Machine Learning, part 2

Einführung in die Informationsverarbeitung

Nils Reiter

January 19, 2023

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

Two Parts

Prediction Model

How do we make predictions on data instances?
(e.g., how do we assign a part of speech tag for a word?)

Learning Algorithm

How do we create a prediction model, given annotated data?
(e.g. how do we create rules for assigning a part of speech tag for a word?)

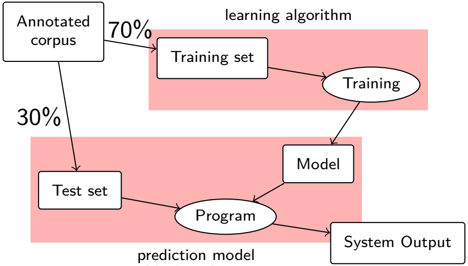
Two Parts

Prediction Model

How do we make predictions on data instances?
(e.g., how do we assign a part of speech tag for a word?)

Learning Algorithm

How do we create a prediction model, given annotated data?
(e.g. how do we create rules for assigning a part of speech tag for a word?)



Decision Tree

- Classification: Based on trees, recursive learning and prediction
- Pros
  - Highly transparent
  - Reasonably fast
  - Dependencies between features can be incorporated into the model
- Cons
  - No pairwise dependencies
  - May lead to overfitting
- Variants exist

Section 2

Weka

Introduction

Ian H. Witten/Eibe Frank (2005). *Data Mining*. 2nd ed. Practical Machine Learning Tools and Techniques. Elsevier

Introduction

Ian H. Witten/Eibe Frank (2005). *Data Mining*. 2nd ed. Practical Machine Learning Tools and Techniques. Elsevier

- Open source, Java
  - `https://www.cs.waikato.ac.nz/ml/weka/`
- Collection of machine learning algorithms
- Playground, GUI, well documented
- Technical limitation: Data sets have to fit in memory
  - = Doesn't work for *really* large data sets

File formats

CSV (Comma-separated values)

- One record per line
- Feature values separated by comma (or semicolon, or tab)

Example

```
Darth, upper, 5, N
Vader, upper, 5, N
war, lower, 3, V
ein, lower, 3, D
Lord, upper, 4, N
der, lower, 3, D
Sith, upper, 4, N
...
```

File formats

ARFF (Attribute relation file format)

Default format used by Weka

Example

```
@RELATION darth-vader
@ATTRIBUTE token STRING
@ATTRIBUTE case {upper,lower}
@ATTRIBUTE length integer
@ATTRIBUTE pos { N,V,D }
@DATA
"Darth", upper, 5, N
"Vader", upper, 5, N
"war", lower, 3, V
"ein", lower, 3, D
...
```

Syntax of ARFF

- `@RELATION name`
  defines a name for this data set
- `@ATTRIBUTE attribute TYPE`
  defines an attribute with the name "attribute" and the data type TYPE

string	Character strings
numeric, real, integer	Numbers
{ nom1, nom2 }	List of nominal values
date	Dates (yyyy-MM-dd'T'HH:mm:ss)

- `@DATA`
  Now the items

Data types

Examples for nominal values

- { red, green, blue }
- { gabi, paula, anna-katharina }
- { one, two, three }
- { true, false }

Data types

Examples for nominal values

- { red, green, blue }
- { gabi, paula, anna-katharina }
- { one, two, three }
- { true, false }

- Conversion: If all strings in a data set are known, they can be converted automatically in nominal values
- Not all classifiers can work with all data types!

Data sets

- `credit-g.arff`
  - Bank credit applications
  - Collected at Hamburg University (before 1993)
    (documentation has 4-digit zip codes ...)
  - Target class: Binary (good/bad)
- `bike-sharing.csv`
  - Bike rentals in Washington D.C.
  - Target: Predict number of rented bikes

demo

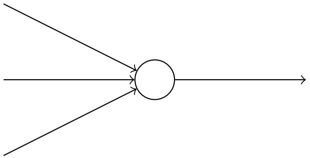
Weka parts

- Preprocess: Remove attributes or instances, rebalance the data set, ...
- Classify: Train and test a classifier
- Cluster: Run a clustering algorithm
- Associate: Investigate associations between features
- Select attributes: Rank attributes according to their importance for a class
- Visualize: Plotting

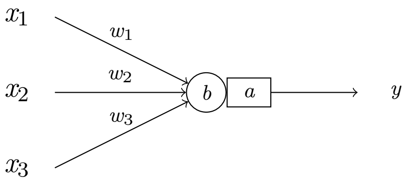
Section 3

Neural Networks

A Neuron



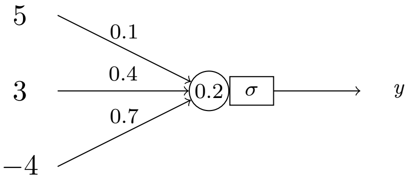
A Neuron



$$\boxed{y = a(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)}$$

A Neuron

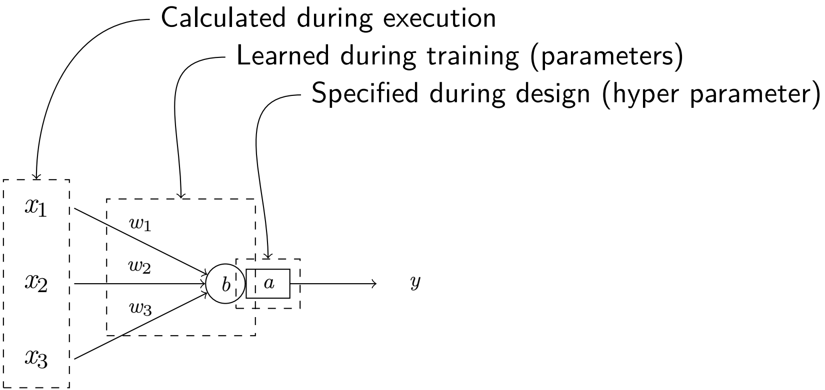
Example



$$
\begin{aligned}
y &= a(w_1 x_1 + w_2 x_2 + w_3 x_3 + b) \\
&= \sigma(0.1 \times 5 + 0.4 \times 3 + 0.7 \times -4 + 0.2) \\
&= \sigma(-0.9) \\
&= 0.289050497374996036536
\end{aligned}
$$

A Neuron

Where do these values come from?



Many Neurons make a Network

Notation

w_{jm}^{kn}: Connection between neuron j in layer k and neuron m in layer n

a: activation function

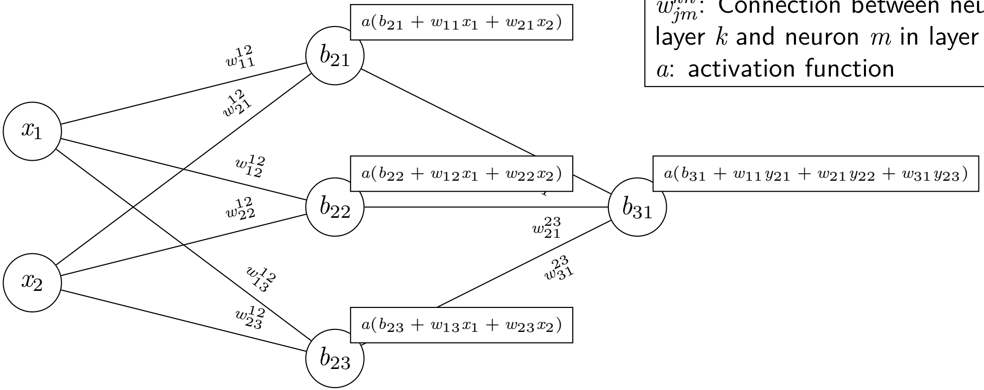


Figure: A simple neural network with 1 hidden layer

Prediction Model

"Forward Pass"

- If we have all the weights, bias terms, numbers of neurons and layers, we can compute the output of the network
  - Conceptually: Applying functions to calculate individual values in sequence

Prediction Model

"Forward Pass"

- If we have all the weights, bias terms, numbers of neurons and layers, we can compute the output of the network
  - Conceptually: Applying functions to calculate individual values in sequence
- Practically, a lot of the computation happens in matrices in parallel
  - Hidden layer
    - Weights: $W_{1,2} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}$
    - Biases $B_2 = (b_{21}, b_{22}, b_{23})$

Prediction Model

"Forward Pass"

- If we have all the weights, bias terms, numbers of neurons and layers, we can compute the output of the network
  - Conceptually: Applying functions to calculate individual values in sequence
- Practically, a lot of the computation happens in matrices in parallel
  - Hidden layer
    - Weights: $W_{1,2} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}$
    - Biases $B_2 = (b_{21}, b_{22}, b_{23})$
- Hidden layer computation: $f_2(X) = \sigma(\underbrace{W_{1,2}^{\intercal} X + B_2}_{\text{matrix operations}})$

Prediction Model

"Forward Pass"

- If we have all the weights, bias terms, numbers of neurons and layers, we can compute the output of the network
  - Conceptually: Applying functions to calculate individual values in sequence
- Practically, a lot of the computation happens in matrices in parallel
  - Hidden layer
    - Weights: $W_{1,2} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}$
    - Biases $B_2 = (b_{21}, b_{22}, b_{23})$
- Hidden layer computation: $f_2(X) = \sigma(\underbrace{W_{1,2}^{\intercal} X + B_2}_{\text{matrix operations}})$
- Deep learning involves a lot of matrix operations
  - GPUs are highly optimized for this
  - Hint: Gaming-GPUs that support CUDA are also usable for deep learning

Feed-Forward Neural Networks

- The above is called a “feedforward neural network”
  - Information is fed only in forward direction

Feed-Forward Neural Networks

- The above is called a “feedforward neural network”
  - Information is fed only in forward direction
- Configuration/design choices
  - Activation function in each layer
  - Number of neurons in each layer
  - Number of layers

Processing Language

- Neural networks operate on numbers
- To process language, we need to preprocess our data

Processing Language

- Neural networks operate on numbers
- To process language, we need to preprocess our data

Word Indices

1. Establish the vocabulary (i.e., the set of all known tokens [in the training corpus])
2. Create a ranking (i.e., count all word types)
3. Decide on a threshold (e.g., the $10\,000$ most frequent words)
4. Replace all words above the threshold by an index number
5. Replace all other words by a special symbol

Processing Language

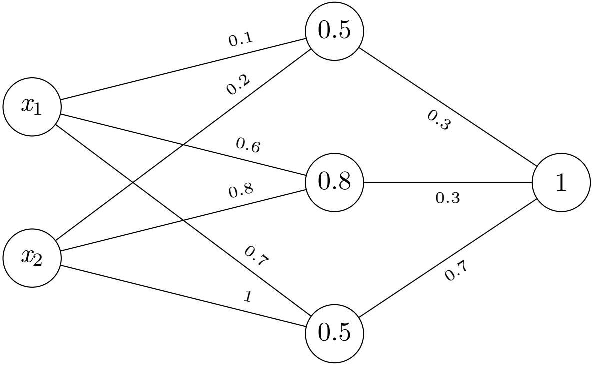
- Neural networks operate on numbers
- To process language, we need to preprocess our data

Word Indices

1. Establish the vocabulary (i.e., the set of all known tokens [in the training corpus])
2. Create a ranking (i.e., count all word types)
3. Decide on a threshold (e.g., the $10\,000$ most frequent words)
4. Replace all words above the threshold by an index number
5. Replace all other words by a special symbol

$\Rightarrow$ "Out of vocabulary" (OOV) words are a challenge for applications

Example



x_1	x_2	y
0	0	0.86169636
1	0	0.87786007
1	1	0.891605
10	10	0.90814614
⋮	⋮	⋮

Figure: Neural network with randomly initialized weights

Learning Algorithm

- We can immediately calculate outcomes (= make predictions), even if all weights are generated randomly
- How do we improve the weights?

Learning Algorithm

- We can immediately calculate outcomes (= make predictions), even if all weights are generated randomly
- How do we improve the weights?
- Gradient Descent
    1. Initialize all weights randomly
    2. Calculate and derive the loss (the ‘wrongness’) of the current weights on the training data
    3. Check if we have found the optimal solution
    4. If not, calculate the direction in which the loss decreases
    5. Go back to 3.

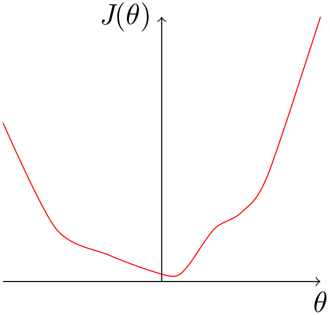
Section 4

Gradient Descent

Loss function: Intuition

- Loss should be as small as possible
- Total loss can be calculated for given parameters θ
  - θ is a vector containing all weights and bias terms in the network
- Idea:
  - We change θ until we find the minimum of the function
  - We use the derivative to find out if we are in a minimum
  - The derivative also tells us in which direction to go

Loss function: Intuition



Loss and Hypothesis Function

- Hypothesis function h
  - Calculates outcomes, given feature values x
  - Done by the neural network
- Loss function J
  - Calculates 'wrongness' of h, given parameter values θ (and a data set)
  - In reality, θ represents millions of parameters

Loss function: Definition

- Different loss function are in use
- Which one to use depends on our aims

Binary Cross-Entropy Loss

- Loss function used for binary classification problems
- Assumption: Output of the network is in $[0;1]$, 0/1 representing the two classes

$$J(\theta) = -\frac{1}{m}\sum_{i=0}^{m} y_i \log h_\theta(x_i) + (1-y_i)\log(1-h_\theta(x_i))$$

Loss function: Definition

Binary Cross-Entropy Loss

$$J(\theta) =$$

m Number of training instances

y_i The true outcomes (from training data)

x_i The input values

Loss function: Definition

Binary Cross-Entropy Loss

$$J(\theta) = -\frac{1}{m}\sum_{i=0}^{m}$$

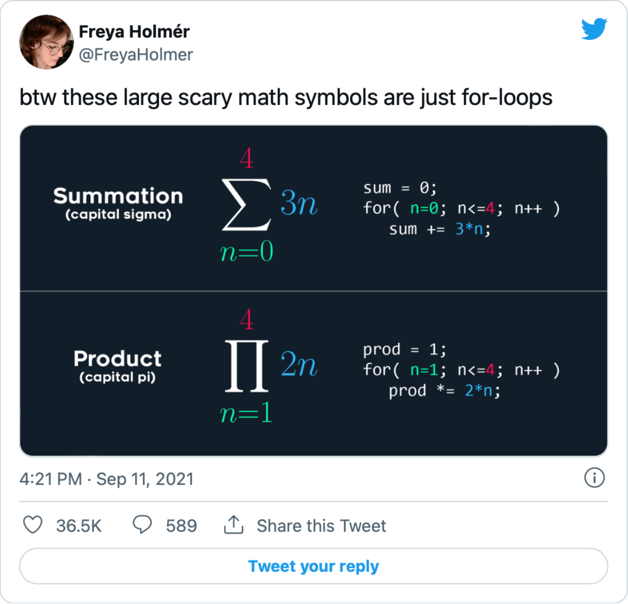
- m Number of training instances
- y_i The true outcomes (from training data)
- x_i The input values

Loss function: Definition

Binary Cross-Entropy Loss

$$J(\theta) = -\frac{1}{m}\sum_{i=0}^{m}$$

- m Number of training instances
- y_i The true outcomes (from training data)
- x_i The input values



Loss function: Definition

Binary Cross-Entropy Loss

$$J(\theta) = -\frac{1}{m}\sum_{i=0}^{m} \underbrace{y_i \log_2 h_\theta(x_i)}_{0 \text{ iff } y_i=0}$$

- m Number of training instances
- y_i The true outcomes (from training data)
- x_i The input values

Loss function: Definition

Binary Cross-Entropy Loss

$$J(\theta) = -\frac{1}{m}\sum_{i=0}^{m} \underbrace{y_i \log_2 h_\theta(x_i)}_{0 \text{ iff } y_i=0} + \underbrace{(1-y_i)\log_2(1-h_\theta(x_i))}_{0 \text{ iff } y_i=1}$$

- m Number of training instances
- y_i The true outcomes (from training data)
- x_i The input values

Section 5

Summary

Summary

- Weka
  - Open source java software for classical machine learning
  - Preprocessing, classification
  - Sentiment analysis, bag of words
- Neural networks
  - Consist of neurons, which represent mathematical functions
  - Prediction model: Calculation by pipelining functions
  - Learning algorithm: Gradient descent