Classification of biomedical texts

Introduction to Machine Learning for NLP

(Slides partially taken from Ulf Leser)

Mario Sänger (WBI, HU Berlin)

saengema@informatik.hu-berlin.de
Studentische Hilfskraft, Forschung und Lehre

In der Arbeitsgruppe "Wissensmanagement in der Bioinformatik" am Institut für Informatik der Humboldt-Universität zu Berlin ist ab 1.6.2020 eine studentische Hilfskraftstelle (40h/Monat, 2 Jahre) zu besetzen. Der/die Stelleninhaber*in unterstützt uns in der Lehre (als Korrektor*in bzw. Tutor*in) und arbeitet an Forschungsprojekten am Lehrstuhl mit. Diese beschäftigen sich mit angewandtem Maschinellem Lernen, biomedizinischen Text Minings, Informationsintegration, der skalierbaren verteilten Datenanalyse, und Bioinformatik für individualisierte Medizin.

Aufgaben
- Erstellung von Softwareprototypen
- Mitarbeit an Forschungsprojekten im Umfeld der biomedizinischen Datenanalyse
- Unterstützung in der Lehre

Voraussetzungen
- Studium der Informatik oder eines angrenzenden Fachs
- Vertiefte Erfahrung im Programmieren
- Erfahrung in der statistischen Datenanalyse und/oder der Bioinformatik
- Grobes Interesse an der angewandten Forschung
- Ein hohes Maß an Eigenmotivation und Kommunikationsfähigkeit / Teamfähigkeit
- Gutes Englisch

https://www.informatik.hu-berlin.de/de/forschung/gebiete/wbi/jobs/shk_haushalt_2004
Supervised text classification

- Given a set $D$ of documents and a set of classes $C$. A classifier is a function $f: D \rightarrow C$

**Problems**
- Finding enough training data
- Finding the best pre-processing (tokenization, case, POS tag set ...)
- Finding the best features
- Finding a good classifier (~ assigning as many docs as possible to their correct class)
Outline

- Machine Learning
  - Overview
  - Challenges and problems

- Classification methods
  - Nearest Neighbour
  - Linear classifiers

- Artificial neuronal networks
  - Motivation
  - Feed forward networks
Machine Learning

Overview

Problems and Challenges
What is Machine Learning (ML)?

- Tom M. Mitchell (1997): “A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$ if its performance at tasks in $T$, as measured by $P$, improves with experience $E$”

Machine Learning (ML)

- Perform a specific tasks **without** using explicit instructions
  - Build a mathematical model based on example data
  - ML models rely on **patterns and inference methods** in order to make predictions or decisions
- Integrates different disciplines

Types of Machine Learning (ML)

• Supervised learning: build a mathematical model based on data sets that contain both the inputs and the desired outputs (labels)
  • Examples: classification and regression

• Unsupervised learning: find structures in unlabelled data
  • Examples: clustering and anomaly detection

• Reinforcement learning: software agents ought to take actions in an environment so as to maximize some (notion of) reward
  • Examples: game AIs and robot navigation
Types of machine learning
Supervised Learning

• Given a set $D$ of documents and a set of classes $C$. A classifier is a function $f: D \rightarrow C$

![Diagram](image)

• Problems
  • Finding *enough* training data
  • Finding the best *pre-processing* (tokenization, case, POS tag set ...)
  • Finding the best features
  • Finding a *good classifier* (~ assigning as many docs as possible to their correct class)

Documents $S$ | Feature vectors | Model training
Supervised Learning

- Given a set $D$ of documents and a set of classes $C$. A classifier is a function $f : D \rightarrow C$

- How do we know?
  - Use a (separate) gold standard data set
  - Use training data in two roles (beare of overfitting)
    - Learning the model
    - Evaluating the model
Problem 1: Overfitting

• Let $S$ be a set of texts with their classes (training data)
• We can easily build a perfect classifier for $S$
  • $f(d) = \{f(d'), \text{ if } \exists d' \in S \text{ with } d' = d; \text{ random otherwise}\}$
  • $f$ is perfect for any doc from $S$
  • But: produces random results for any new document
• Improvement:
  • $f(d) = \{f(d'), \text{ if } \exists d' \in S \text{ with } d' \sim d; \text{ random otherwise}\}$
  • Improvement depends on $|S|$ and definition of “$\sim$”
• Overfitting
  • If the model strongly depends on $S$, $f$ overfits - it will only work well if all future docs are very similar to the docs in $S$
  • You cannot find overfitting when evaluation is performed on $S$ only
Against Overfitting

- **f must generalize**: Capture features that are typical for all docs in D, not only for the docs in S
- But usually we only have S for evaluation ...
  - We need to extrapolate the quality of f to unknown docs

- **Usual method**: Cross-validation
  - Divide S into k disjoint partitions (typical: k=10)
  - Learn model on k-1 partitions and evaluate on the k'th
  - Perform k times, each time evaluating on another partition
  - Estimated quality on new docs = *average performance* over k runs
Cross-validation

- Example $k = 5$:
  - Divide gold standard data into 5 disjoint partitions

<table>
<thead>
<tr>
<th>Fold</th>
<th>Train</th>
<th>Eval</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

avg: 77.56 %
Cross-validation

- For complex models cross-validation can be prohibitively expensive and time-consuming
  - We have to train and evaluate $k$ models!
- Alternative: Split $S$ into a (fixed) disjoint training and validation partition
  - Model selection will be performed based on the validation set performance
  - Both partitions should be “representative” for $S$
    - Same class and feature distribution (e.g. text length)

\[ S \]

\[ \text{training} \quad \text{validation} \]
Problem 2: Information Leakage

- Developing a classifier is an iterative process
  - Define feature space
  - Evaluate performance using cross-validation
  - Perform error analysis, leading to others features / parameters
  - Iterate until satisfied

- In this process, you “sneak” into the data (during error analysis) you later will evaluate on
  - “Information leakage”: information on eval data is used in training

- Solution
  - Reserve a portion $P$ of $S$ for evaluation
  - Perform iterative process only on $S\backslash P$
  - Final evaluation on $P$; no more iterations
Data organization

- In general the following data setup for $S$ is used:
  - **Training set**: train different variants of classifiers (e.g. different methods, pre-processing, feature sets)
  - **Validation set**: validate performance of the different models and choose the best one
  - **Test set**: final evaluation of the model on hold-back data

![Diagram of data organization](attachment:diagram.png)
Evaluation metrics (binary model)

- We can group the predictions of a classifier $f$ according to the gold standard $S$ into **four categories**:

<table>
<thead>
<tr>
<th></th>
<th>Truth: True</th>
<th>Truth: False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier: True</td>
<td>True Positives (TP)</td>
<td>False Positives (FP)</td>
</tr>
<tr>
<td>Classifier: False</td>
<td>False Negatives (FN)</td>
<td>True Negatives (TN)</td>
</tr>
</tbody>
</table>

- **Precision (P)**: $\frac{TP}{(TP+FP)}$
  - Fraction of truly true instances in the „answer“ of $f$

- **Recall (R)**: $\frac{TP}{(TP+FN)}$
  - Fraction of the truly true instances of $S$ found by $f$
Evaluation metrics (binary model)

- We can group the predictions of a classifier $f$ according to the gold standard $S$ into four categories:

<table>
<thead>
<tr>
<th>Classifier: True</th>
<th>Truth: True</th>
<th>Truth: False</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positives (TP)</td>
<td>False Positives (FP)</td>
<td></td>
</tr>
<tr>
<td>False Negatives (FN)</td>
<td>True Negatives (TN)</td>
<td></td>
</tr>
</tbody>
</table>

- What is more important – recall or precision?
  - Go to [https://menti.com](https://menti.com)
  - Enter code 77 55 83
  - Submit your answer
Evaluation metrics (binary model)

- We can group the predictions of a classifier $f$ according to the gold standard $S$ into **four categories**:

<table>
<thead>
<tr>
<th>Classifier: True</th>
<th>Truth: True</th>
<th>Truth: False</th>
</tr>
</thead>
<tbody>
<tr>
<td>True Positives (TP)</td>
<td>True Positives (TP)</td>
<td>False Positives (FP)</td>
</tr>
<tr>
<td>False Negatives (FN)</td>
<td>False Negatives (FN)</td>
<td>True Negatives (TN)</td>
</tr>
</tbody>
</table>

- **F1-Measure**: $2 \times P \times R / (P + R)$
  - Harmonic mean between precision and recall
  - Favours balanced precision / recall values
Evaluation metrics (binary model)

- We can group the predictions of a classifier $f$ according to the gold standard $S$ into four categories:

<table>
<thead>
<tr>
<th></th>
<th>Truth: True</th>
<th>Truth: False</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classifier: True</strong></td>
<td>True Positives (TP)</td>
<td>False Positives (FP)</td>
</tr>
<tr>
<td><strong>Classifier: False</strong></td>
<td>False Negatives (FN)</td>
<td>True Negatives (TN)</td>
</tr>
</tbody>
</table>

- **Accuracy**: $\frac{TP+TN}{(TP+FP+FN+TN)}$
  - Fraction of correctly predicted instances

- Why not always use accuracy?
Evaluation metrics (binary model)

- We can group the predictions of a classifier $f$ according to the gold standard $S$ into **four categories:**

<table>
<thead>
<tr>
<th>Truth: True</th>
<th>Truth: False</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier: True</td>
<td>True Positives (TP)</td>
</tr>
<tr>
<td>Classifier: False</td>
<td>False Negatives (FN)</td>
</tr>
</tbody>
</table>

- **Accuracy:** $\frac{TP+TN}{(TP+FP+FN+TN)}$
  - Fraction of correctly predicted instances

- **Used in problems with balanced sets of** $TP+FN / FP+TN$
  - Don’t use accuracy, if $FP+TN >>> TP+FN$
Classification Methods

Nearest Neighbour
Support Vector Machine
Classification methods

• There are **many classification methods**
  • Bayesian Networks, Graphical models
  • Decision Trees and Random Forests
  • Linear / Logistic Regression
  • Perceptrons, Neural Networks [deep learning]
  • …

• **Effectiveness of classification** depends on problem, algorithm, feature selection method, sample, evaluation, …

• Differences when using **different classification** methods on the same data/representation are often **astonishing small**
Nearest Neighbor Classifiers

• Definition:

Let $S$ be a set of classified documents, $m$ a distance function between any two documents, and $d$ an unclassified document

- A nearest-neighbor (NN) classifier assigns to $d$ the class of the nearest document in $S$ wrt. $m$
- A $k$-nearest-neighbor (kNN) classifier assigns to $d$ the most frequent class among the $k$ nearest documents in $S$ wrt. $m$

• Remarks

- Very simple and effective, but slow
- We may weight the $k$ nearest docs according to their distance to $d$
- We need to take care of multiple docs with the same distance
Illustration – Separating Hyperplanes

Voronoi diagram in 2D-space
(for 1NN)
Properties

• Assumption: Similar docs (in feature space) have the same class; docs in one class are similar
  • Depends a lot on the text representation (bag of words)
  • Depends a lot on the distance function
  • These assumptions can be verified before using a kNN!

• kNN in general more robust than NN

• What do we learn during training of kNN classifier?
Properties

- Assumption: Similar docs (in feature space) have the same class; docs in one class are similar
  - Depends a lot on the text representation (bag of words)
  - Depends a lot on the distance function
  - These assumptions can be verified before using a kNN!
- kNN in general more robust than NN
- Example of lazy learning
  - Actually, there is no learning (only docs)
  - Actually, there is no model (only docs)
Linear Classifiers

• Many common classifiers are (log-)linear classifiers
  • Naïve Bayes, Perceptron, Linear and Logistic Regression, Maximum Entropy, Support Vector Machines

• If applied on a binary classification problem, all these methods somehow compute a hyperplane which (hopefully) separates the two classes
  • Despite similarity, noticeable performance differences exist – Which feature space is used?
  • Which of the infinite number of possible hyperplanes is chosen?
  • How are non-linear-separable data sets handled?
Characteristics of text data

- **High dimensionality**: 100k+ features
- **Sparsity**: Feature values are almost all zero
- **Most documents are very far apart** (i.e., not strictly orthogonal, but only share very common words)
- **Consequence**: Most document sets are well separable
  - This is part of why linear classifiers are quite successful in this domain
- The trick is more of finding the “right” separating hyperplane instead of just finding (any) one
Example: Linear Classifiers – 2D

- Hyperplane separating classes in high dimensional space
- But which?
Support Vector Machine (SVM) - Idea

- SVMs: Hyperplane which maximizes the margin
  - I.e., is as far away from any data point as possible
  - Cast in a linear optimization problem and solved efficiently
  - Classification only depends on support vectors – efficient
    - Points most closest to hyperplane

http://res.cloudinary.com/dyd911kmh/image/upload/f_auto,q_auto:best/v1526288454/index2_ub1uzd.png
Artificial neural networks

Motivation

Feed forward networks
How to find good features?

- Many AI tasks can be solved by designing the **right set of features** to extract and apply a (simple) ML approach
  - However, for many tasks is difficult to know what features should be extracted
  - Can take decades for an entire community of researchers for difficult problems
- **Example**: Identify cars in photographs
  - We know cars have wheels – presence of a wheel maybe a good feature
  - Unfortunately it is hard to describe a wheel in terms of pixels
  - Simple geometric shape – but it’s image may be complicated by shadows falling on it, the sun glaring off the metal parts, ...
• Of course, it is very difficult to extract such high-level features / factors from raw data
  • Need very sophisticated (nearly human-level) understanding of the raw data

• One solution to the problem: representation learning (RL)
  • Use machine learning to discover not only the mapping from representation to output but also the representation itself
  • Representation learning ~ feature learning

• Deep Learning: A RL technique that learns representations that are expressed by simpler representations
  • Build more complex concepts out of simpler concepts
Example: image classification

- Mapping from pixels to an object identity is **very complicated**
  - Instead, use a series of simpler nested mappings

- Every layer builds a **higher abstractions** based on the former layer’s output

- Final layer uses **most abstract** representations to make the prediction

http://www.deeplearningbook.org/contents/intro.html
Artificial neural networks (~ Deep Learning)

- A method for **non-linear** classification
  - Long history - but also forgotten for a long time
    - First works range back to the 1950s / 60s
    - Extremely hyped since about 2005
  - Basic concepts inspired by **biological networks**
    - But, it isn’t the goal to simulate / model these networks

- Today: **state-of-the-art** in machine translation, image recognition, gaming, machine reading, ...
Concepts

- ANNs are composed of **artificial neurons** (~ basic unit)
  - Neurons receive **input signals** from input data or other neurons
  - Input signals will be weighted and aggregated to a scalar value through an **aggregation function** (e.g. weighted sum)
  - Neuron’s output is determined by an **activation function** (e.g. 1 if weighted sum is > 0 or else 0)
Activation functions

- Use activation functions with a **continuous** value range
  - Small changes in the weights and biases cause only a small change in their output
  - Often: activations **saturate** for very large and/or small values

\[
f(x) = \frac{1}{1 + e^{-x}}
\]

\[
tanh(x) = \frac{2}{1 + e^{-2x}} - 1
\]

\[
f(x) = \begin{cases} 
0 & \text{for } x < 0 \\
 x & \text{for } x \geq 0
\end{cases}
\]

[Activation functions](http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/)
Structure of neural networks

- Neurons are organized and stacked in **layers**
  - The neurons of each layer work on the activations from the former layer
  - Each layer learns **more complex** abstractions (≈ decisions) of the input based on the former layer’s abstractions

[Diagram of neural network]

Input layer

Hidden layers

Output layer

http://www.opennn.net/
Example: Text classification

- Let’s suppose, we have a corpus of news articles and we want to perform **automatic categorization** of these articles
  - We want to distinguish articles from **five different categories**: politics, economy, culture, lifestyle, sport

- Assume we have a set $S$ of **labelled examples** $(x_i, y_i)$
  - $x_i$ : TF-IDF vector of text from article $i$ (details next slide)
  - $y_i$ : The gold standard label for article $i$
    - In following we will often refer to the label as **one-hot encoded vector**

<table>
<thead>
<tr>
<th></th>
<th>politics</th>
<th>economy</th>
<th>culture</th>
<th>lifestyle</th>
<th>sport</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y(x_1) = economy$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$y(x_2) = lifestyle$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Example: Text classification

- **Input:** TF-IDF vectors of the articles
  - Let’s say we have a vocabulary with 10,000 distinct token
  - Each component of the vector is modelled as separate input neuron
Example: Text classification

- **Output**: One of the five classes politics, economy, culture, lifestyle, sport
  - Each class gets one dedicated neuron in the output layer
  - We select the output neuron which fires resp. has the highest activation as prediction

![Diagram of a neural network with 10,000 input neurons and five output neurons labeled politics, economy, culture, lifestyle, sport. The network structure shows connections between the input and output layers.](image-url)
Learning a ANN

- Feedforward (and many other) ANN can be efficiently learned using backpropagation

- Idea
  - Initialize weights at random
  - Compute loss function for training samples
  - Adjust weights level wise along the gradient of the loss function
  - Repeat until convergence
  - Trick: Fast and repeated computation of the gradients

- Variation of stochastic gradient descent (SGD)
Example: cost function

- Quadratic cost function (mean squared error)

\[ C(w, b) = \frac{1}{|S|} \sum_i \| y_i - a_i \|^2 \]

- \( w \) and \( b \) all weights and biases of the network
- \( n = |S| \) is number of training examples
- \( a_i \) activation of the neurons in the output layer
Example: cost function

- Quadratic cost function (mean squared error)

$$C(w, b) = \frac{1}{|S|} \sum_{i} \| y_i - a_i \|^2$$

<table>
<thead>
<tr>
<th></th>
<th>politics</th>
<th>economy</th>
<th>culture</th>
<th>lifestyle</th>
<th>sport</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a_1$</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>$y_1 - a_1$</td>
<td>-0.2</td>
<td>0.6</td>
<td>-0.2</td>
<td>-0.1</td>
<td>-0.1</td>
</tr>
<tr>
<td>$|y_1 - a_1|^2$</td>
<td>0.04</td>
<td>0.36</td>
<td>0.04</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.46</td>
</tr>
</tbody>
</table>
Example: cost function

- Quadratic cost function (mean squared error)

\[ C(w, b) = \frac{1}{|S|} \sum_{i} \|y_i - a_i\|^2 \]

- \( w \) and \( b \) all weights and biases of the network
- \( n = |S| \) is number of training examples
- \( a_i \) activation of the neurons in the output layer

- Properties
  - \( C(w, b) \) becomes small \( C(w, b) \approx 0 \) when \( y_i \) is approximately equal to the network output \( a_i \) for all instances
  - In contrast, a large \( C(w, b) \) means that output \( a_i \) is not close to \( y_i \) for many instances
Gradient Descent (sketch)

https://miro.medium.com/max/1005/1*6TVU8yGpXNYDkkpOfnJ6Q.png
Many design choices

- Activation (aggregation) function?
- Number of hidden layers?
- Number of units per hidden layer?
- Connections only between adjacent layers?
- Only "forward" connections?
- Central issue: "Learnability"
  - Different choices lead to different problems
  - Especially back-links increase complexity (and expressiveness)
Organization

Frameworks & Courses

Next steps
NLP frameworks - Python

- **scikit-learn (ML)**
  - [https://scikit-learn.org/stable/](https://scikit-learn.org/stable/)

- **PyTorch (DL)**
  - [https://pytorch.org/](https://pytorch.org/)

- **Tensorflow (DL)**
  - [https://www.tensorflow.org/](https://www.tensorflow.org/)

- **Keras (DL)**
  - [https://keras.io/](https://keras.io/)
NLP frameworks - Java

- **Weka 3 Workbench (ML)**
  - [https://www.cs.waikato.ac.nz/ml/weka/](https://www.cs.waikato.ac.nz/ml/weka/)

- **LibSVM (ML)**
  - [https://www.csie.ntu.edu.tw/~cjlin/libsvm/](https://www.csie.ntu.edu.tw/~cjlin/libsvm/)

- **DeepLearning4J (DL)**
  - [https://deeplearning4j.org/](https://deeplearning4j.org/)
General literature

- Text books:
  - Manning et al.: *Foundations of statistical natural language processing* (Online)
  - Manning et al.: *Introduction to Information Retrieval* (Online)
  - Bishop: *Pattern recognition and machine learning* (Online)
  - Hastie et al.: *The Elements of Statistical Learning* (Online)
  - Goodfellow et al.: *Deep learning* (Online)
Online courses

- **US San Diego – Machine Learning:**
  - [https://www.youtube.com/playlist?list=PL_onPhFckVQhUzcTVgQiC8W2ShZKWIm0s](https://www.youtube.com/playlist?list=PL_onPhFckVQhUzcTVgQiC8W2ShZKWIm0s)

- **Fast.ai – Introduction to machine learning:**

- **Coursera – Machine learning:**
  - [https://de.coursera.org/learn/machine-learning](https://de.coursera.org/learn/machine-learning)

- **Stanford – Natural Language processing with deep learning:**
  - [https://www.youtube.com/playlist?list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z](https://www.youtube.com/playlist?list=PLoROMvodv4rOhcuXMZkNm7j3fVwBBY42z)
Next steps

- I will ....
  - ... send you literature hints and recommendations
  - ... release the training data until end of next week

- You have to ...
  - ... get familiar with your topic
  - ... communicate with your group members
  - ... become acquainted with the framework you want to work
  - ... start to implement your classification pipeline

- Please contact me until 29.05. to discuss your approach
General recommendations

• Experiment with **different variants** of your approach
  • Investigate different **data pre- and post-processing** steps
  • Try different **feature selection** strategies, perform hyperparameter-search, use additional information, .....  

• There are a plethora of tutorials and blog posts in internet
  • Do **not blindly copy** source code – understand what you are do

• **Don’t be afraid** to ask questions
  • Get in touch with me instead of being stuck with a problem for weeks
Thank you for your attention!
Questions?