

Natural Language Processing 2: Deep learning, transformer models

HS Sprachtechnologie für eine bessere Welt (Winter term 2022/23)

Nils Reiter,
`nils.reiter@uni-koeln.de`

November 8, 2022

07.-11. November 2022



PUBLIC CLIMATE SCHOOL





Was ist die PCS?

Die Public Climate School ist eine Aktionswoche zu Themen rund um die Klimakrise – voller Vorträge, Workshops & Diskussionsrunden von und mit Wissenschaftler:innen, Expert:innen und Dozierenden verschiedenster Fachgebiete.

Sie hat zum Ziel, Bewusstsein und Aufklärung für die herausragende Bedeutung der Klimakrise für eine lebenswerte Zukunft auf dieser Erde zu schaffen und Bildung für nachhaltige Entwicklung sowie Klimabildung für alle zu ermöglichen.





PCS steht für...

Public

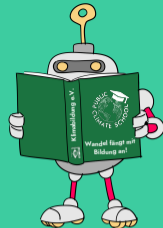
weil sie für alle Menschen offen ist – explizit nicht nur für Studierende.

Climate

weil Themen rund um die Klimakrise in den Vordergrund gerückt werden.
Wir versuchen Klimabildung möglichst vielen Menschen zugänglich zu machen.

School

weil die Klimakrise in allen Fach- und Lebensbereichen relevant ist und damit Schulen, Hochschulen und Unis in die Verantwortung bringt, das Wissen in die breite Gesellschaft zu tragen.





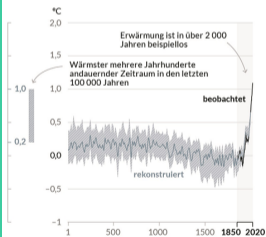
Warum?



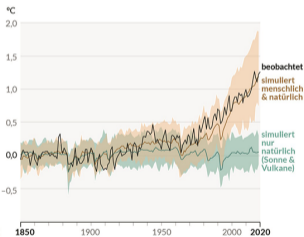
Der Einfluss des Menschen hat das Klima in einem Maße erwärmt, wie es seit mindestens 2 000 Jahren nicht mehr der Fall war

Änderungen der globalen Oberflächentemperatur gegenüber 1850-1900

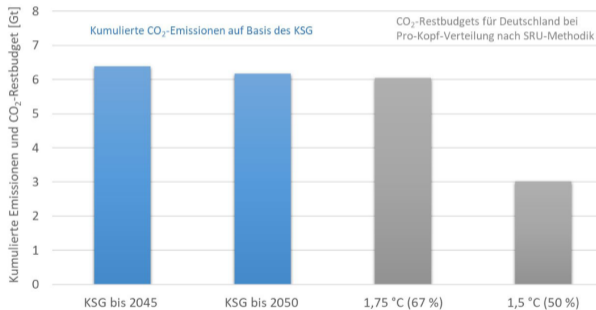
(a) Änderung der globalen Oberflächentemperatur (dekadisches Mittel) wie rekonstruiert (1-2000) und beobachtet (1850-2020)



(b) Änderung der globalen Oberflächentemperatur (Jahresmittel) wie beobachtet und auf Basis menschlicher & natürlicher beziehungsweise nur natürlicher Faktoren simuliert (jeweils 1850-2020)



Kumulierte CO₂-Emissionen auf Basis des Klimaschutzgesetzes (KSG) im Vergleich mit Pro-Kopf Restbudgets ab 2022



Der Klimawandel ist menschengemacht!

Wir sind aktuell nicht auf dem 1,5 Grad-Pfad!

GEFÖRDERT VOM



Bundesministerium
für Bildung
und Forschung



Was ist die PCS? Schulprogramm ▾ FAQ Downloads & Presse Rückblick ▾

Wandel fängt mit Bildung an!

Bei der Public Climate School handelt es sich um ein digitales Bildungsprogramm, das von Studierenden der Fridays For Future Bewegung koordiniert und von vielen Wissenschaftler:innen, Expert:innen, Schüler:innen und Lehrer:innen mitgestaltet wird. Sie hat zum Ziel, Bewusstsein und Aufklärung für die herausragende Bedeutung der Klimakrise für eine lebenswerte Zukunft auf dieser Erde zu schaffen und Klimabildung für alle zu ermöglichen.

Die nächste Public Climate School ist vom **07.-11.11.2022**
Jeder Tag steht unter einem bestimmten Thema:

Schulprogramm

Uniprogramm

Klima-TV



info@publicclimateschool.de

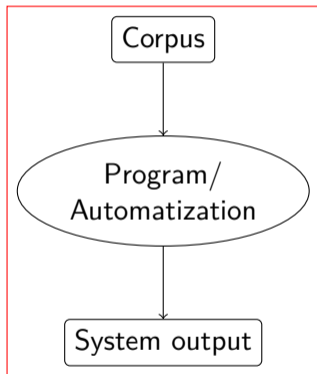


www.publicclimateschool.de



@studentsforfuture_germany

Automatization



Today

1. Decision Trees (cont'd)
2. Neural Networks
3. Gradient Descent
4. Word2Vec

Next week

1. Encoder-Attention-Decoder
2. BERT



Decision Trees

Decision Trees

Prediction Model – Toy Example



- ▶ What are the instances?
 - ▶ Situations we are in (this is not really automatisable)
- ▶ What are the features?
 - ▶ Consciousness
 - ▶ Clothing situation
 - ▶ Promises made
 - ▶ Whether we are driving
 - ▶ ...

Decision Trees

Prediction Model

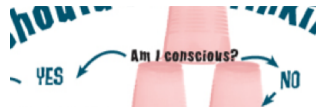
- ▶ Each non-leaf node in the tree represents one feature
- ▶ Each leaf node represents a class label
- ▶ Each branch at this node represents one possible feature value
 - ▶ Number of branches = $|v(f_i)|$ (number of possible values)



Decision Trees

Prediction Model

- ▶ Each non-leaf node in the tree represents one feature
- ▶ Each leaf node represents a class label
- ▶ Each branch at this node represents one possible feature value
 - ▶ Number of branches = $|v(f_i)|$ (number of possible values)
- ▶ Make a prediction for x :
 1. Start at root node
 2. If it's a leaf node
 - ▶ assign the class label
 3. Else
 - ▶ Check node which feature is to be tested (f_i)
 - ▶ Extract $f_i(x)$
 - ▶ Follow corresponding branch
 - ▶ Go to 2



Decision Trees

Learning Algorithm

- ▶ Core idea: The tree represents splits of the training data
 1. Start with the full data set D_{train} as D
 2. If D only contains members of a single class:
 - ▶ Done.
 3. Else:
 - ▶ Select a feature f_i
 - ▶ Extract feature values of all instances in D
 - ▶ Split the data set according to f_i : $D = D_v \cup D_w \cup D_u \dots$
 - ▶ Go back to 2

- ▶ Remaining question: How to select features?

Decision Trees

Feature Selection

- ▶ What is a good feature?
 - ▶ One that maximizes homogeneity in the split data set

Decision Trees

Feature Selection

- ▶ What is a good feature?
 - ▶ One that maximizes homogeneity in the split data set
- ▶ “Homogeneity”
 - ▶ Increase
 $\{\spadesuit\spadesuit\spadesuit\heartsuit\} = \{\heartsuit\} \cup \{\spadesuit\spadesuit\spadesuit\}$
 - ▶ No increase
 $\{\spadesuit\spadesuit\spadesuit\heartsuit\} = \{\spadesuit\} \cup \{\spadesuit\spadesuit\heartsuit\}$

Decision Trees

Feature Selection

- ▶ What is a good feature?
 - ▶ One that maximizes homogeneity in the split data set

- ▶ “Homogeneity”

- ▶ Increase

$$\{\spadesuit\spadesuit\spadesuit\heartsuit\} = \{\heartsuit\} \cup \{\spadesuit\spadesuit\spadesuit\} \leftarrow \text{better split!}$$

- ▶ No increase

$$\{\spadesuit\spadesuit\spadesuit\heartsuit\} = \{\spadesuit\} \cup \{\spadesuit\spadesuit\heartsuit\}$$

- ▶ Homogeneity: Entropy/information

Shannon (1948)

Decision Trees

Feature Selection

- ▶ What is a good feature?
 - ▶ One that maximizes homogeneity in the split data set
- ▶ “Homogeneity”
 - ▶ Increase
 - $\{\spadesuit\spadesuit\spadesuit\heartsuit\} = \{\heartsuit\} \cup \{\spadesuit\spadesuit\spadesuit\} \leftarrow \text{better split!}$
 - ▶ No increase
 - $\{\spadesuit\spadesuit\spadesuit\heartsuit\} = \{\spadesuit\} \cup \{\spadesuit\spadesuit\heartsuit\}$
- ▶ Homogeneity: Entropy/information
- ▶ Rule: Always select the feature with the highest *information gain* (IG)
 - ▶ (= the highest reduction in entropy = the highest increase in homogeneity)

Shannon (1948)

Decision Trees

Entropy (Shannon, 1948)

$$H(X) = - \sum_{i=1}^n p(x_i) \log_b p(x_i)$$

number of classes present in X
 relative frequency of the class
 entropy

Examples (with $b = 2$)

▶ $H(\{\spadesuit\spadesuit\spadesuit\spadesuit\}) = -\frac{4}{4} \log_2 \frac{4}{4} = 0$

▶ $H(\{\spadesuit\spadesuit\spadesuit\heartsuit\}) = - \left(\underbrace{\frac{3}{4} \log_2 \frac{3}{4}}_{\spadesuit} + \underbrace{\frac{1}{4} \log_2 \frac{1}{4}}_{\heartsuit} \right) = 0.562$

▶ $H(\{\spadesuit\spadesuit\heartsuit\heartsuit\}) = \dots = 0.693$

aaabbc
↑ ↑ ← 100
1 10

Decision Trees

Entropy (Shannon, 1948)

$$H(X) = - \sum_{i=1}^n p(x_i) \log_b p(x_i)$$

entropy \swarrow
 n \swarrow number of classes present in X
 $p(x_i)$ \swarrow relative frequency of the class

$$\log_2(x) = y$$

Examples (with $b = 2$)

▶ $H(\{\spadesuit\spadesuit\spadesuit\spadesuit\}) = -\frac{4}{4} \log_2 \frac{4}{4} = 0$

▶ $H(\{\spadesuit\spadesuit\spadesuit\heartsuit\}) = - \left(\underbrace{\frac{3}{4} \log_2 \frac{3}{4}}_{\spadesuit} + \underbrace{\frac{1}{4} \log_2 \frac{1}{4}}_{\heartsuit} \right) = 0.562$

▶ $H(\{\spadesuit\spadesuit\heartsuit\heartsuit\}) = \dots = 0.693$

Decision Trees

Feature Selection (2)



$$\begin{aligned}
 H(\{\spadesuit\spadesuit\spadesuit\heartsuit\}) &= H([3, 1]) \\
 &= 0.562 \\
 H(\{\heartsuit\}) &= H([1]) = 0 \\
 H(\{\spadesuit\spadesuit\spadesuit\}) &= H([3]) \\
 &= 0
 \end{aligned}$$



$$\begin{aligned}
 H(\{\spadesuit\spadesuit\spadesuit\heartsuit\}) &= H([3, 1]) \\
 &= 0.562 \\
 H(\{\spadesuit\}) &= H([1]) = 0 \\
 H(\{\spadesuit\spadesuit\heartsuit\}) &= H([2, 1]) \\
 &= 0.637
 \end{aligned}$$

Decision Trees

Feature Selection (3)



$$H(\{\spadesuit\spadesuit\spadesuit\heartsuit\}) = 0.562$$

$$H(\{\heartsuit\}) = 0$$

$$H(\{\spadesuit\spadesuit\spadesuit\}) = 0$$



$$H(\{\spadesuit\spadesuit\spadesuit\heartsuit\}) = 0.562$$

$$H(\{\spadesuit\}) = 0$$

$$H(\{\spadesuit\spadesuit\heartsuit\}) = 0.637$$

$$IG(f_1) = H(\{\spadesuit\spadesuit\spadesuit\heartsuit\}) - \varnothing(H(\{\heartsuit\}), H(\{\spadesuit\spadesuit\spadesuit\}))$$

$$= 0.562 - 0 = 0.562$$

$$IG(f_2) = H(\{\spadesuit\spadesuit\spadesuit\heartsuit\}) - \varnothing(H(\{\spadesuit\}), H(\{\spadesuit\spadesuit\heartsuit\}))$$

$$= 0.562 - \left(\frac{3}{4} \cdot 0.637 + \frac{1}{4} \cdot 0\right)$$

Decision Trees

Summary

- ▶ Classification algorithm
- ▶ Built around trees, recursive learning and prediction
- ▶ Detailed example in the appendix
- ▶ Pros
 - ▶ Highly transparent
 - ▶ Reasonably fast
 - ▶ Dependencies between features can be incorporated into the model
- ▶ Cons
 - ▶ Often not very good
 - ▶ No pairwise dependencies
 - ▶ May lead to overfitting
 - ▶ Only nominal features
- ▶ Variants exist



Task Types and Methods

Task	Method(s)
Classification	Decision tree, support vector machine, neural network , naïve Bayes, k -nearest neighbors, ...
Clustering	k -means, affinity propagation, ...

Task Types and Methods

Task	Method(s)
Classification	Decision tree, support vector machine, neural network , naïve Bayes, k -nearest neighbors, ...
Clustering	k -means, affinity propagation, ...
Ranking	Ranking SVM, PageRank, ...
Sequence labeling	Hidden Markov models, conditional random field, neural networks ...
Segmentation	...

Supervised Machine Learning

Two parts to understand

Prediction Model

How do we make predictions on data instances?

(e.g., how do we assign a part of speech tag to a (unlabeled) word?)

Learning Algorithm

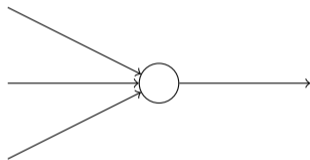
How do we create a prediction model, given annotated data?

(e.g., how do we create rules for assigning a part of speech tag to a word?)

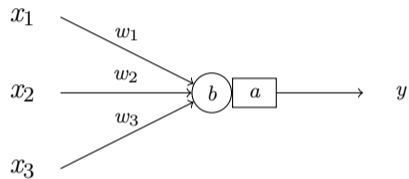
Section 2

Neural Networks

A Neuron



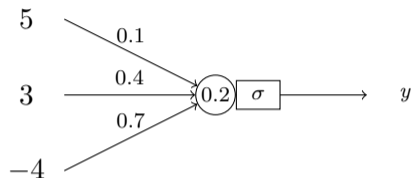
A Neuron



$$y = a(w_1x_1 + w_2x_2 + w_3x_3 + b)$$

A Neuron

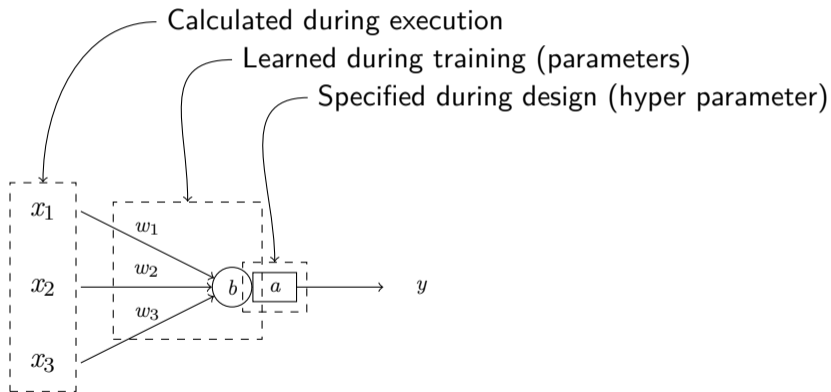
Example



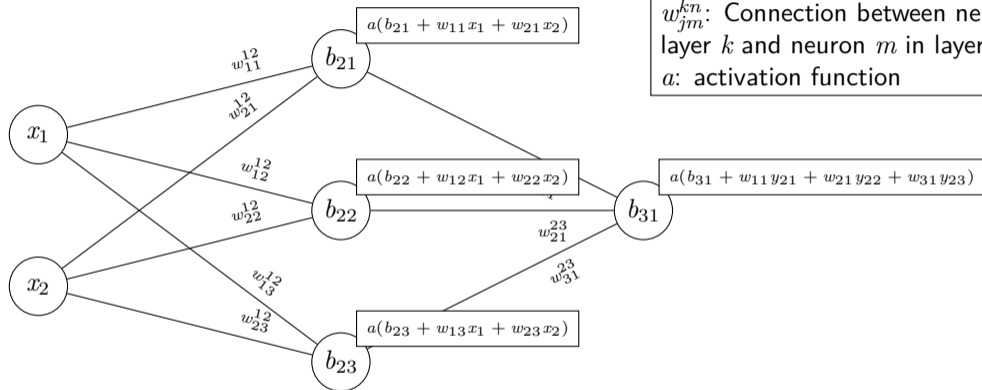
$$\begin{aligned}y &= a(w_1x_1 + w_2x_2 + w_3x_3 + b) \\ &= \sigma(0.1 \times 5 + 0.4 \times 3 + 0.7 \times -4 + 0.2) \\ &= \sigma(-0.9) \\ &= 0.2890504973749960365369\end{aligned}$$

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: $f_2(X) = \sigma(\underbrace{W_{1,2}^T X + B_2}_{\text{matrix operations}})$

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: $f_2(X) = \sigma(\underbrace{W_{1,2}^T X + B_2}_{\text{matrix operations}})$
- ▶ 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

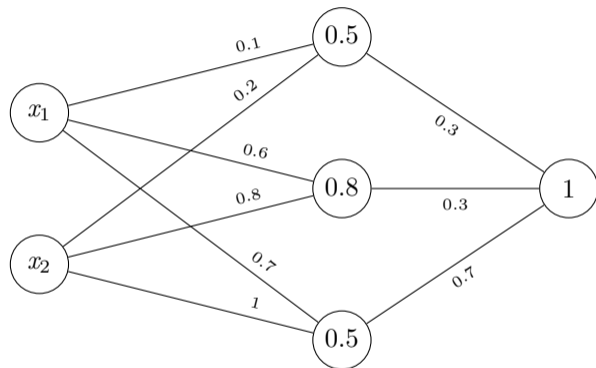
Processing Language

- ▶ 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
- ⇒ “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
\vdots	\vdots	\vdots

Figure: Neural network with randomly initialized weights

```
5 import numpy as np
6 from tensorflow import keras
7 from tensorflow.keras import layers
8
9 # setup the model architecture
10 model = keras.Sequential()
11 model.add(layers.InputLayer(input_shape=(2,)))
12 model.add(layers.Dense(3, activation="sigmoid"))
13 model.add(layers.Dense(1, activation="sigmoid"))
14
15 model.compile() # compile it
16
17 w1 = [ # weights between neurons
18     np.array([[0.1,0.6,0.7],[0.2,0.8,1]]),
19     # bias terms
20     np.array([0.5,0.8,0.5]) ]
21
22 w2 = [ # weights between neurons
23     np.array([[0.3],[0.3],[0.7]]),
24     # bias terms
25     np.array([1]) ]
26 model.layers[0].set_weights(w1)
27 model.layers[1].set_weights(w2)
```

Neural network with manually
specified weights as above

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.

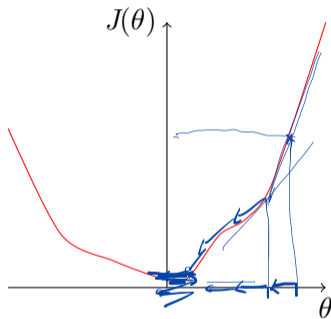
Section 3

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_{θ}
 - ▶ 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

 $h(x)$ $J(\theta)$

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

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

y	$h(x)$	lit	$1-h(x)$
0	0.1	+	
0	0.9	-	
1	0.1	-	0.9
1	0.9	+	0.1

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



Freya Holmér
@FreyaHolmer



btw these large scary math symbols are just for-loops

Summation
(capital sigma)

$$\sum_{n=0}^4 3n$$

```
sum = 0;
for( n=0; n<=4; n++ )
  sum += 3*n;
```

Product
(capital pi)

$$\prod_{n=1}^4 2n$$

```
prod = 1;
for( n=1; n<=4; n++ )
  prod *= 2*n;
```

4:21 PM · Sep 11, 2021



36.5K



589



Share this Tweet

[Tweet your reply](#)

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

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

Anschalter

m Number of training instances

y_i The true outcomes (from training data)

x_i The input values

Summary and Remarks

- ▶ Neural network consists of layers of neurons
- ▶ Training goal: Find weights, such that the training instances are correctly predicted
- ▶ Training method: Gradient descent
- ▶ Training does not have to be completed in one go
 - ▶ Pausing at any time is possible
 - ▶ Training can continue with a different data set

Section 5

Summary

Summary

Neural networks

- ▶ Consist of neurons, which combine values from previous neurons
- ▶ Matrix computation
- ▶ Can 'learn' any relation between input and output





Gradient descent

- ▶ Basic form to train a neural network
- ▶ Start with random weights, then iteratively improve
- ▶ Loss: Quantification of the wrongness of the current weights

Word2Vec

- ▶ Take learned weights as vector representation for input
- ▶ Allows "semantic calculation"

References I

-  Jurafsky, Dan/James H. Martin (2019). *Speech and Language Processing*. 3rd ed. Draft of October 16, 2019. Prentice Hall.
-  Mikolov, Tomáš/Kai Chen/Greg Corrado/Jeffrey Dean (2013). “Efficient Estimation of Word Representations in Vector Space”. In: *arXiv cs.CL*. URL: <https://arxiv.org/pdf/1301.3781.pdf>.
-  Mikolov, Tomáš/Ilya Sutskever/Kai Chen/Greg S Corrado/Jeff Dean (2013). “Distributed Representations of Words and Phrases and their Compositionality”. In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges/L. Bottou/M. Welling/Z. Ghahramani/K. Q. Weinberger. Curran Associates, Inc., pp. 3111–3119. URL: <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>.
-  Shannon, Claude E. (1948). “A mathematical theory of communication”. In: *The Bell System Technical Journal* 27.3, pp. 379–423.