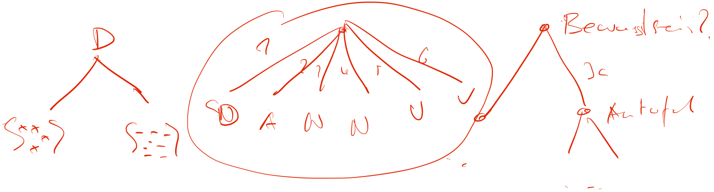
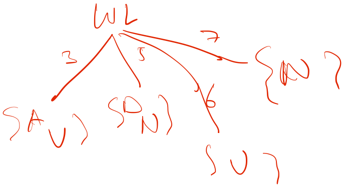
# Recap

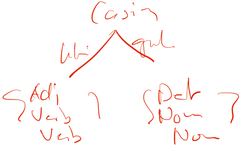


## Decision Tree

- Classification algorithm
- Built around trees, recursive learning and prediction
- Pros
  - Highly transparent (if the number of features is not very large)
  - Reasonably fast
  - Dependencies between features can be incorporated into the model
- Cons
  - No pairwise dependencies
  - May lead to overfitting
  - Only nominal features
- Variants exist

| Id | WL | Casing | WA |
|---|---|---|---|
| 1 | 5. | Groß | Det |
| 2 | 3. | klein | Adj |
| 3 | 7. | groß | Nom |
| 4 | 5 | groß | Nom |
| 5 | 3 | klein | Verb |
| 6 | 6. | klein | Verb |





UNIVERSITÄT
ZU KÖLN

# Machine Learning, Part 2 (= Deep Learning)

## Einführung in die Informationsverarbeitung

Nils Reiter

November 9, 2023

INSTITUT FÜR
DIGITAL HUMANITIES
UNIVERSITÄT ZU KÖLN

# Introduction: Neural Networks

- The machinery behind ChatGPT & co
- Old idea, but rediscovered in late 00s
- Basic idea: A (mathematical) function that takes input, does some computation, produces output
    - E.g.: $f(\underline{x_1}, \underline{x_2}, \underline{x_3}) \boxed{= 0.5x_1 - 0.7x_2 + 0.1x_3 + 0.2}$

# Intuition

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 2 | 2 | 3 |
| 2 | 4 | 8 |
| 3 | 4 | 8,2 |
| $\vdots$ |  | $\vdots$ |

Allowed operations:
Addition, Subtraction,
Multiplication

$$y = (x_1 + x_2 + 2)/2$$

$$y = 2 + \frac{2}{2} = x_1 + \frac{x_2}{2} = x_1 + 0,5 x_2$$

$$y = 0,5 \cdot x_1 + x_2$$

$$y = \_\_ x_1 + \_\_ x_2 + \_\_$$

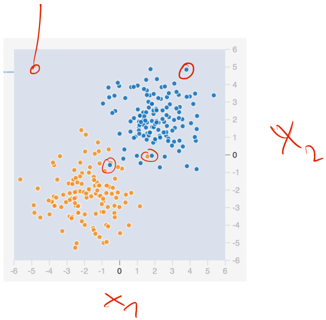
$$y = 0,2 x_1 + 2,5 x_2 - 2,4$$

# More complex example

$x_1 = 4 \quad x_2 = 5$

$$y = \overset{7}{\boxed{w_1}}\overset{4}{x_1} + \overset{-150}{\boxed{w_2}}\overset{5}{x_2} + \underset{\text{gr. ?}}{\boxed{b}}$$

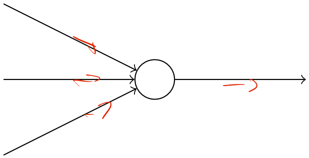
$= \text{blau}$



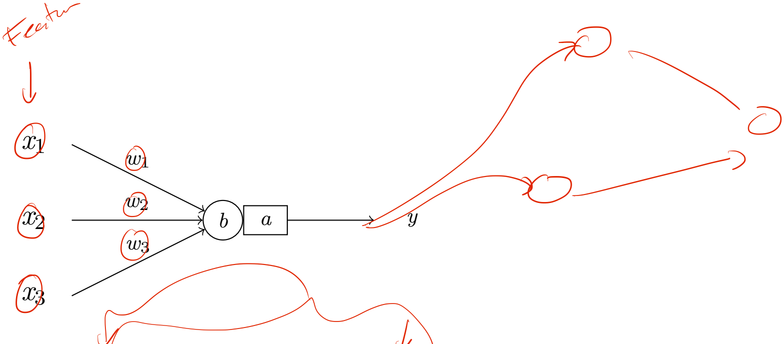
- We have collected data for a binary classification (indicated by colors)
- Our goal: Estimate a function that takes in $x_1$ and $x_2$ values and outputs probabilities for classes orange and blue
  - E.g., $x_1$: word length, $x_2$: position in a sentence, output: is the word a noun?

Neural Network Playground (runs in your browser!)

# A Neuron



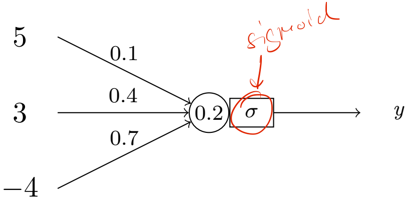
# A Neuron

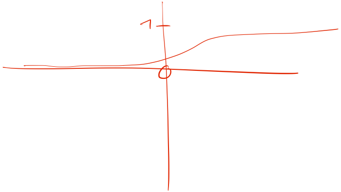


$$\boxed{y = a(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)}$$

# A Neuron

## Example

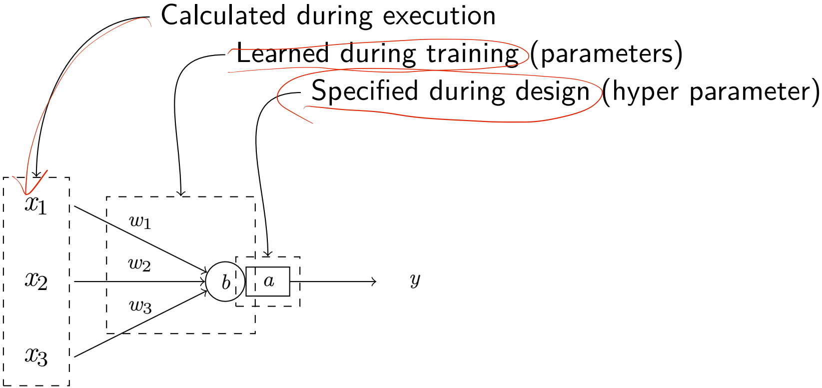




$$
\begin{aligned}
y &= a(w_1x_1 + w_2x_2 + w_3x_3 + \boxed{b}) \\
&= \sigma(0.1 \times 5 + 0.4 \times 3 + 0.7 \times -4 + 0.2) \\
&= \sigma(-0.9) \\
&= 0.289050497374996036536 9
\end{aligned}
$$

# A Neuron

## Where do these values come from?



Calculated during execution

Learned during training (parameters)

Specified during design (hyper parameter)

# Many Neurons make a Network

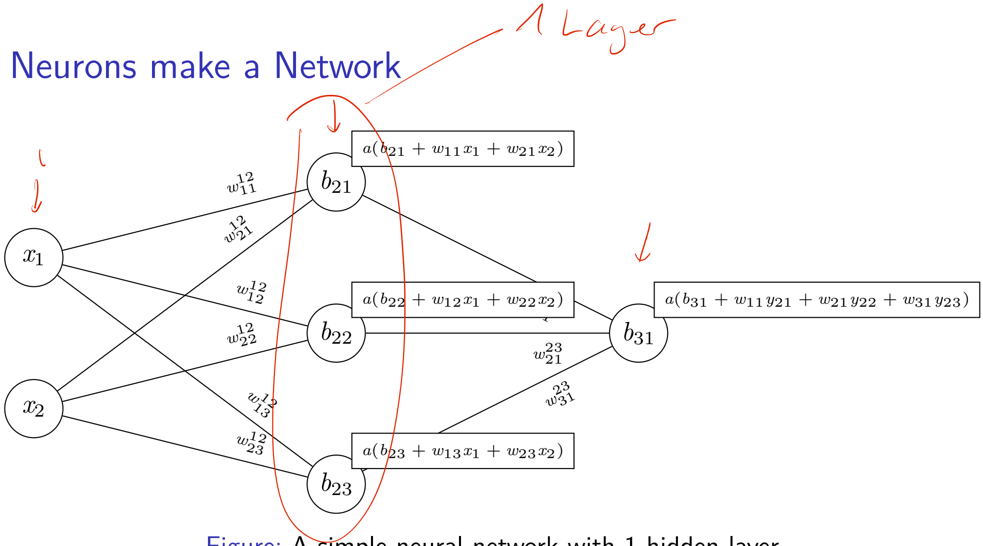


Figure: A simple neural network with 1 hidden layer

# 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

# Learning Algorithm

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

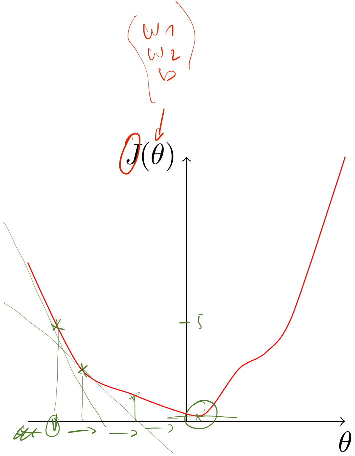
# Learning Algorithm

- 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.

# Loss function: Intuition

- Loss should be as small as possible
- Total loss can be calculated for given parameters $\theta$
  - $\theta$ is a vector containing all weights and bias terms in the network
- Idea:
  - We change $\theta$ 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

$h(15, 17)$ orange

$=$ ~~blau~~

$h(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$

$h(23, 75) =$ blau

| | $x_1$ | $x_2$ | Hund? |
|---|---|---|---|
| Geb | 15 | 17 | ✓ → orange |
| 23.02.23 | 23 | 75 | — → blau |

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

# 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

# Word2Vec

## Literature basis

### Two very influential papers by Mikolov et al.

Tomáš Mikolov/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

Tomáš Mikolov/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

### Software package

word2vec – `https://github.com/tmikolov/word2vec`
(other implementations do exist)

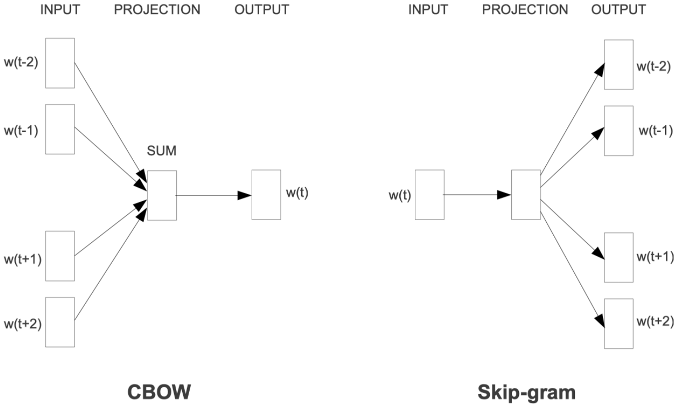
# Word2Vec

## Core Idea

- Define a classification task for which we have huge training data sets
  - Given a word, predict predict possible context words
  - Training data: Any text collection (e.g., Wikipedia)
- Train a neural network
- Extract learned weights and use as embeddings ☑

# Word2Vec

## Two tasks



### Continuous Bag of Words (CBOW)

Context words used to predict a single word

### Skip-Gram

One word used to predict its context

# Word2Vec

## Skip-gram

- Context: $\pm 2$ words around target word $t$

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
              c1         c2 t       c3   c4
```

# Word2Vec

## Skip-gram

[0.7, 0.5 ...]

- Context: $\pm 2$ words around target word $t$

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
              c1         c2 t       c3   c4
```

- Classifier:
  - Predict for $(t, c)$ wether $c$ are *really* context words for $t$
  - In other words: Probability of $\vec{t}$ and $\vec{c}$ being positive examples: $p(+|\vec{t}, \vec{c})$

# Word2Vec

## Skip-gram

- Context: $\pm 2$ words around target word $t$

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
              c1         c2 t       c3  c4
```

- Classifier:
  - Predict for $(t, c)$ wether $c$ are *really* context words for $t$
  - In other words: Probability of $\vec{t}$ and $\vec{c}$ being positive examples: $p(+|\vec{t}, \vec{c})$
- Probability is based on similarity
  - "a word is likely to occur near the target if its embedding is similar to the target embedding"

    Jurafsky/Martin (JM19, 112)

# Word2Vec

## Skip-gram

- Context: $\pm 2$ words around target word $t$

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
              c1         c2 t       c3   c4
```

- Classifier:
  - Predict for $(t, c)$ wether $c$ are *really* context words for $t$
  - In other words: Probability of $\vec{t}$ and $\vec{c}$ being positive examples: $p(+|\vec{t}, \vec{c})$
- Probability is based on similarity
  - "a word is likely to occur near the target if its embedding is similar to the target embedding"

    Jurafsky/Martin (JM19, 112)
  - Similarity of vectors? Cosine!

# Word2Vec

## Skip-gram

- Context: $\pm 2$ words around target word $t$

```
... lemon, a [tablespoon of apricot jam, a] pinch ...
              c1         c2 t       c3   c4
```

- Classifier:
  - Predict for $(t, c)$ wether $c$ are *really* context words for $t$
  - In other words: Probability of $\vec{t}$ and $\vec{c}$ being positive examples: $p(+|\vec{t}, \vec{c})$
- Probability is based on similarity
  - “a word is likely to occur near the target if its embedding is similar to the target embedding”

    Jurafsky/Martin (JM19, 112)
  - Similarity of vectors? Cosine!
  - Similarity → probability? Mathematical conversion

# Summary

- Neural Networks
  - Many individual neurons in combination
  - During training weights will be adapted, until best approximation to training data is reached
  - Prediction: Application of weights, i.e., multiplication and addition
- Word2Vec
  - Use a neural network to predict wether two words are really appearing together in texts
  - Extract learned weights as embeddings
  - Leads to embeddings that are similar, if the words appear in similar contexts
    - E.g., Berlin and Paris appear in the contexts of country capital

# 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`.