **learning**) – adjustment of model parameters θ so that the **classification error is minimized**
 - The error is computed on a labeled training set – this is the **training error**
- The **training error** is minimized with an **optimization method**
 - ML algorithms differ in **optimization criteria** and **optimization method** they use
- We want to know how classifier works on **new, unseen instances**
 - This property is called **generalization** – the classifier **must generalize well**
 - **Testing error** – the error computed on instances not used for training
- ML models can be of different **complexity**
 - The **more parameters** the model has, the **more complex** it is
 - The model may be **too simple** or **too complex** for the task at hand
 - **Underfitting** (model too simple for the task): both training and test errors are big
 - **Overfitting** (model too complex for the task): training error small, test error big

Outline

19

- Recap of Lecture #8
- Primer on Machine Learning
- **Text Classification**
- Text Clustering
- Learning to Rank

Text Classification

20

- **Text Classification** is the automated categorization of some unit text (sentence, paragraph, document) into one (or more) of predetermined labels
 - E.g., classify news stories into high-level topics: *politics, sport, culture, entertainment*
- Why text classification in IR?
 - Automatically assigned classes/labels provide an **additional semantic layer**
 - These additional semantic annotations can be exploited to **rerank/filter** results
 - E.g., **Query**: „*lionel messi*” (but retrieve only documents categorized as *sport*)
- Some popular ML algorithms for text classification:
 - Traditional: Naive Bayes classifier, **Logistic regression**, (linear) SVM
 - Recent: **Convolutional neural networks (CNN)**

Text representations

21

- For the majority of text classification algorithms, **instances of text** need to be transformed to **numeric vector representations**
 - **Exceptions**: Naive Bayes classifier and Decision Trees/Random Forests which can directly use word-based representations of text
- Numeric vector representations may be:
 1. **Sparse** – each text is represented as (potentially weighted) vectors of word occurrences, the size of the vector is the size of vocabulary
 2. **Dense** – each text is represented by a semantic dense vector (or by a concatenation of dense vectors of its constituent words)
- Traditional text classification models like **logistic regression** or **SVM ignore** the order of words in the text
 - I.e., they use **bag-of-words** representation of text
- Convolutional neural networks do take into account the order of words in the text
 - They compute **abstract representations** of subsequences of text

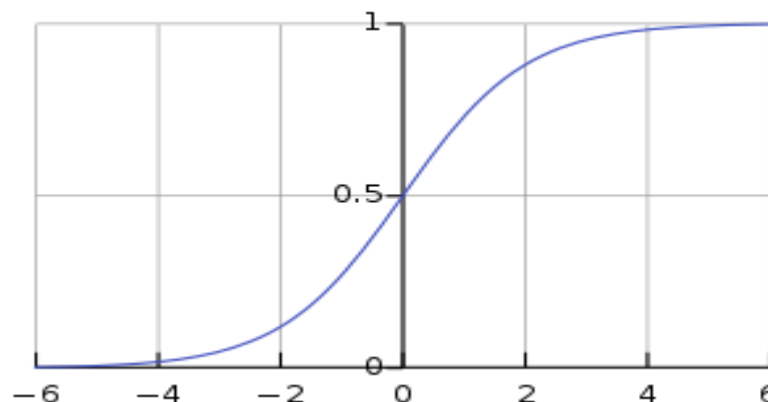
Logistic regression

22

- Despite its name, **logistic regression** is a classification algorithm
 - We will focus on binary classification – logistic regression computes the probability that some instance \mathbf{x} belongs to some class ($y = 1$)

$$h(\mathbf{x} | \boldsymbol{\theta}) = P(y = 1 | \mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{x})} = \sigma(\boldsymbol{\theta}^T \mathbf{x})$$

- Logistic regression is based on a logistic function: $\sigma(a) = 1 / (1 + e^{-a})$
- The logistic function maps the input value to the output interval $[0, 1]$



Logistic regression

23

- Looking at the logistic regression formula (and the properties of log. function):
 - $h(\mathbf{x}|\boldsymbol{\theta}) > 0.5$ (i.e., instance belongs to the class) if and only if $\boldsymbol{\theta}^T \mathbf{x} > 0$
 - $h(\mathbf{x}|\boldsymbol{\theta}) < 0.5$ (i.e., instance doesn't belong to the class) if and only if $\boldsymbol{\theta}^T \mathbf{x} < 0$
- In order to make predictions, we need to know the **parameter vector $\boldsymbol{\theta}$**
 - We learn the values of parameters by **minimizing some error function** for the set of **training instances**
 - Logistic regression minimizes the so-called **cross-entropy error**

$$J(\boldsymbol{\theta}) = - \sum_i y^i * \log(h(\mathbf{x}^i|\boldsymbol{\theta})) + (1 - y^i) * \log(1 - h(\mathbf{x}^i|\boldsymbol{\theta}))$$

- $J(\boldsymbol{\theta})$ is minimized (i.e., parameters $\boldsymbol{\theta}$ are optimized) via numeric optimization
 - Most commonly using **stochastic gradient descent (SGD)**

Convolutional neural network

24

- **Convolutional neural network** is a neural machine learning model that has been successfully used for text and image classification tasks
 - Unlike bag-of-words classifiers, treats text as an ordered sequence of words
 - Requires a **dense representation** of text as input – we typically represent text as (2D) **concatenation of word embeddings**
- CNNs parameters are **convolution filters** – real-valued matrices that are being used to compute the convolution with the partso of the input sequence
- The convolutional layer is followed by the **max-pooling layer** – where only the top K largest convolution scores are taken
- The final prediction is made by the **softmax regression** (generalization of the logistic regression for more than two labels)

Convolutional neural network

25

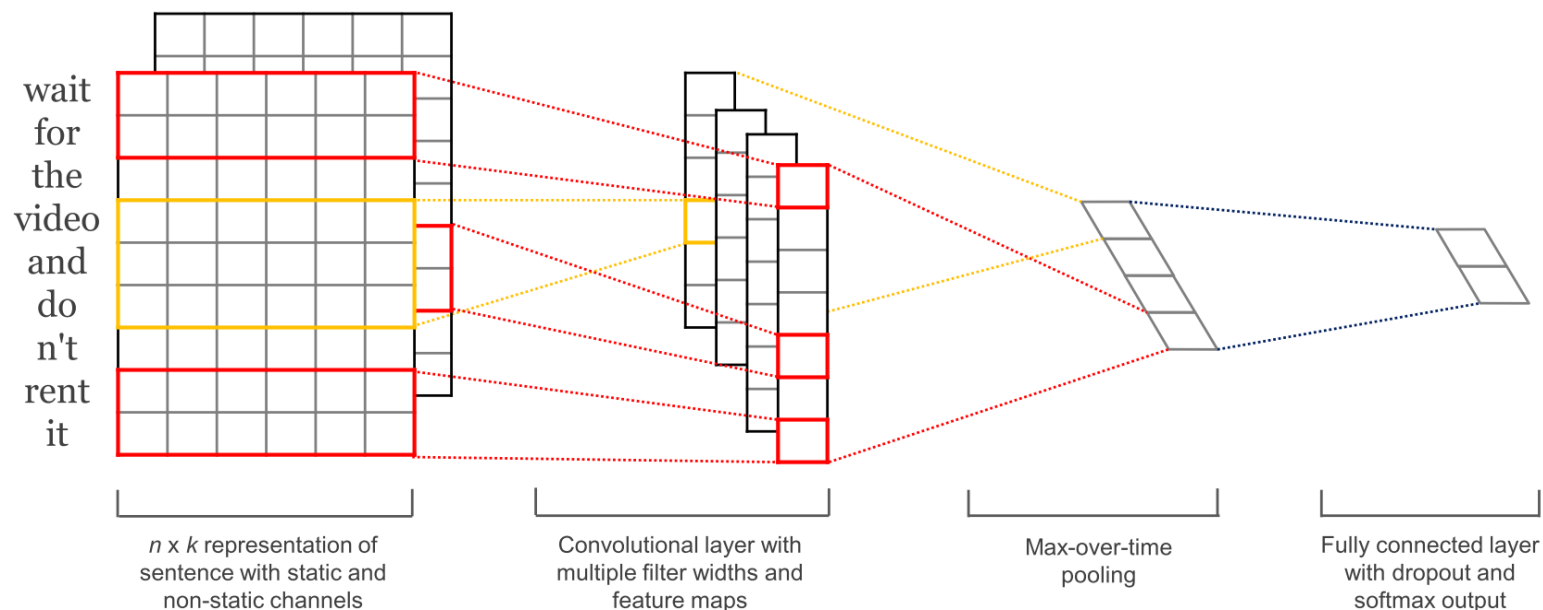


Image taken from: <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>

- CNNs parameters (real-values of all convolution filter matrices) are learned by **propagating the classification error** via **backpropagation algorithm**

Outline

26

- Recap of Lecture #8
- Primer on Machine Learning
- Text Classification
- **Text Clustering**
- Learning to Rank

Cluster Analysis

27

- **Cluster analysis** (or, colloquially, **clustering**) is a multivariate statistical technique that allows automated generation of groupings in data
- Components of clustering:
 1. An **abstract representation** of an object using which the object is compared to other objects
 2. A **function** that measures the **distance or similarity** between the objects based on their abstract representations
 3. A **clustering algorithm** that groups the objects based on the similarities / distances computed from their representations
 4. **(optional) Constraints** with respect to cluster membership, cluster proximity, shape of the clusters, etc.

Text clustering

28

- **Representations of text** for clustering are typically similar as for text classification (only we lack the labels)
 - **Sparse vectors** (binary or weighted, e.g., using TF-IDF)
 - **Dense vectors** (latent or semantic representations)
 - Sometimes also more structured representations like **trees** or **graphs**
- Common **distance/similarity** functions
 - Euclidean distance, cosine similarity/distance, Jaccard coefficient, Kullback-Leibler divergence, tree/graph kernels for structured representations (trees/graphs)
- Clustering algorithms:
 1. Sequential – e.g., **single pass clustering**
 2. Hierarchical – e.g., agglomerative clustering, divisive clustering
 3. Cost-function optimization clustering – e.g., **K-means**, mixture of Gaussians

Cluster information retrieval

29

- Why clustering in information retrieval?
 - We have already seen clustering at work in speeding up VSM retrieval (**leaders**)
- **Cluster information retrieval model**
 - **Cluster hypothesis** (**van Rijsbergen, 1979**): Documents similar in content tend to be relevant for the same queries
 - Steps:
 1. Collection documents are pre-clustered
 2. The query is matched against cluster centroids
 3. All documents from clusters represented by top-ranked centroids are returned (ranked)
 - **Improves efficiency** as the query needs not be compared with all documents
 - No comparison with documents from clusters with low-ranked centroids

Single pass clustering

30

- Simplest clustering algorithm
 - The number of clusters does not need to be predefined
- Algorithm:
 1. Start by putting the first text t_1 into the first cluster $c_1 = \{t_1\}$
 2. For all other texts, t_2, \dots, t_n , one by one
 - I. Measure the distance/similarity with all existing clusters c_1, \dots, c_k
 - The similarity with the cluster is avg/max of similarities with instances in cluster
 - II. Identify the cluster c_i with which the current text t_j has the largest similarity (or smallest distance)
 - III. If the similarity between t_j and c_i is above some predefined threshold λ , add the text t_j to cluster c_i
- Although single-pass clustering doesn't explicitly require it, the number of clusters is **indirectly determined** by the value of the threshold λ

K-means

31

- Arguably the most famous and widely used clustering algorithm
- Requires the number of clusters k to be predefined – k clusters, $\mathbf{S} = \{S_1, S_2, \dots, S_k\}$, represented by mean vectors $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_k$
- **K-means** clusters instances $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ by finding the partition \mathbf{S} that **minimizes the within-cluster distances** (maximizing the within-cluster similarities):

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

- **Q:** How to find the optimal clusters (i.e., minimize the above sum of within-cluster distances)?
- **A:** Using iterative optimization

K-means

32

- Algorithm for learning the centroids:

- Randomly** pick k mean vectors $\mu_1, \mu_2, \dots, \mu_k$ in the same space (i.e., of same dimensionality) as instance vectors
 - K-means++** is an extension that **more intelligently** chooses the initial mean vectors

- Iterate the following two steps **until convergence**:

- Assign each instance \mathbf{x}_j to the cluster with the closest mean vector μ_i :

$$S_i^{(t)} = \left\{ \mathbf{x}_j : \|\mathbf{x}_j - \mu_i^{(t)}\|^2 \leq \|\mathbf{x}_j - \mu_j^{(t)}\|^2, \forall j, 1 \leq j \leq k \right\}$$

- For each cluster, **update** the mean vector of a cluster

- Set the mean vector to the mean of the instances in the cluster

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

Outline

33

- Recap of Lecture #8
- Primer on Machine Learning
- Text Classification
- Text Clustering
- [Learning to Rank](#)

Learning to Rank

34

- So far, each IR model was ranking the documents according to a **single** similarity function between the document and the query
 - VSM: **cosine** between the (**sparse**) TF-IDF vectors of the document and query
 - Latent/semantic IR: **cosine** between **dense** semantic vectors
 - Probabilistic IR: $P(d, q \mid \text{relevance})$
 - Language modelling for IR: $P(q \mid d)$
- **Idea:** **Combine** different similarity scores as features of a **supervised model**

$$\vec{f}(d, q) = \begin{pmatrix} VSM_q(d) \\ P(q|d) \\ \text{Jaccard}(q\text{terms}, d\text{terms}) \end{pmatrix}$$

Learning to Rank

35

- Learning to rank is a supervised information retrieval paradigm that
 - Describes instances of document-query pairs (d, q) with a **range of features**
 - **Learns** (with some ML algorithm) the **mapping** between these **features** and **relevance**
- Three different learning-to-rank approaches:
 - 1. Point-wise approach**
 - Classify a **single** document-query (d, q) pair for relevance
 - 2. Pair-wise approach**
 - Classify, for a **pair of documents**, which one is more relevant for the query, i.e., whether $r(d_1, q) > r(d_2, q)$ or $r(d_1, q) < r(d_2, q)$
 - 3. List-wise approach**
 - Classify the **whole ranking** as either correct or wrong

Learning to Rank

36

- **Point-wise** learning to rank
 - Train a supervised classifier that for a given query q classifies each document as relevant or non-relevant
 - Binary classification task: document is either **relevant** or **non-relevant**
 - Training instances:
 - Query-document pairs (q, d) with relevance annotations
- Issues with point-wise learning to rank
 - Do not care about **absolute relevance**, but **relative order of documents** by relevance
 - If pairs (q, d_1) and (q, d_2) are classified as relevant, which document to rank higher?
 - Supervised classifiers usually have confidence/probability scores assigned to predictions
 - Rank d_1 higher than d_2 if the classifier is more confident about relevance of pair (q, d_1)

Learning to Rank

37

- **Pair-wise** learning to rank
 - Train a supervised classifier that for a given query q and two documents d_1 and d_2 predicts which document is more relevant for the query
 - Binary classification task:
 - Class 1: „ d_1 more relevant than d_2 ”
 - Class 2: „ d_1 less relevant than d_2 ”
 - Training instances:
 - Triples (q, d_1, d_2) consisting of queries and document pairs
 - We may need comparison features – compare d_1 and d_2 with respect to q
 - E.g., binary feature: $VSM(q, d_1) > VSM(q, d_2)$
 - Generating gold labels from relevance annotations:
 - For query q we have: $d_1(r)$, $d_2(nr)$, $d_3(r)$, $d_4(nr)$
 - We create the following training instances:
 - $\{(q, d_1, d_2), 1\}$, $\{(q, d_1, d_4), 1\}$, $\{(q, d_2, d_3), 2\}$, $\{(q, d_3, d_4), 1\}$

Learning to Rank

38

- Issues with pair-wise learning to rank
 - If we don't use comparison features (but direct similarities of d_1 and d_2 with q as features), the model **may not generalize well for new queries!**
 - We only obtain **independent pair-wise decisions**
 - **Q:** What if pair-wise decisions are mutually inconsistent?
 - E.g., $(q, d_1, d_2) \rightarrow 1$, $(q, d_2, d_3) \rightarrow 1$, $(q, d_1, d_3) \rightarrow 2$
 - We need an **additional postprocessing step**
 - To turn the **sorted pairs** into a **ranking**, i.e., **partial ordering** into **global ordering**
 - **Inconsistencies** need to be resolved
 - E.g., In a set of conflicting decisions, the one with the lowest classifier confidence is discarded
 - **Another issue:** we effectively treat pairs from the bottom of ranking **same** as those from the top of the ranking (and eval. metrics don't treat them equally!)

Learning to Rank

39

- List-wise ranking approach
 - Instead of learning **decisions for individual documents or pairs of documents**, learn to **classify entire rankings** as correct or wrong
 - Training instances: query and an entire ranking of documents (q, d_1, \dots, d_n)
 - Binary classification task:
 - Class 1: the ranking (q, d_1, \dots, d_n) is **correct**
 - Class 2: the ranking (q, d_1, \dots, d_n) is **incorrect**
 - **Advantage**: optimization criteria for the machine learning algorithm can be the concrete IR evaluation metric we're looking to optimize
- Issues with list-wise approach
 - Entire ranking just **one** training instance
 - Difficult to collect many positive training instances
 - Informative **features for the whole ranking** are difficult to design

Now you...

40

- Know the very basics of machine learning
- Understand supervised text classification
- Know some methods for (unsupervised) text clustering
- Understand how to combine different ranking functions (and other features) in a supervised IR setting – learning to rank