Dimension Reduction & Introduction to Neural Networks

LIAD MAGEN

Agenda

> Introduction to Neural Networks
> Word2Vec

Review – Approaching ML Project

- Explore the data (always look at the data)
  - Formulate questions – and find their answers
  - Plot the data
- Plan experiments
  - Decide which features should be used
  - Decide on the models
- Perform Experiments
- Optimize Hyperparameters
- Use the test set to choose the best model

Review – Approaching ML Project

- Explore the data (always look at the data)
  - Formulate questions – and find their answers
  - Plot the data
- Plan experiments
  - Decide which features should be used
  - Decide on the models
- Perform Experiments
- Optimize Hyperparameters
- Use the test set to choose the best model

How do we estimate the best model?

> How do we choose a score?
> Is the plain score enough?
> What if we have multiple scores?

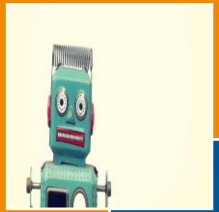
The Big Picture

Supervised

- Decision Tree
- Random Forest
- Logistic Regression
- Naïve Bayes
- K-Nearest Neighbor
- Support Vector Machine
- Neural Networks

Unsupervised

- Latent Dirichlet Allocation
- K-Means
- Dimension Reduction
  - PCA



Reinforcement Learning

Introduction to
Neural Networks

Basic Machine Learning for NLP

> N-Grams
> Bag-of-Words
> Word-Classes (WordNet, Stemming, Lemmas)

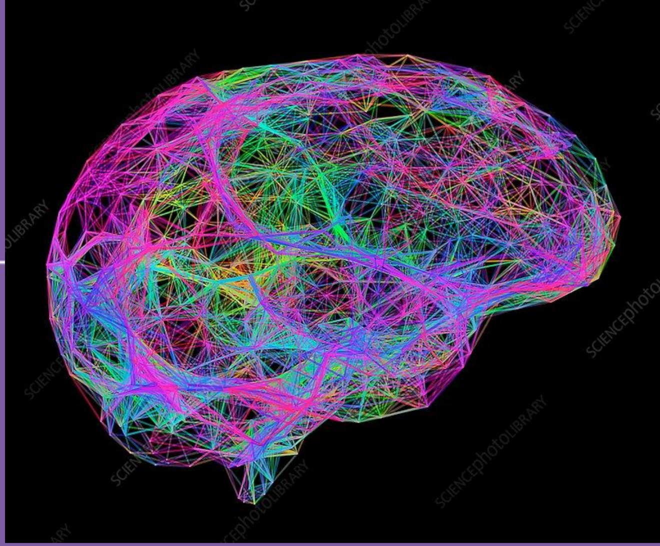
> Unsupervised Dimensionality Reduction (PCA)
> Unsupervised Clustering (K-Means, LDA)
> Supervised classification (Logistic Reg, SVM, Naïve Bayes,…)

Issues with ML for NLP

- Bag-of-words:
  - Sum of one-hot codes
  - Ignore the word order
- N-gram Language Models
  - Probabilities are estimated from counting on large corpus
  - Smoothing is used to prevent unseen events (OOV) → Zero probabilities.
- Word Classes
  - Similar words should share parameter estimation
  - Require an external, labeled, dataset (hard to obtain, single-lingual).

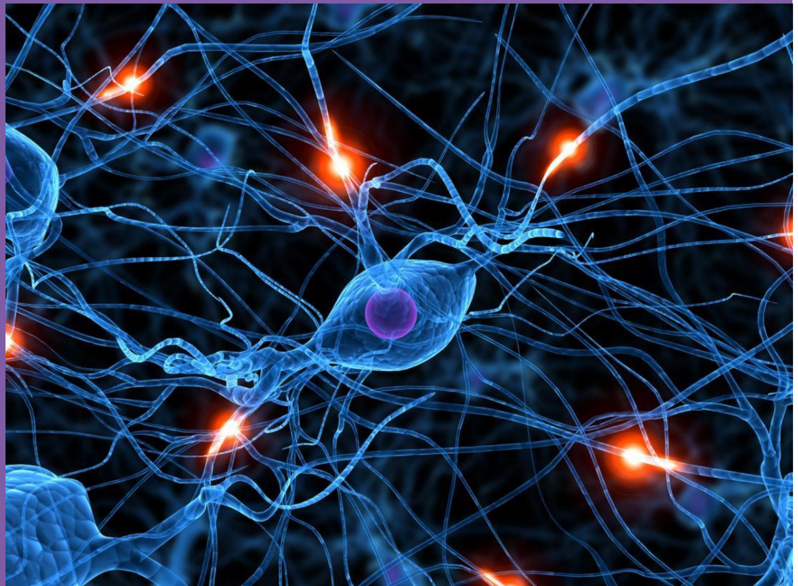
Neural Networks: NLP Motivation

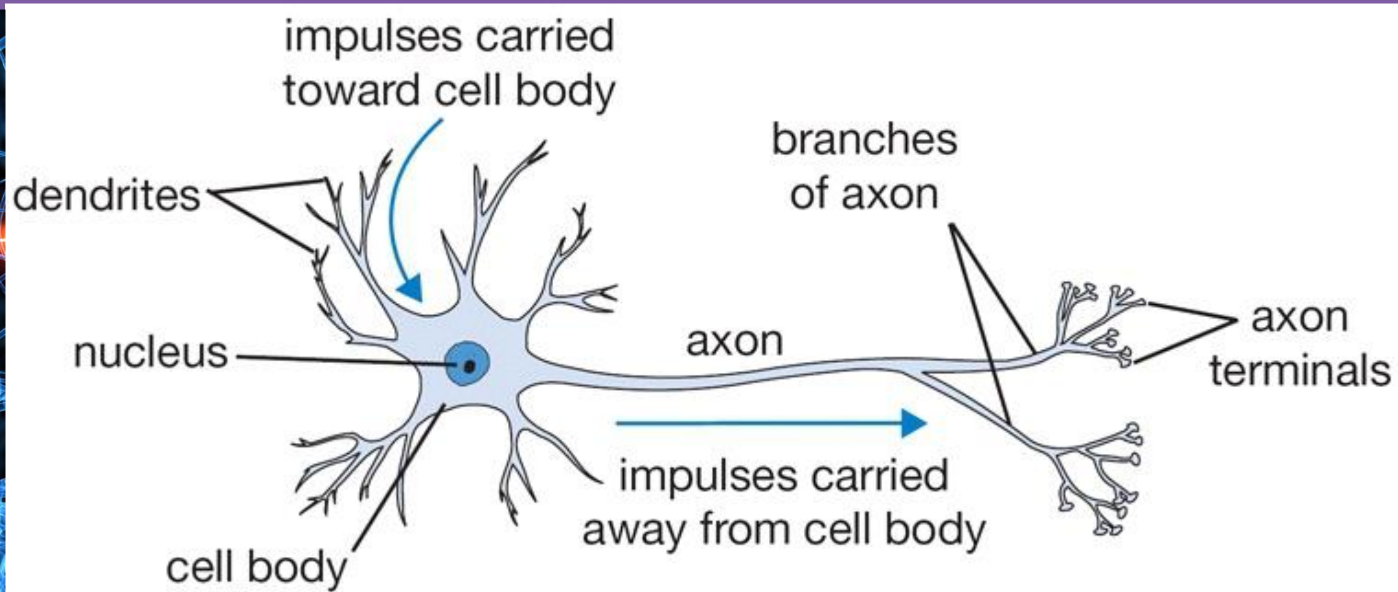
- We would like to have better techniques than plain word-counting.
- A method that can learn on its own, without too much need of a specialized person.
- (Are we there yet?)



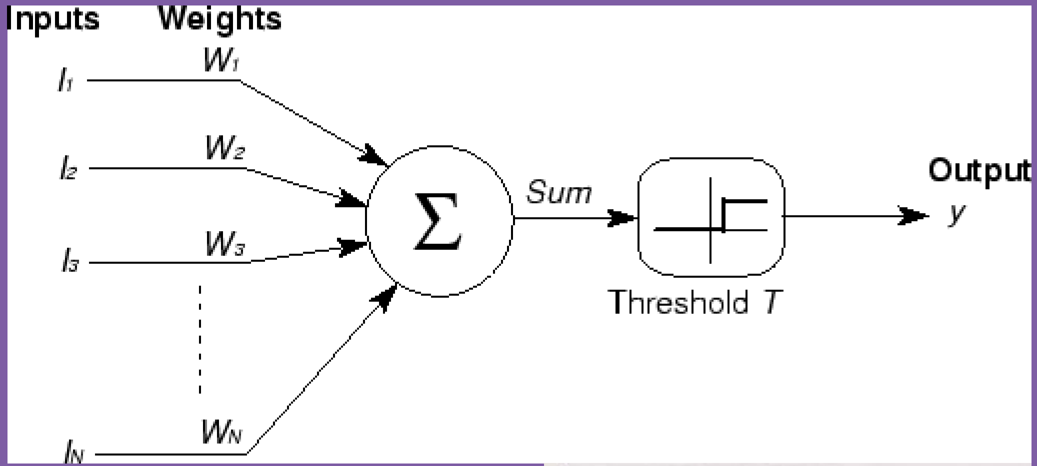
Biological Motivation

- There are ~86B neurons in our brain
- Neurons are connected to other neurons through an axon
- The structure resemble to a network, where information is passed from one neuron to another





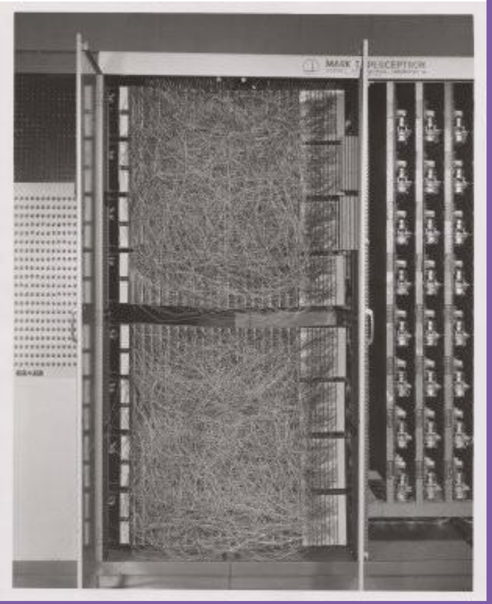
From Biology to Computation

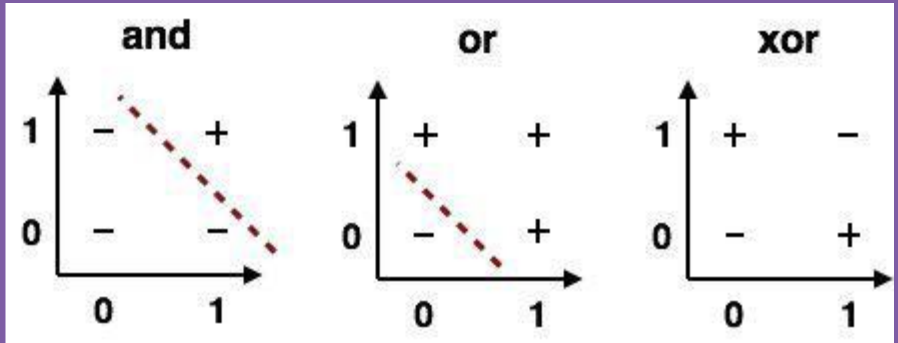


> 1943 – McCulloch-Pitts neuron
> A linear threshold gate

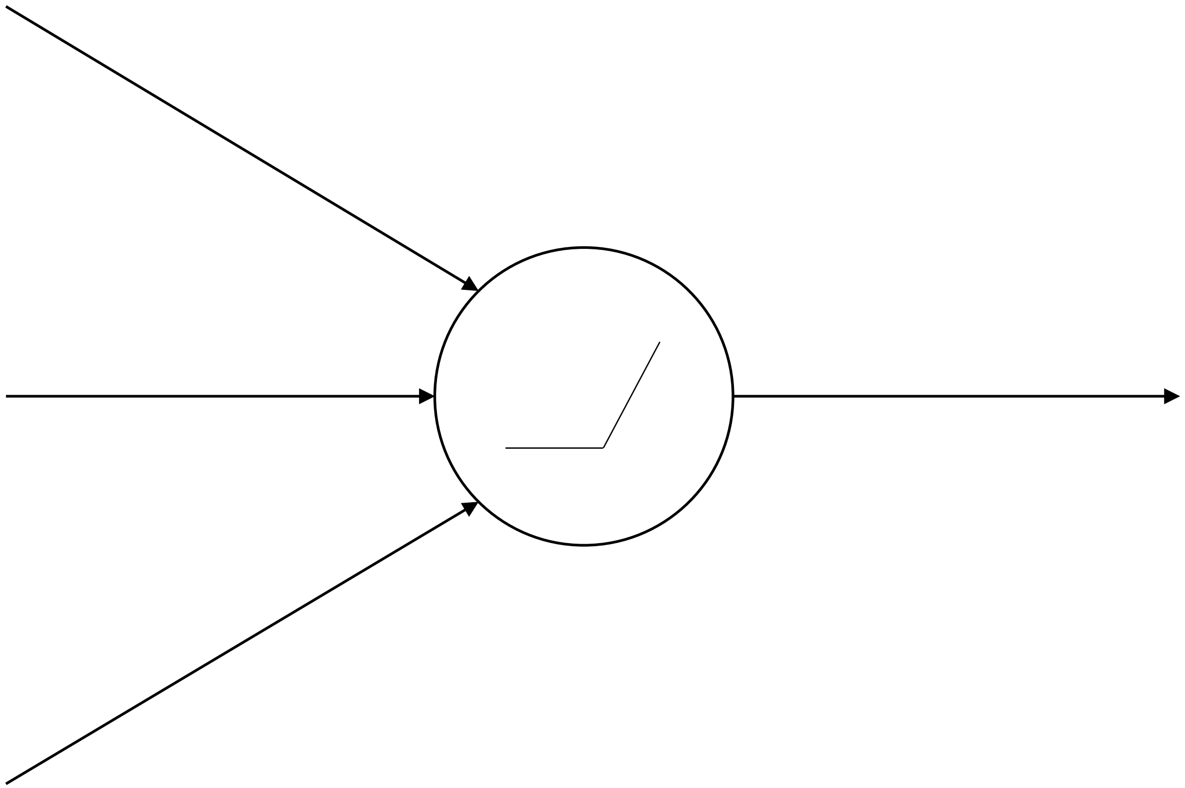
> 1958 – The Perceptron
> The weights were learned through data-examples

> Only capable of simple linear decision boundaries



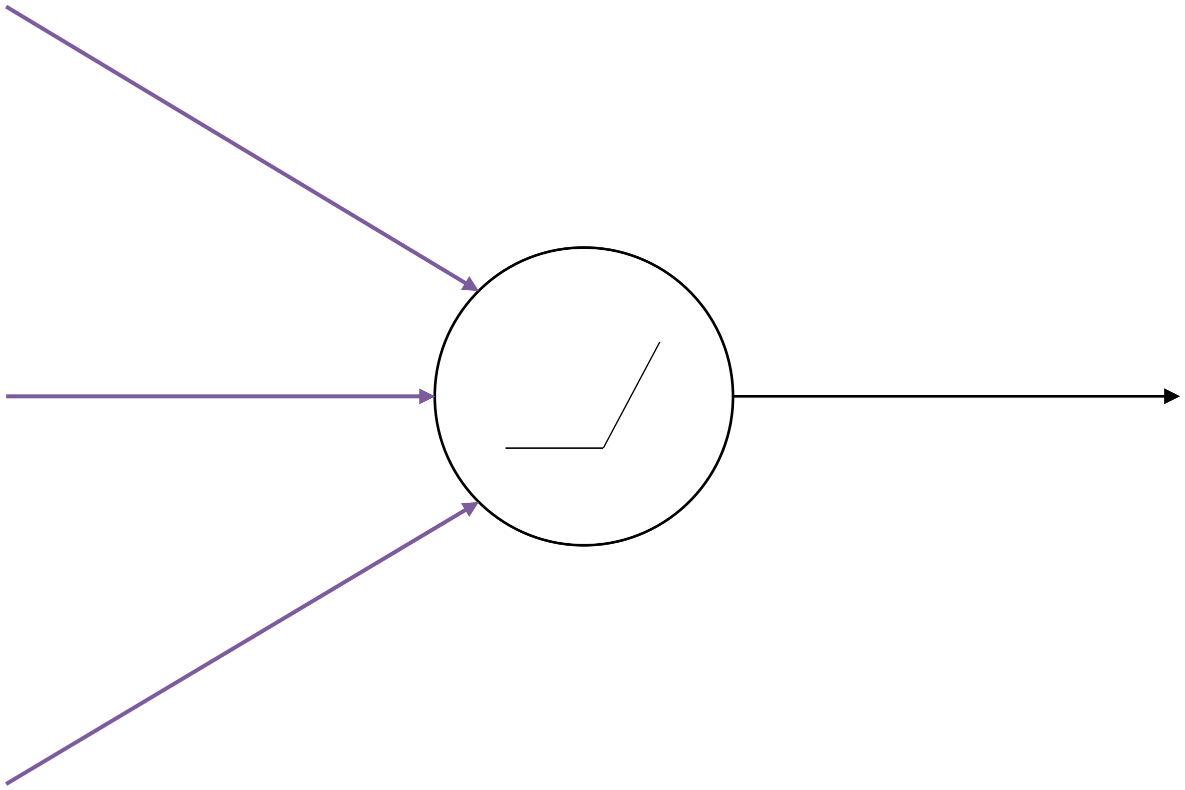


Neuron (Perceptron)

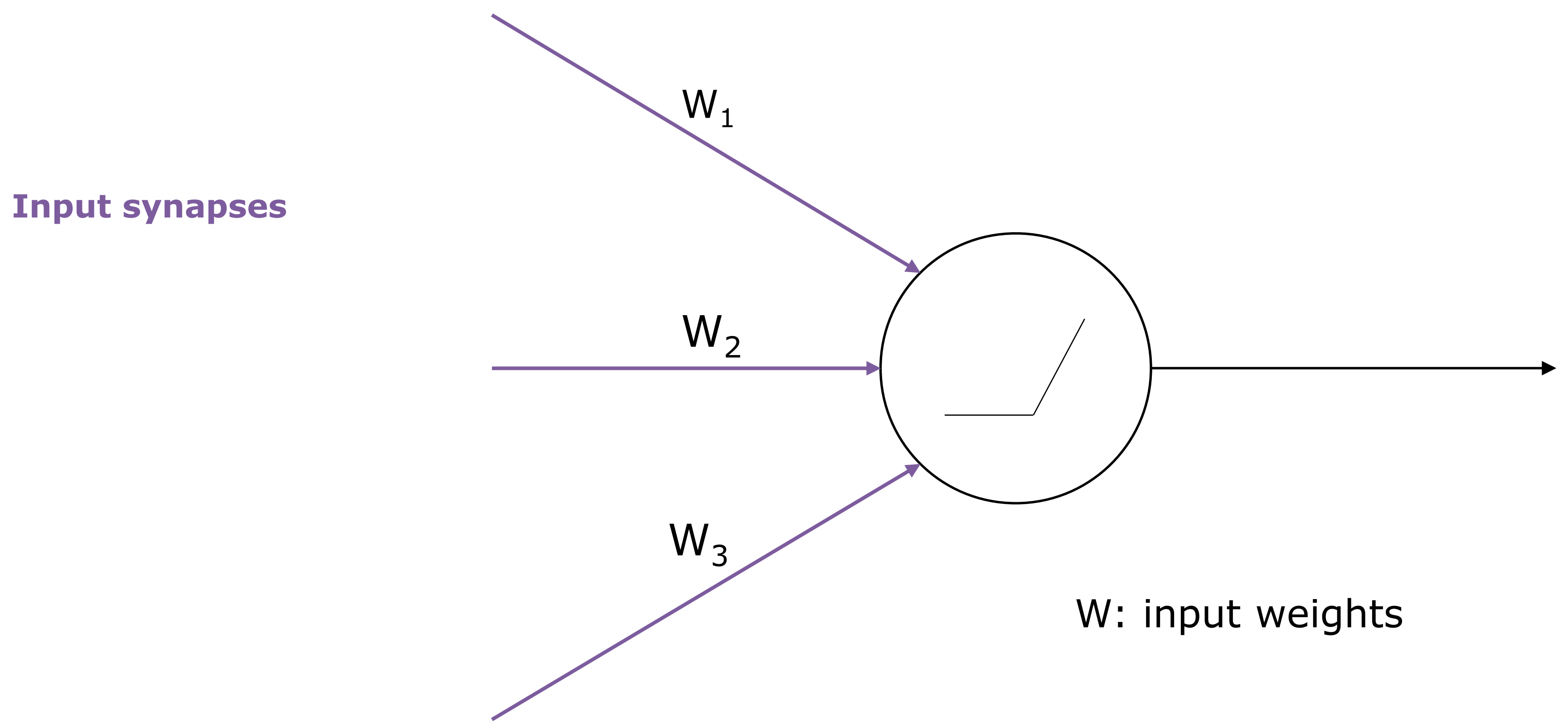


Neuron (Perceptron)

Input synapses



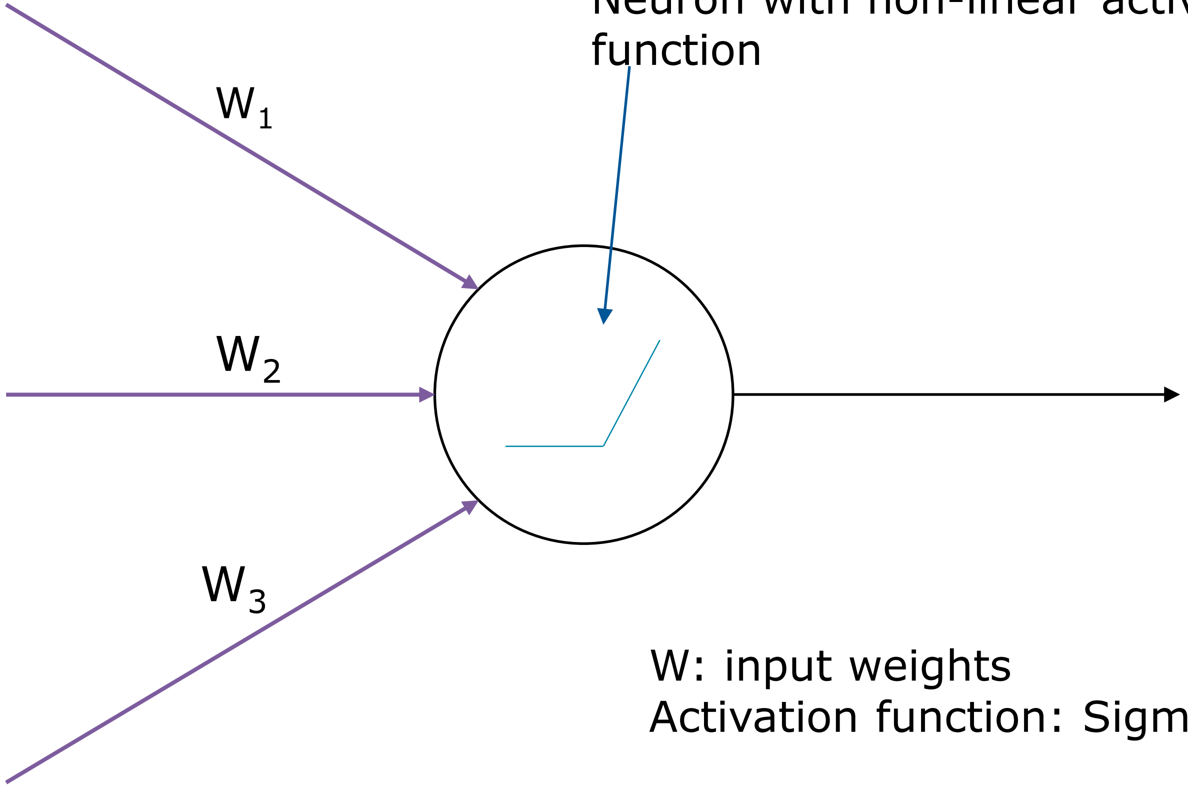
Neuron (Perceptron)



Neuron (Perceptron)

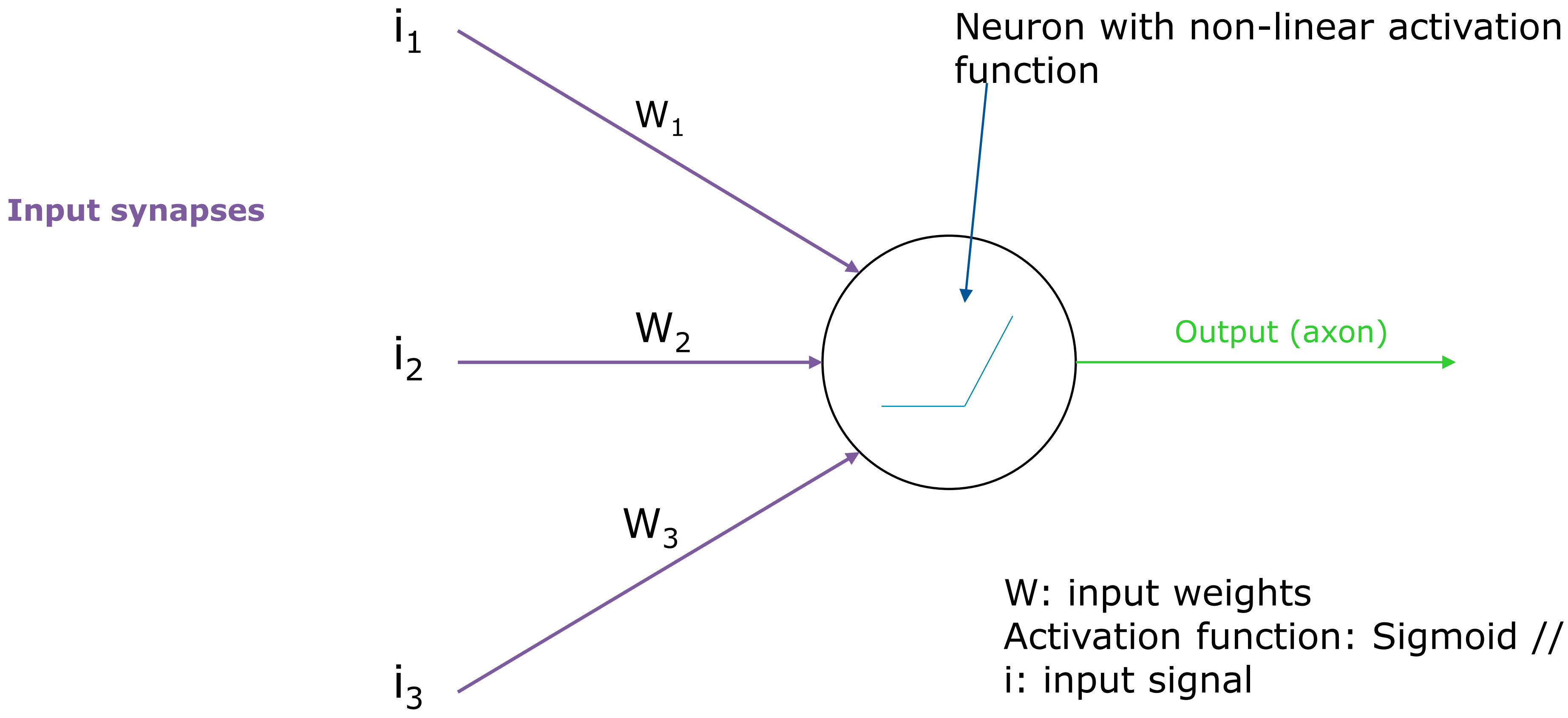
Neuron with non-linear activation function

Input synapses



W: input weights
Activation function: Sigmoid // max(0, value)

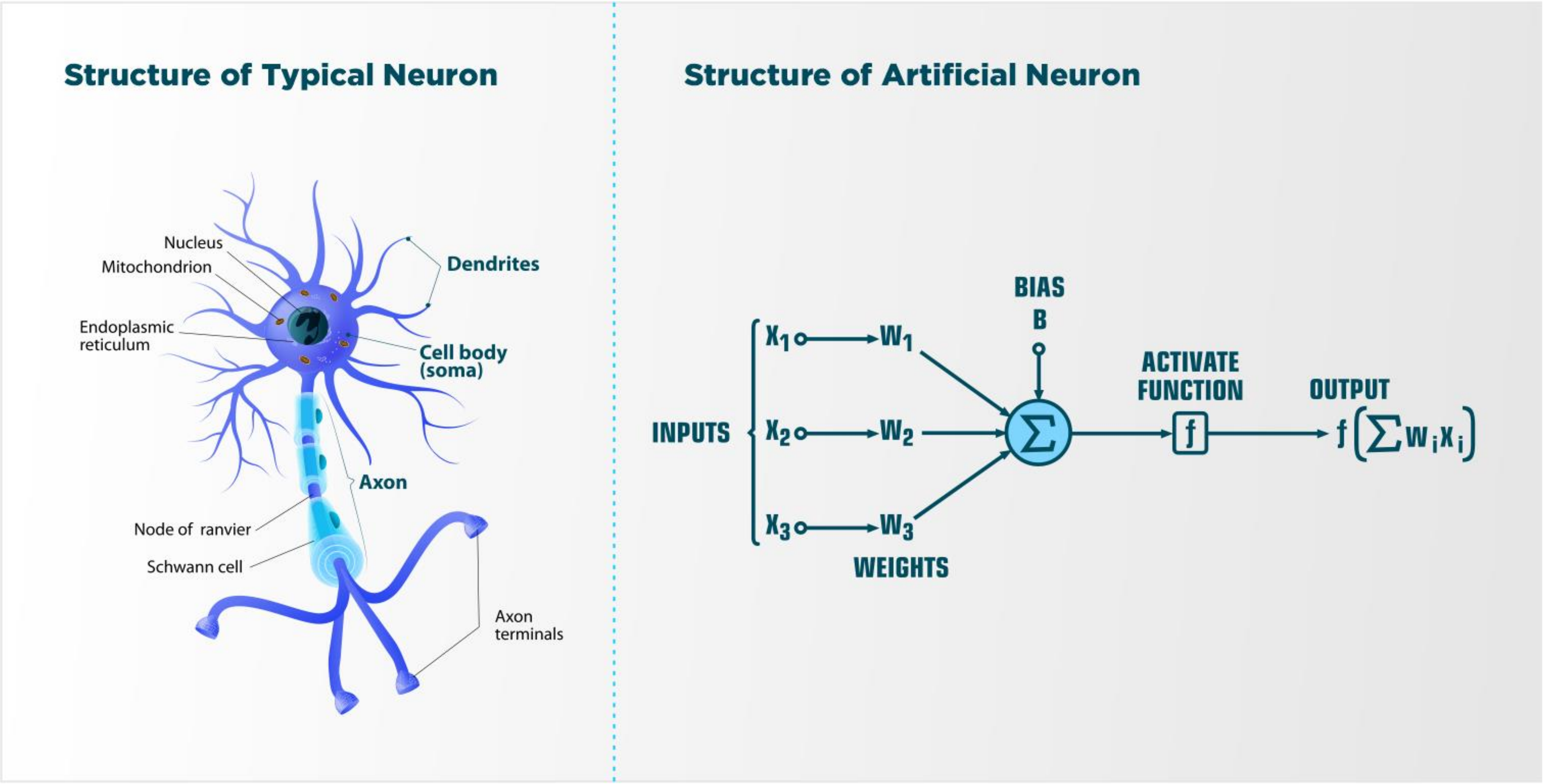
Neuron (Perceptron)



W: input weights
Activation function: Sigmoid // max(0, value)
i: input signal

$$Output = \max(0, I \cdot W)$$

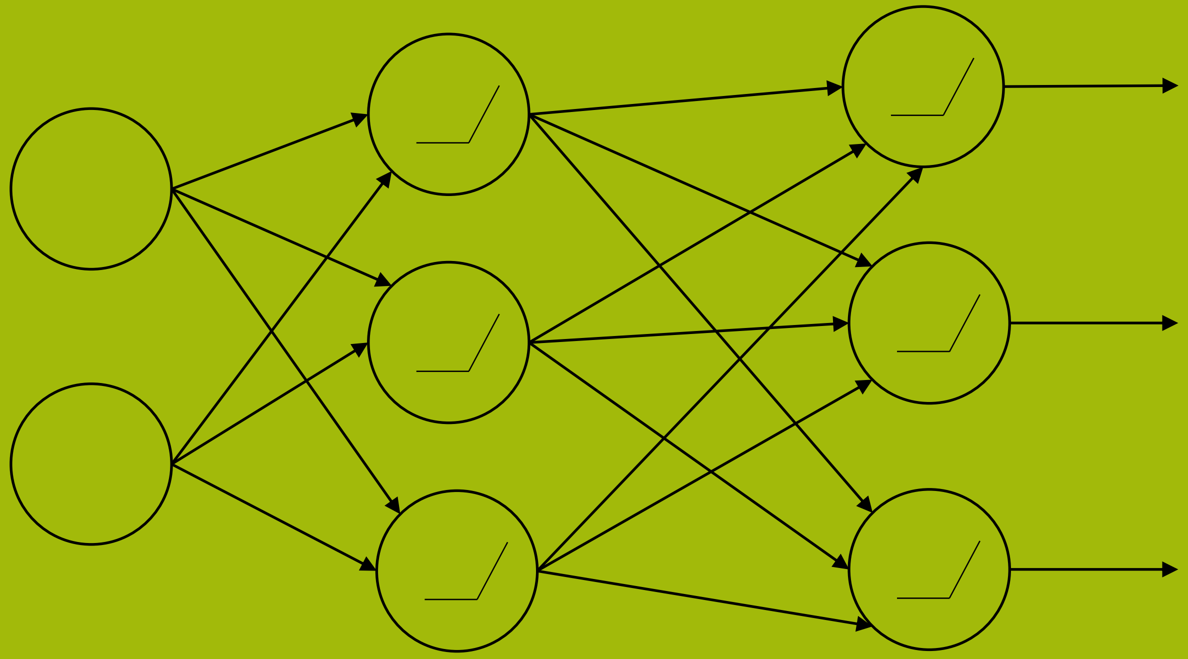
Perceptron vs Artificial Neuron



Neuron (Perceptron)

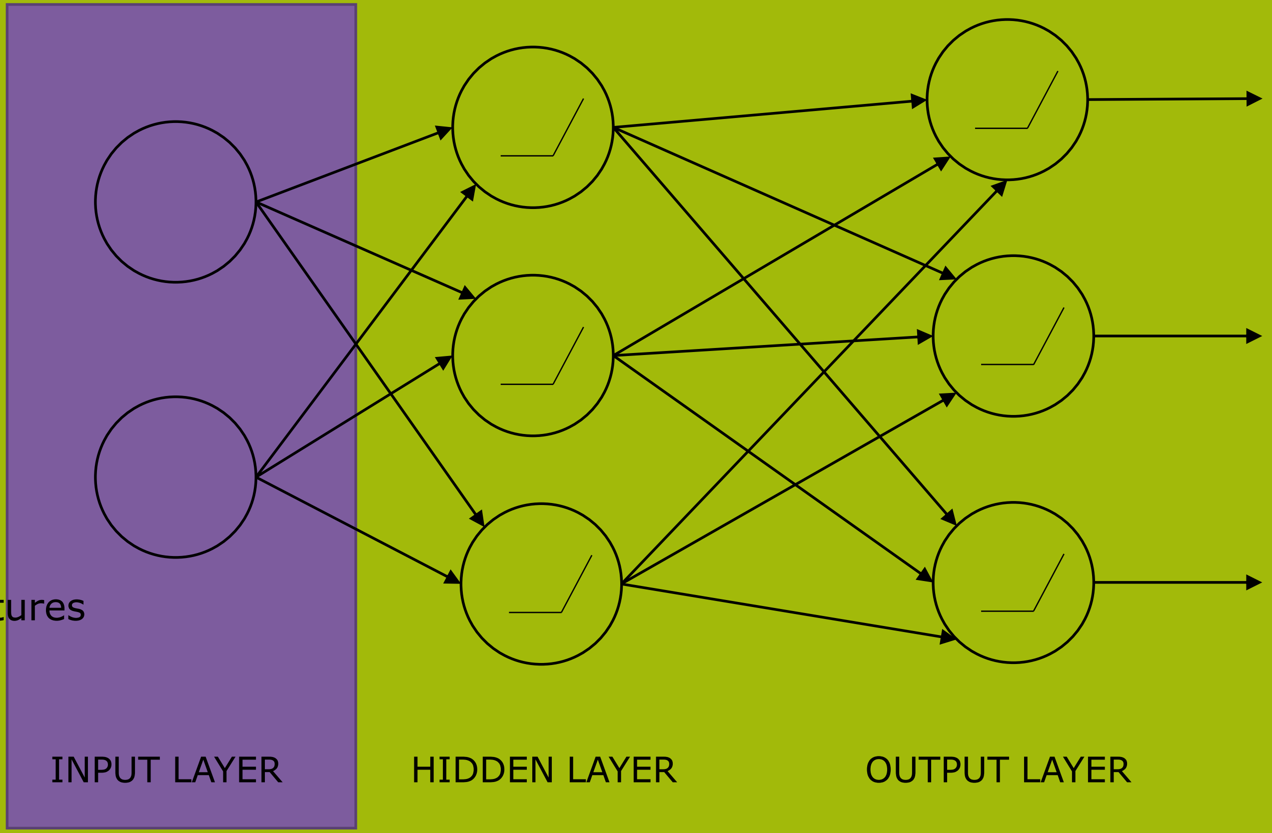
- The perceptron model is quite different from the biological neurons:
  - Those communicate by sending spike signals at various frequencies
  - The learning in brains seems also quite different
- It would be better to think of artificial neural networks as non-linear projections of data (and not as a model of brain)

Neural Network Layers



INPUT LAYER HIDDEN LAYER OUTPUT LAYER

Neural Network Layers



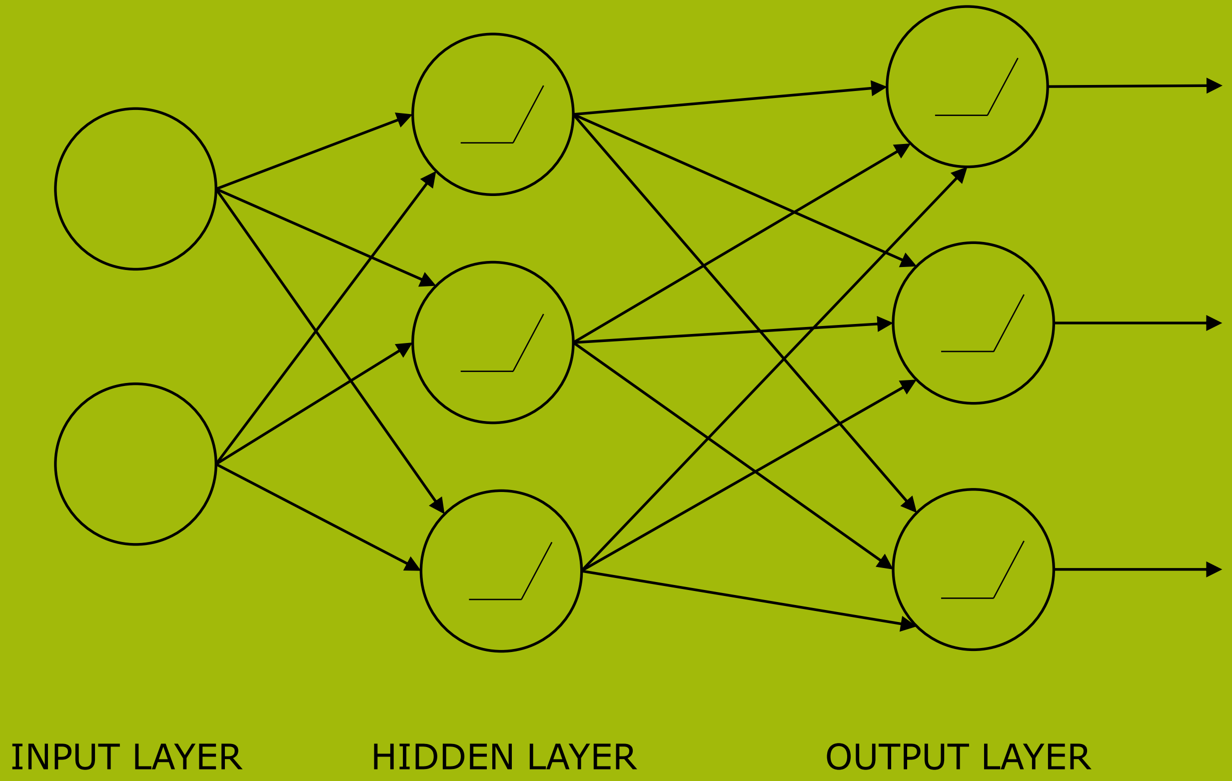
Input layer size

As many as the features vector size (length)

Output layer size

For classification:
Number of the classes

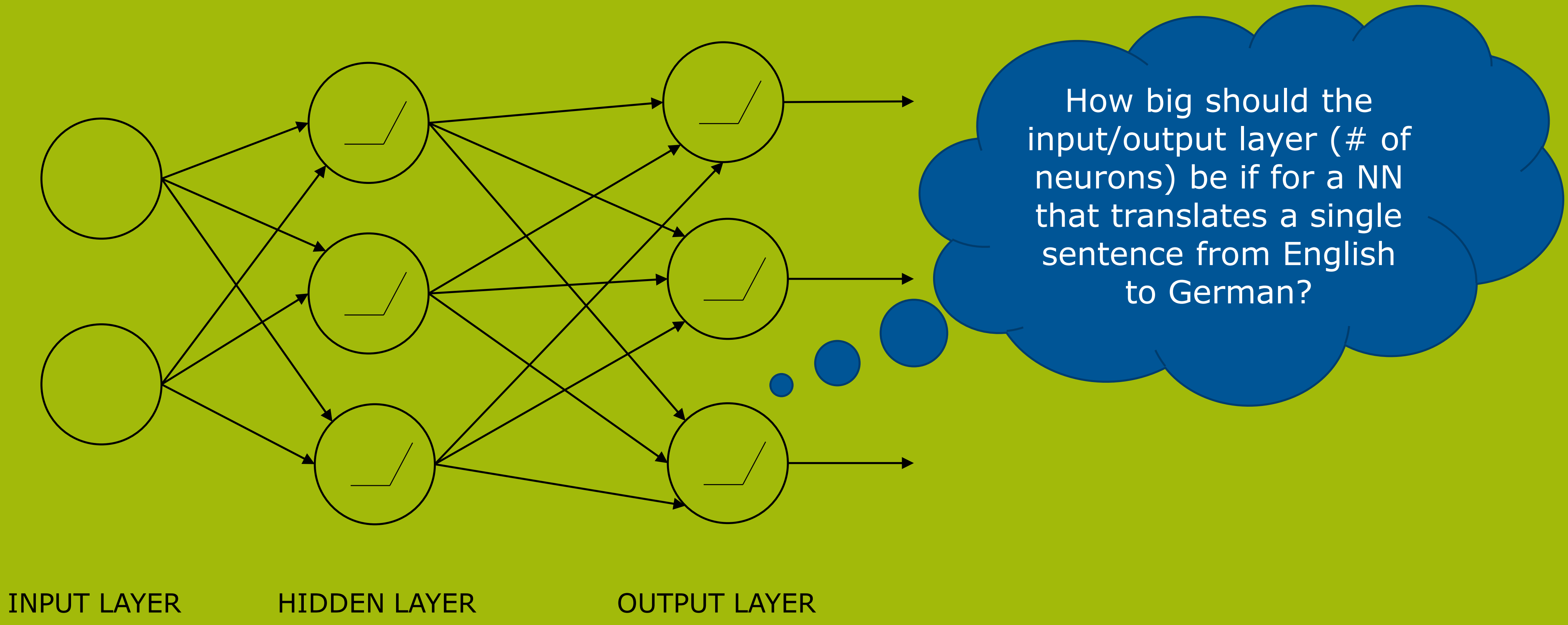
Neural Network Layers



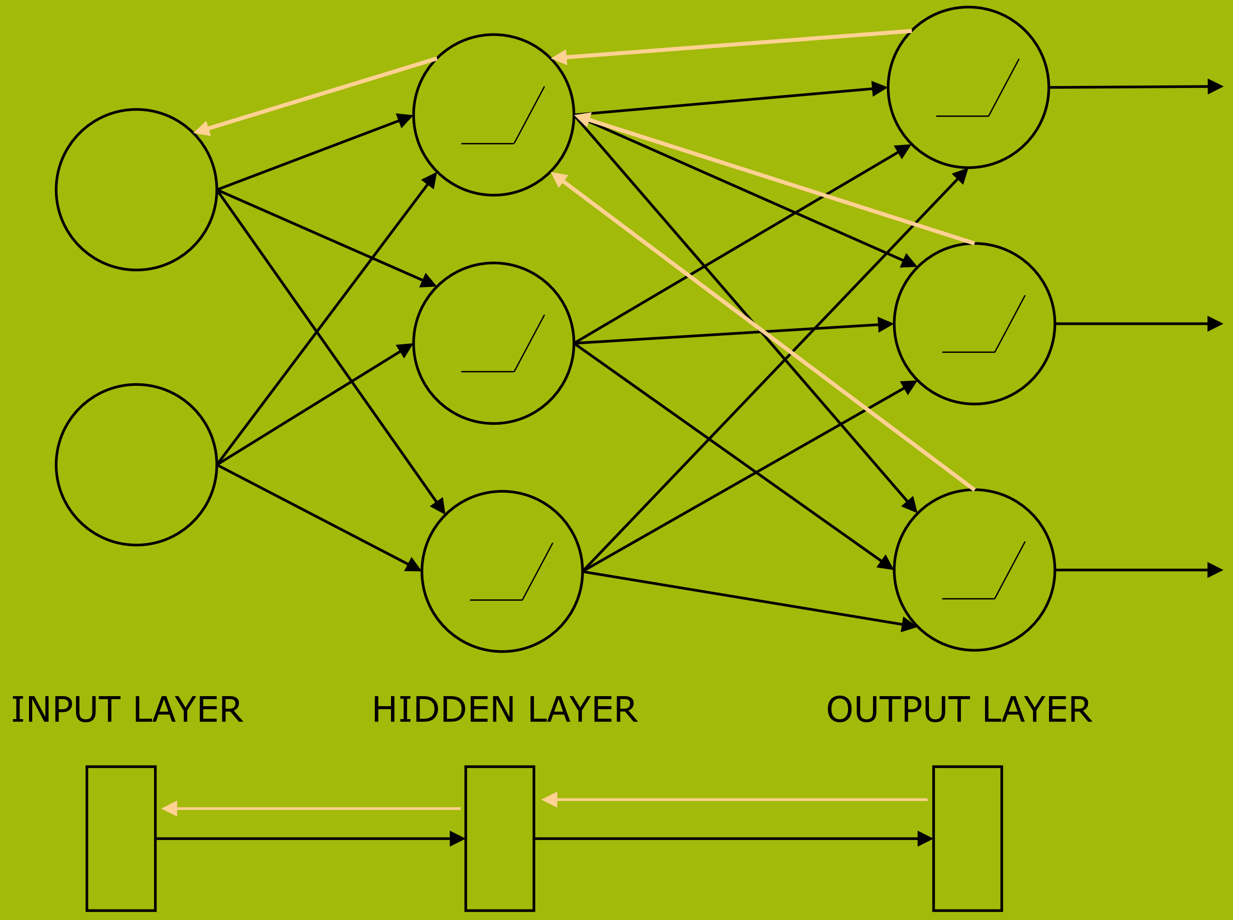
Output layer size

For classification:
Number of the classes

Neural Network Layers



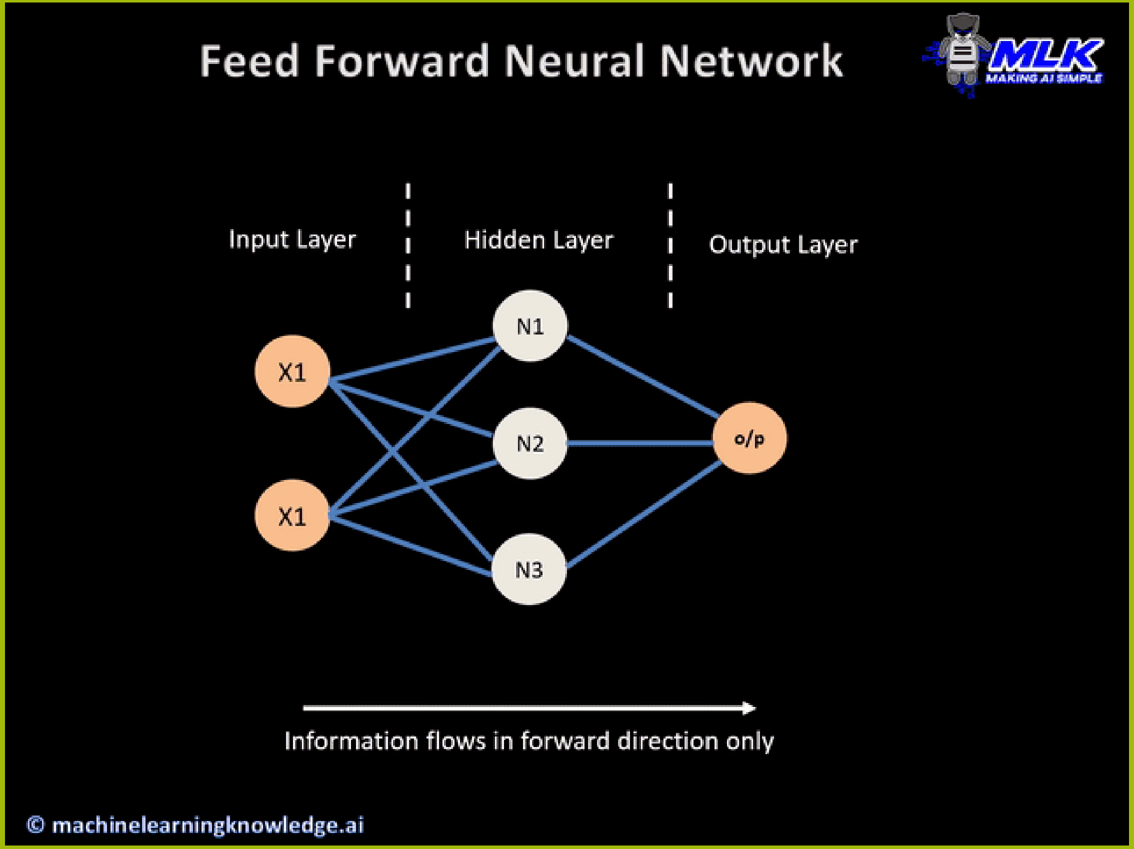
Neural Network Layers



- During the training, the network computes the values for a certain input through the network and compares the result to the given output.
- The gradient of the error is used to correct the neuron weights.

The Inference Part – Feed Forward

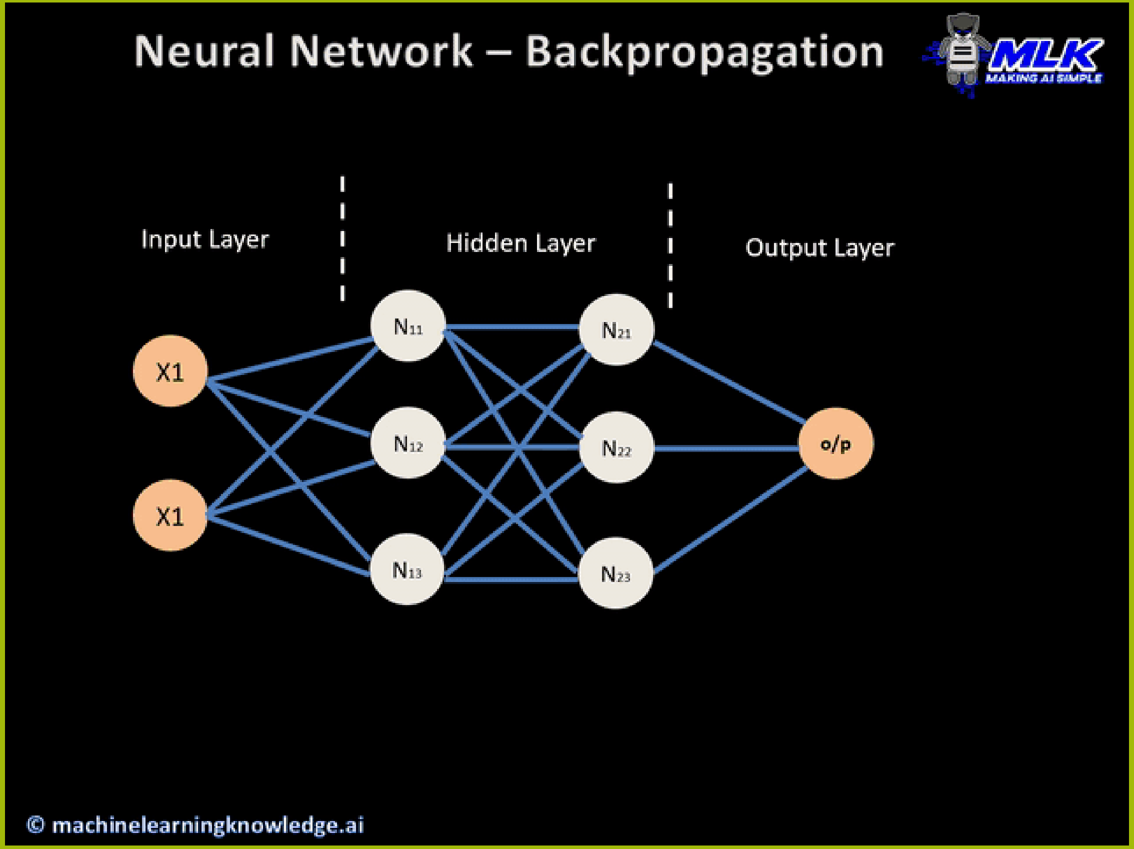
Given an input, predicting the output by calculating the values through the network



Backpropagation – Training Networks to Learn

Backpropagation occurs only during the training.

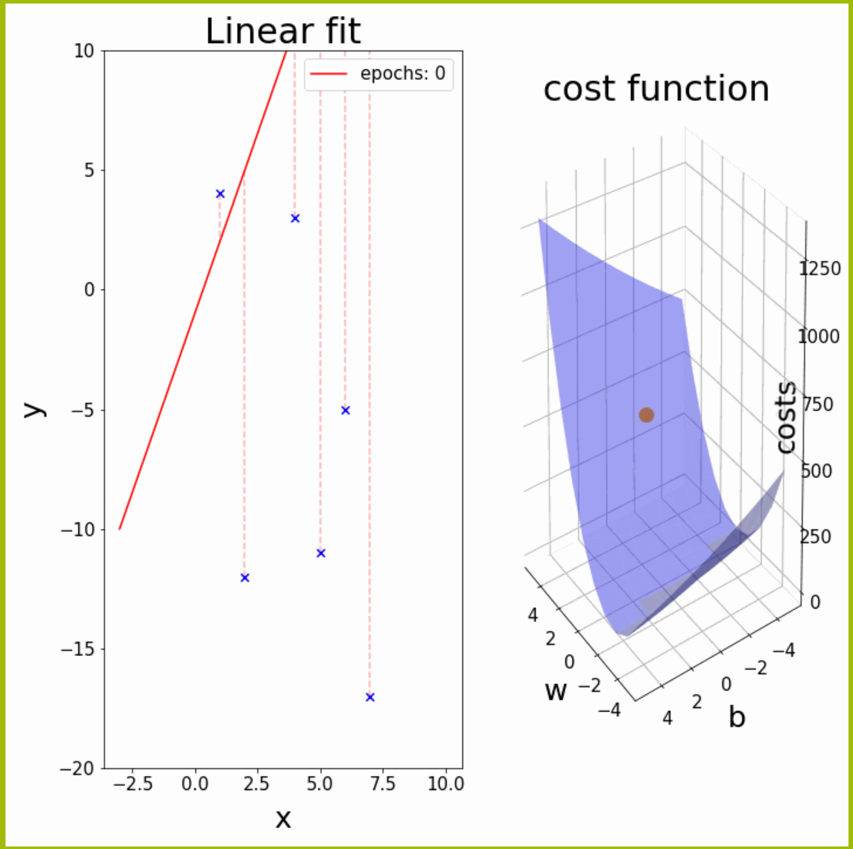
The network calculates the error (**loss function**) and gently updates the network weights, so next time when they “see” this sample, the prediction is closer to the desired result.



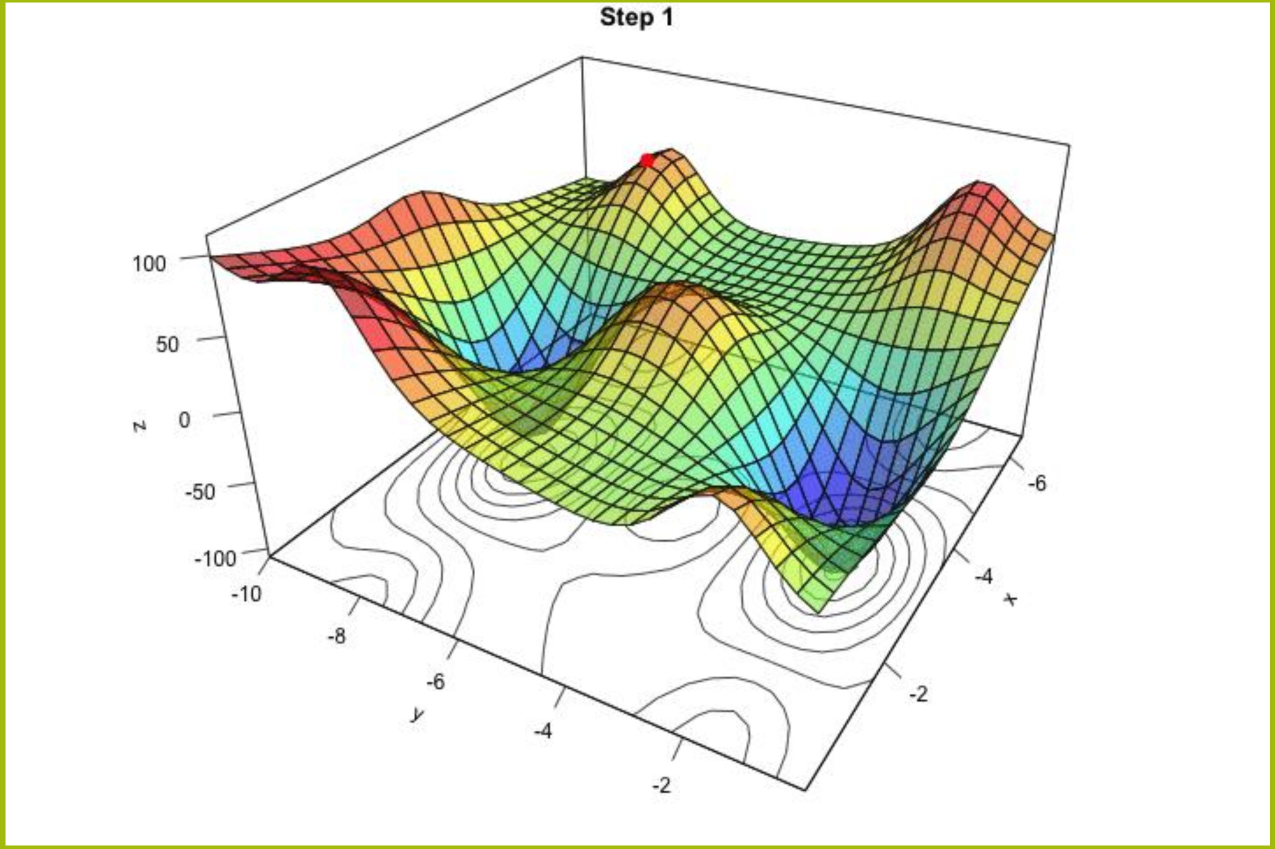
Error calculation

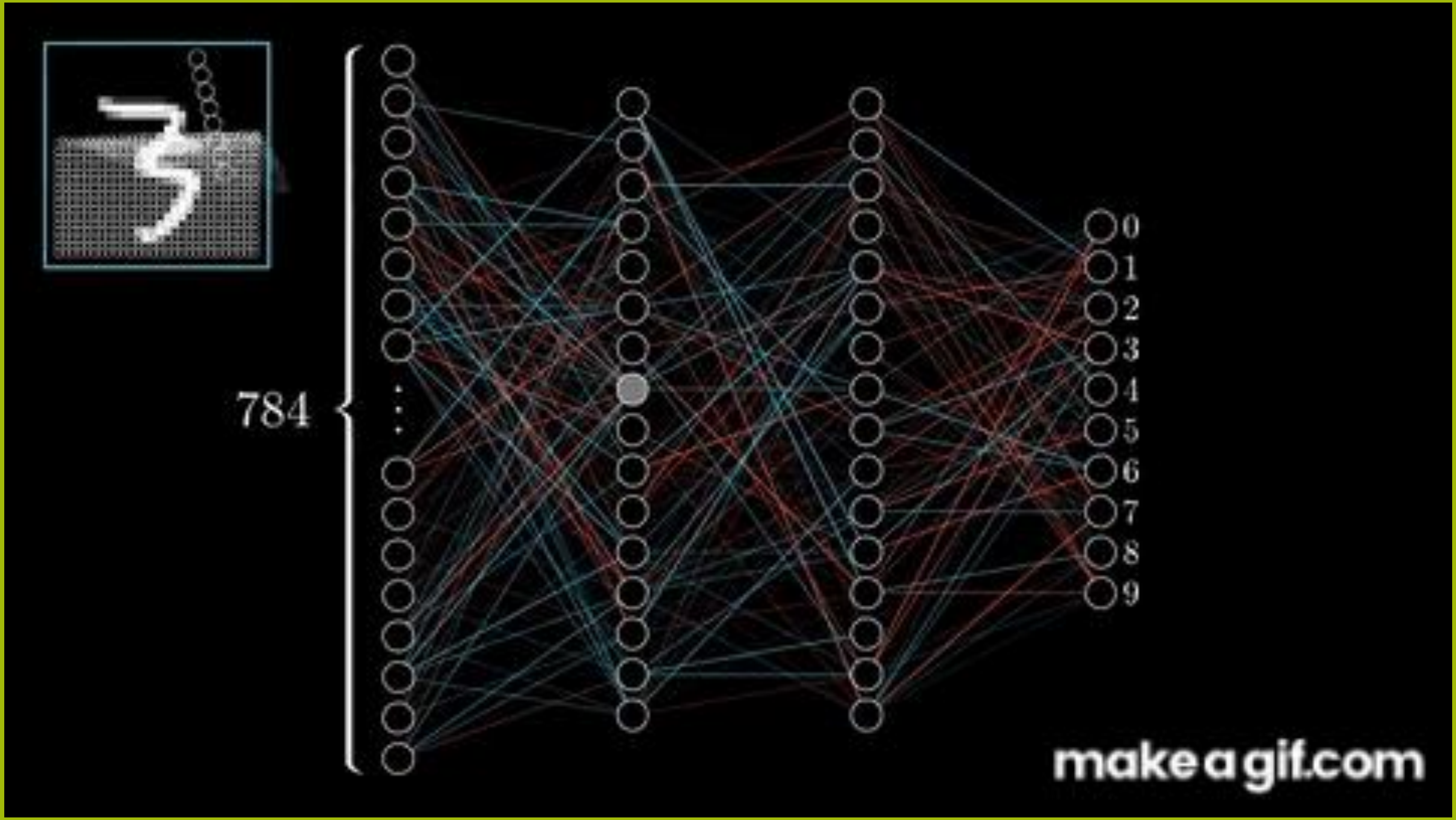
- Loss function
  - Calculates the distance between the network output and the desired value.
  - You decide which loss function to use, based on the task:
    - Mean Squared Error (MSE) – regression
    - Binary Cross Entropy – binary classification
    - Categorical Cross Entropy Loss – multiclass (one-hot-vectors)
    - Negative Log-Likelihood loss – multiclass
    - Multilabel margin loss – multilabel
    - And many, many more…
      torch.nn — PyTorch 1.13 documentation

Error Calculation – behind the scenes



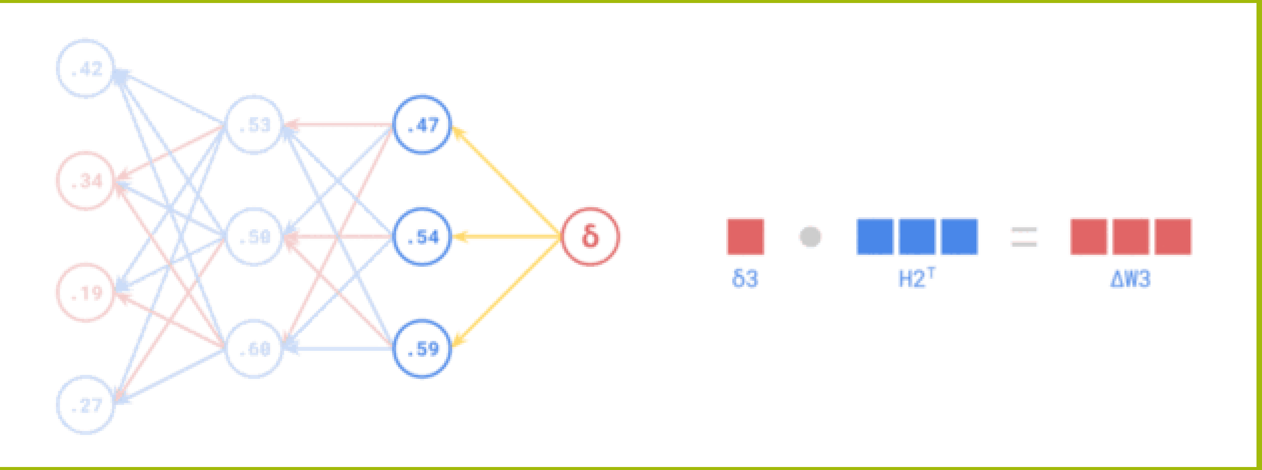
Error Calculation – behind the scenes

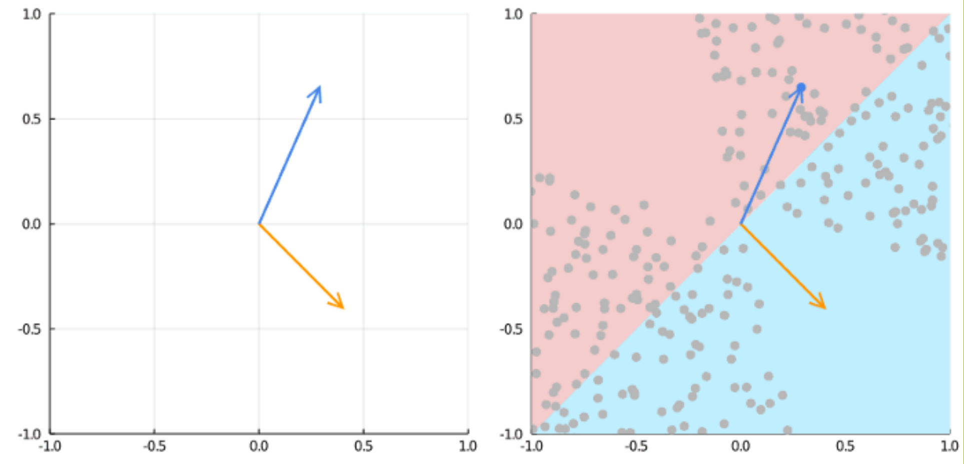



784
0
1
2
3
4
5
6
7
8
9
makeagif.com

FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

For a Deeper Dive





- The Building Blocks of Deep Learning | by Tyron Jung | The Feynman Journal | Medium
- What Makes Backpropagation So Elegant? | by Tyron Jung | The Feynman Journal | Medium

FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Training Does Not Include:

> Hyper-parameters setting must be done manually
  > Choice of activation function (sigmoid, RELU)
  > Number of layers / number of neurons per layer (network architecture)
  > Learning rate
  > Number of epochs (training cycles)
  > Regularization

FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

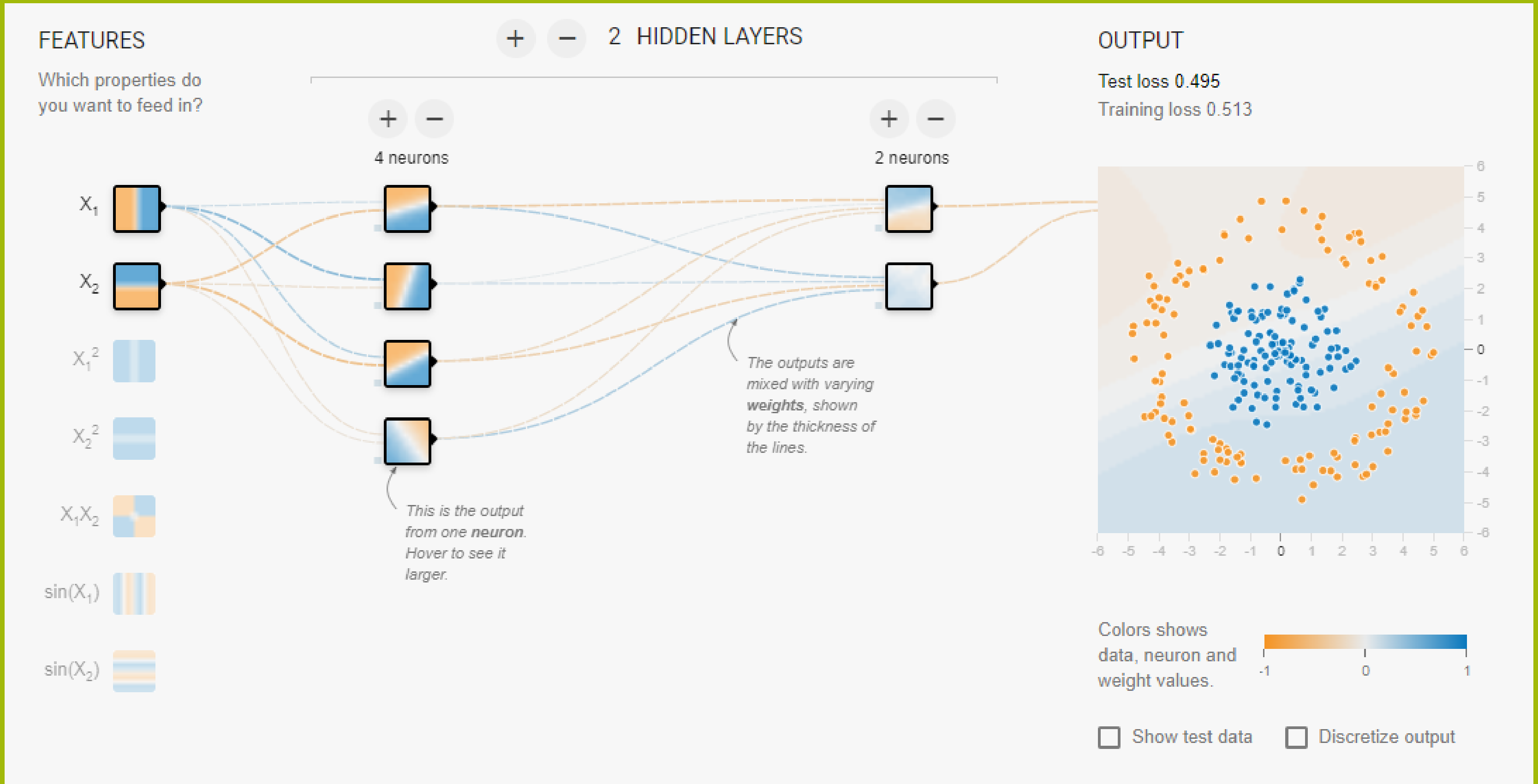
Complicated?

- Many existing frameworks do half of the job for you:
  - Fast.AI
  - Tensorflow + Keras
  - PyTorch + PyTorch-lightning // HuggingFace // spaCy // flair

- Best to start by re-using an existing model and modifying it if needed.

Neural Network - Demo

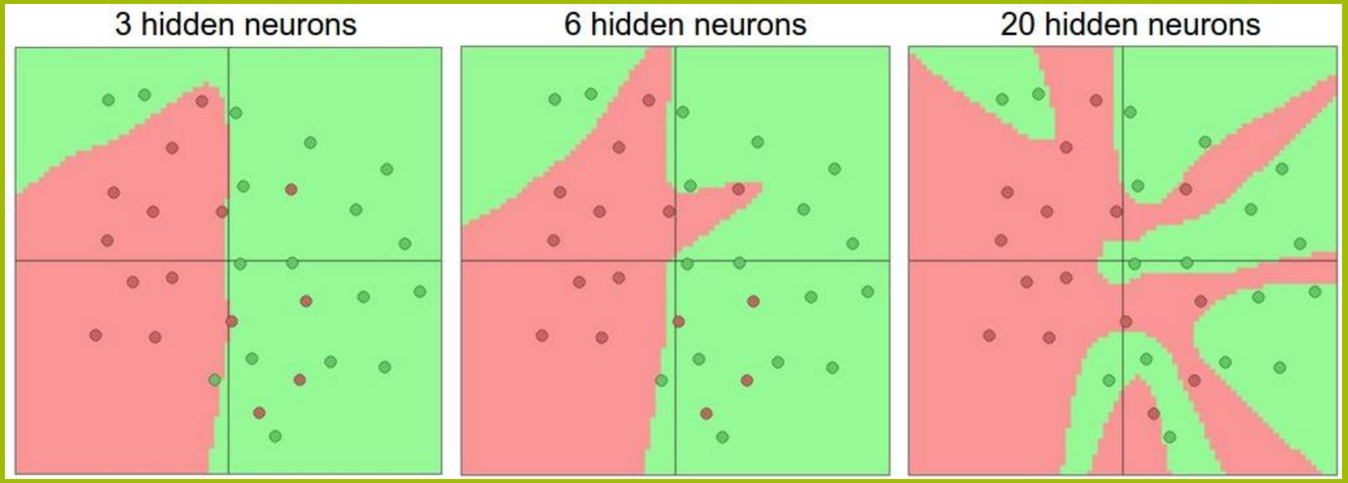
> https://playground.tensorflow.org/

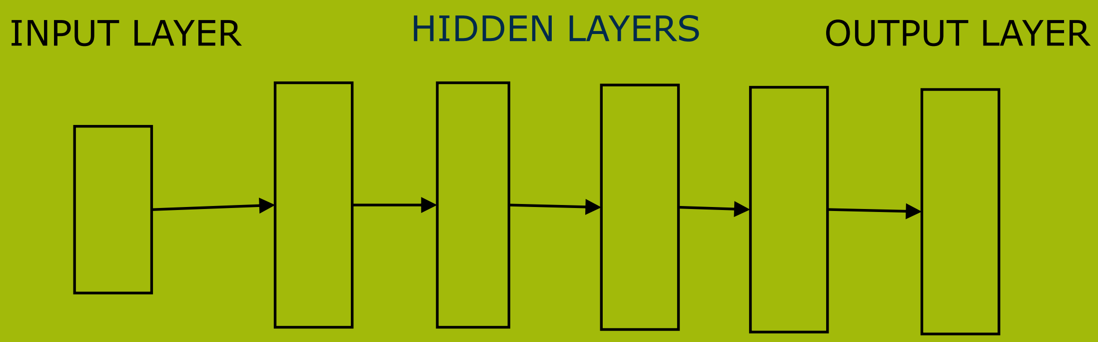


FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Deep Learning

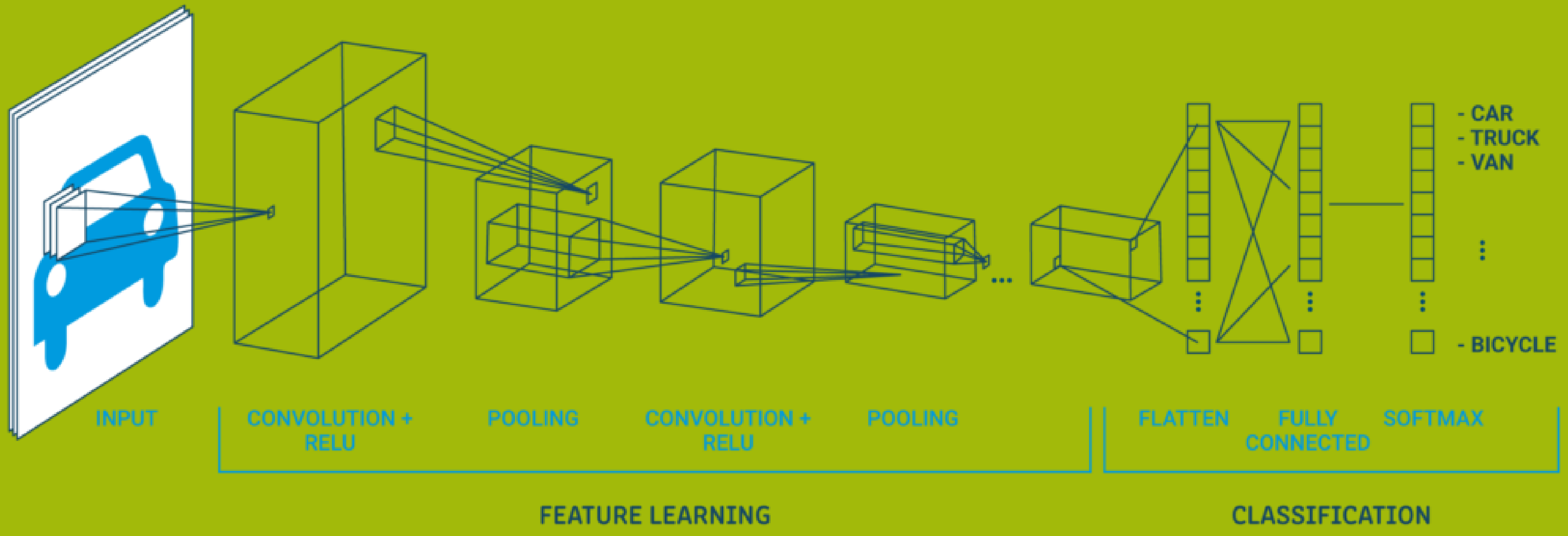
- > Deep Learning means more hidden layers → More computational steps, more degree of freedom
- > It can learn patterns that cannot be learned efficiently with shallow models.





Deep Learning

> Many architectures are researched daily.
> Still an open and (very) active research problem



Language Modeling

Are these sentences correct?

- This is a pen
- Pen this is a

Or these?

– He briefed to reporters on the chief contents of the statement
– He briefed reporters on the chief contents of the statement
– He briefed to reporters on the main contents of the statement
– He briefed reporters on the main contents of the statement

Language Model

Reminder:

- Joint probability: $P(X = x, Y = y)$
  - Independence, Chain Rule
- Conditional Probability: $P(X = x | Y = y)$

Naïve Bayes + n-grams:

- $P(home | there\ is\ no\ place\ like)$
- $P(there\ is\ no\ place\ like\ home)$
- $P(< stop > | there\ is\ no\ place\ like\ home)$

FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Language Model (Unigram)

> Every word in the vocabulary is assigned with a probability.
> W_1, W_2 - Random Variables (one per word)

Naïve Bayes + uni-gram model:

> $P(W = w) = P(W_1 = w_1, W_2 = w_2, \ldots, W_{L+1} = stop) =$

$\left(\prod_{l=1}^{L} p(W_l = w_l | W_{1:l-1} = w_{1:l-1}\right) p(W_{L+1} = stop | W_{1:L} = w_{1:L}) =$

$(\prod_{l=1}^{L} p(w_l | history_l)) p(stop | history_L)$

FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Language Model

How can we handle unknown words?

Part of some Unigram Distribution

[rank 1]
p(the) = 0.038
p(of) = 0.023
p(and) = 0.021
p(to) = 0.017
p(is) = 0.013
p(a) = 0.012
p(in) = 0.012
p(for) = 0.00
...

…
[rank 1001]
p(joint) = 0.00014
p(relatively) = 0.00014
p(plot) = 0.00014
p(DEL1SUBSEQ) = 0.00014
p(rule) = 0.00014
p(62.0) = 0.00014
p(9.1) = 0.00014
p(evaluated) = 0.00014

Language Model – Word-History

- How many words back should we calculate?
  - Full-history model: every word is assigned some probability, conditioned on *every* history
  - N-Gram model: every word is assigned some probability, conditioned on a *fixed-length* history (n-1)

Language Model

- The bigger n is, the bigger its perplexity, and the more coherent it is.
- Doesn't have to be words.
  Can be composed of:
  - characters
  - combinations of frequent characters
  - phrases
  - bytes (8-bits)
  - etc.

Word representations

- So far, we've modeled words as discrete symbols
  car = [model column] 2
  automobile = [model column] 8

- One-hot representation:
  car = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, ..., 0]
  automobile = [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ..., 0]
  Dimension: # of words in vocabulary

- How can we compute similarity of two words?

Similarity between words

"fast" is like "rapid"

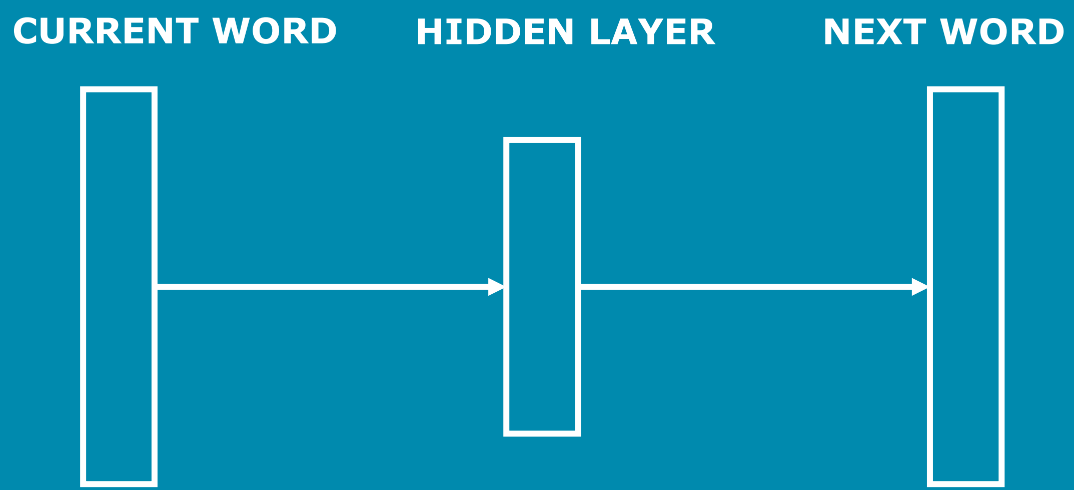
"tall" is like "height"

> Question answering:

Q: "How tall is Mt. Everest?"

Candidate A: "The official height of Mount Everest is 29029 feet"

Basic Neural Network applied to NLP



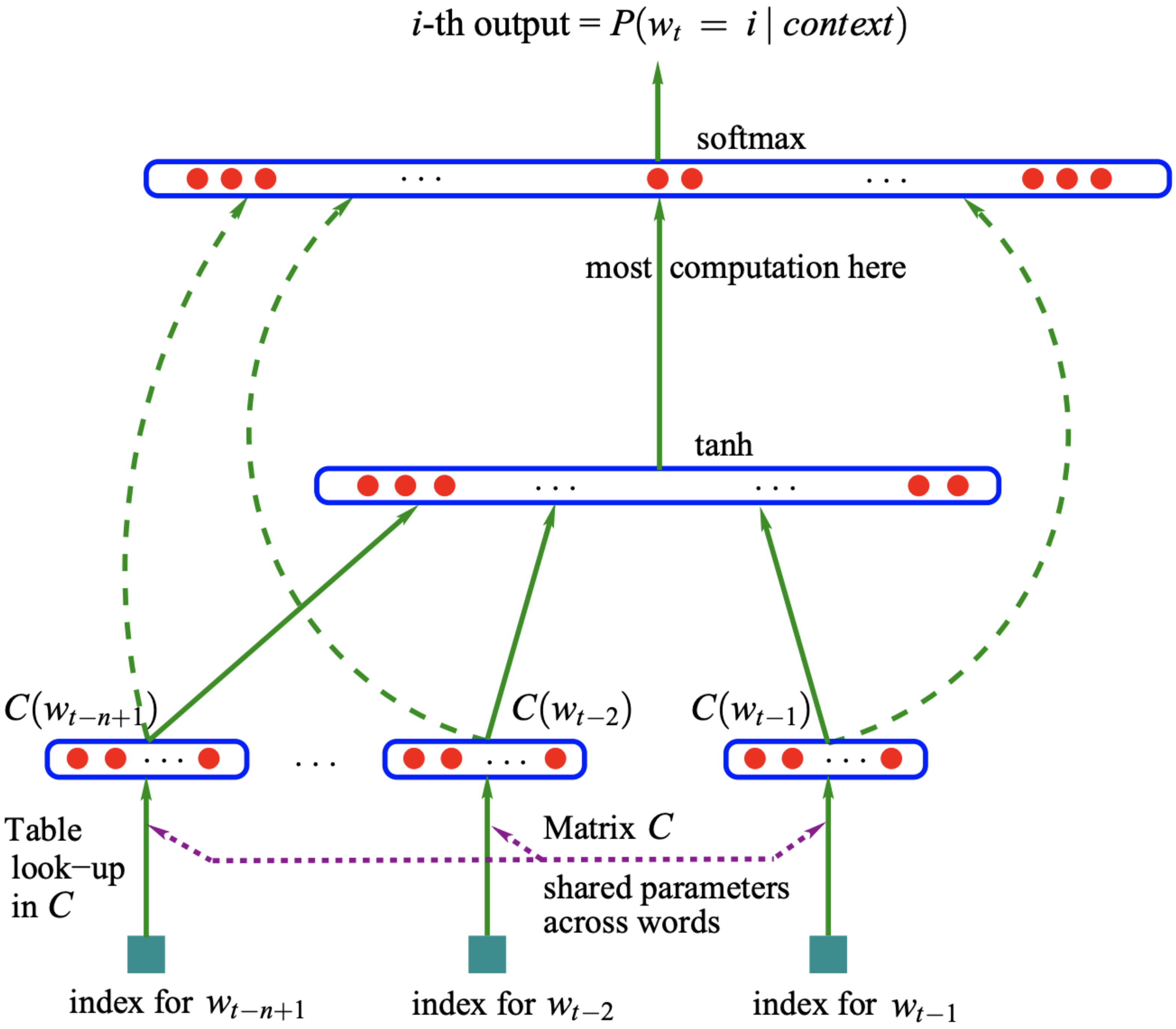
- N-gram neural language model: predicts the next word
- The input is encoded as a one-hot-encoder
- The model will learn compressed (dense), continuous representations of words (usually the matrix of weights between the input and hidden layers)

A Neural Probabilistic Language Model

2003 - Bengio et. Al

- The hidden layer has N rows, where |N|=|vocab|
- The trained matrix creates vectors for each word.
- These vectors had interesting properties:
  - Similar words were (relatively) near each other in the vector space.

- But… expensive to train (3 weeks for 5 epochs)



A fixed-window neural Language Model

output distribution

$$\hat{\boldsymbol{y}} = \mathrm{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

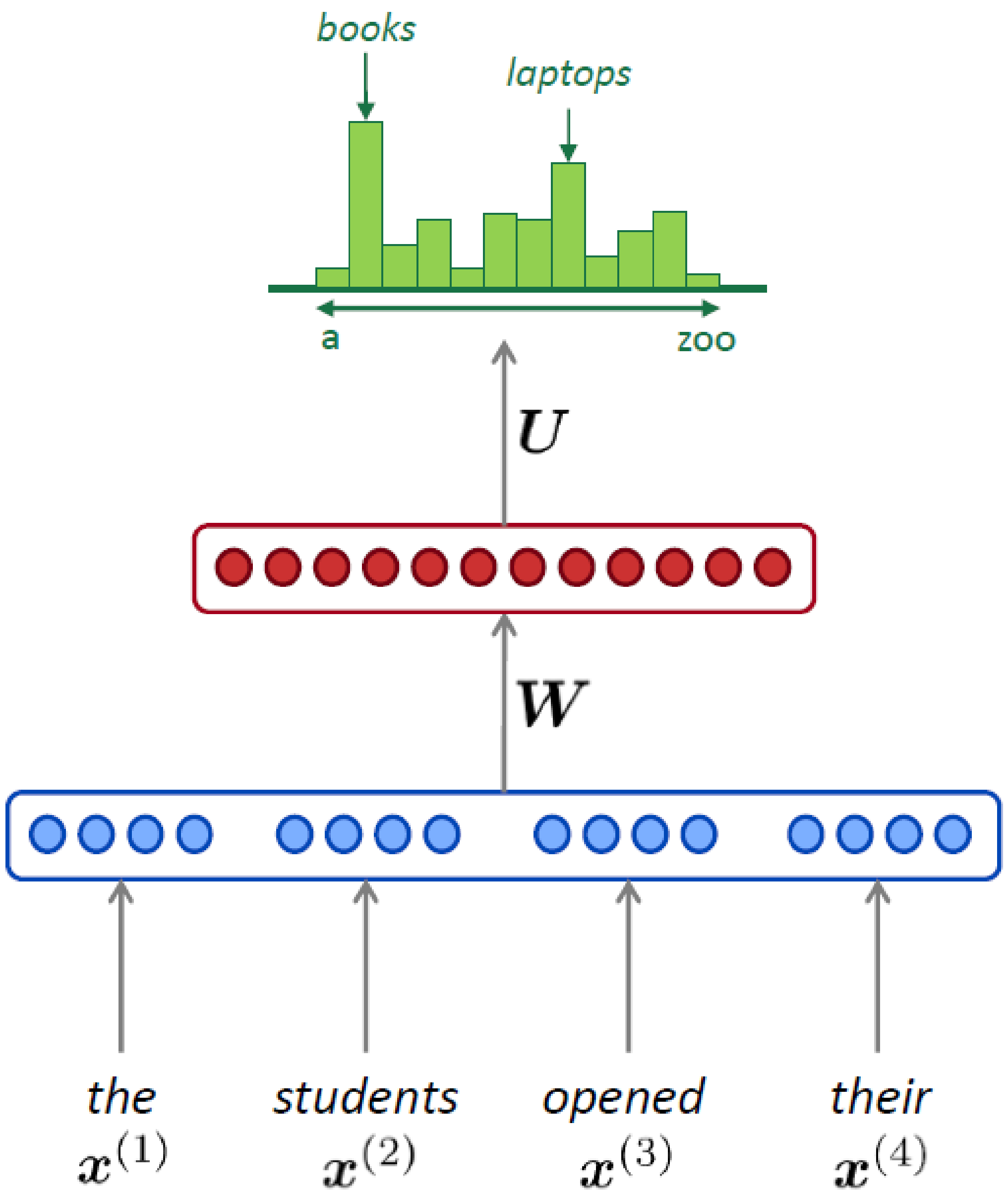
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

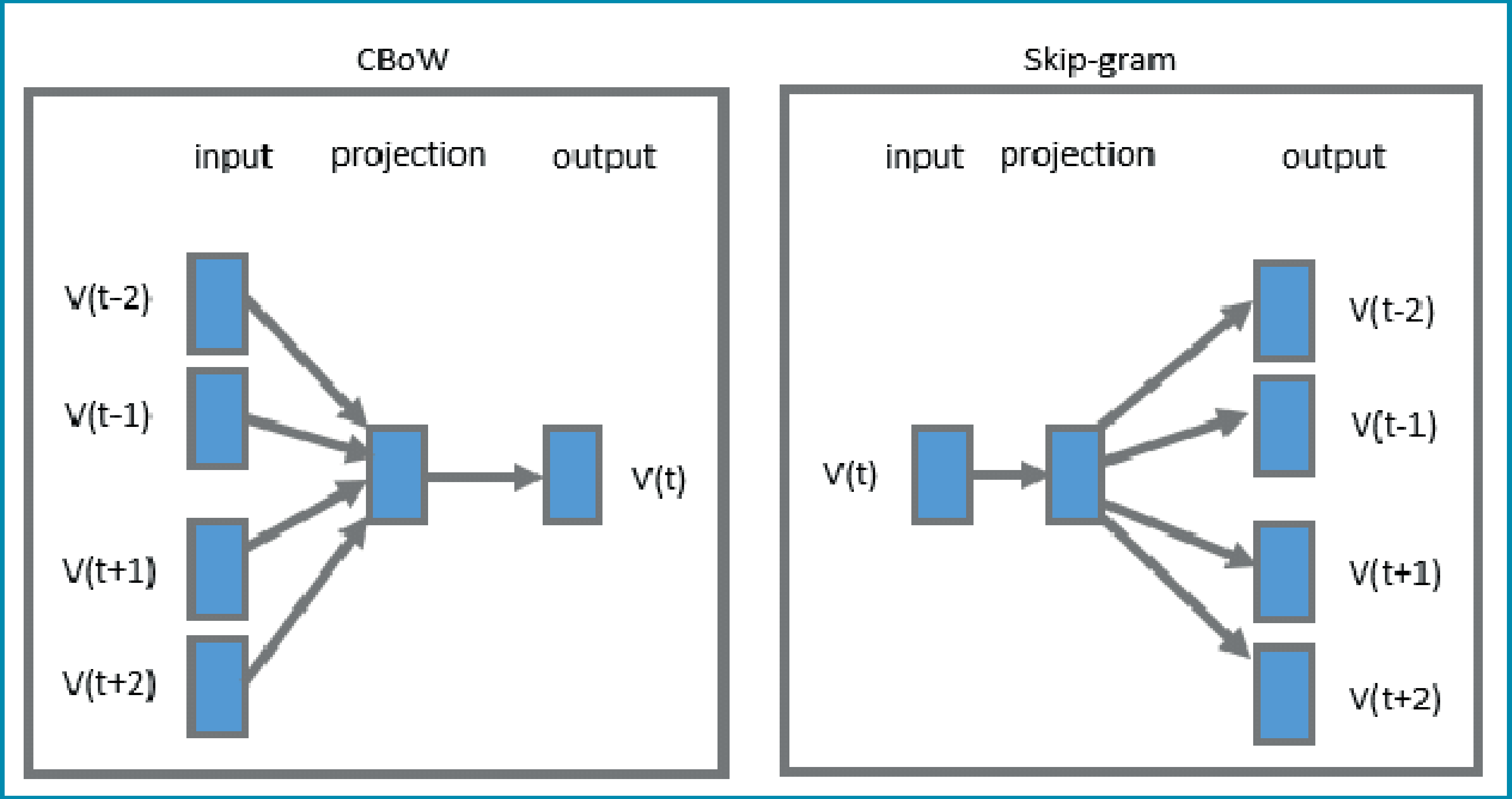
words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$



UNIVERSITY OF APPLIED SCIENCES

Word Vectors (Mikolov et. al 2013)



Word Vectors - Intuition

Distributional Hypothesis (J.R. Firth 1957)

- Words that occur in **similar contexts** tend to have **similar meanings**
  - “You shall know a word by the company it keeps”
  - “If A and B have almost identical environments ”

- Words which are **synonyms** tend to occur in the **similar context**

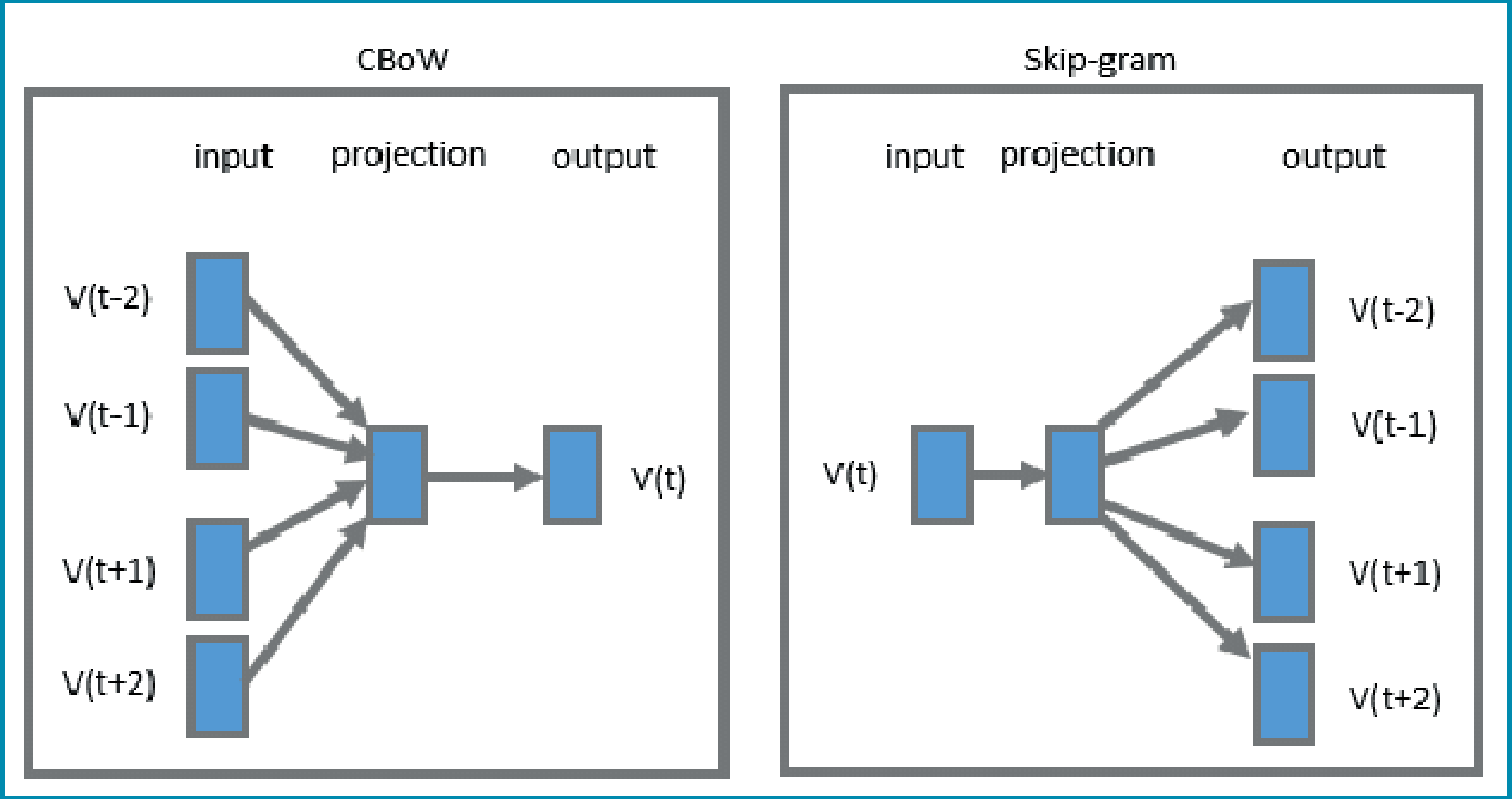
Intuition of distributional word similarity

- Suppose we wonder what is **tesgüino**?
  - A bottle of tesgüino is on the table
  - Everybody likes tesgüino
  - Tesgüino makes you drunk

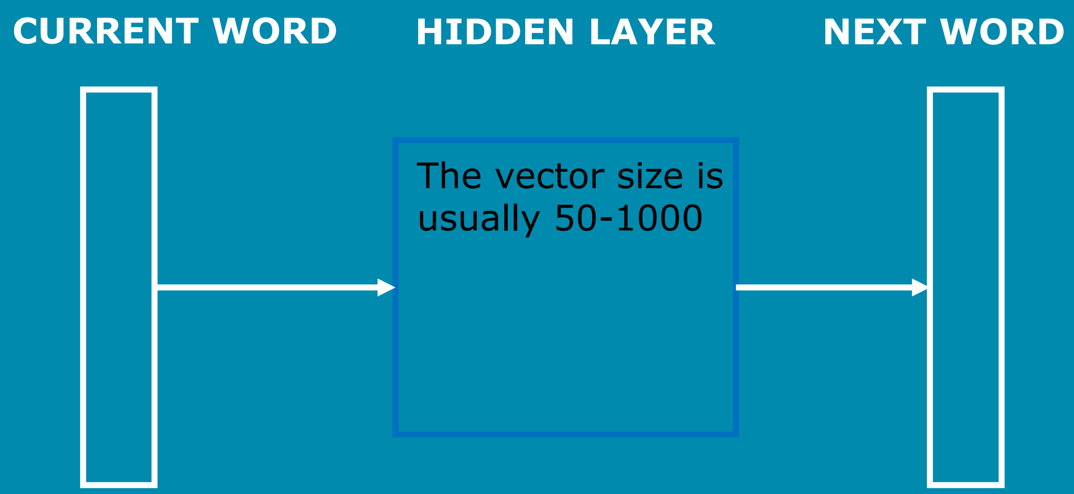
From context words, humans can guess tesgüino
an alcoholic beverage like beer

- Intuition for algorithm:
  - Two words are **similar** if they have **similar** word contexts

Word Vectors (Mikolov et. al 2013)



Word Vectors (Mikolov et. al 2013)

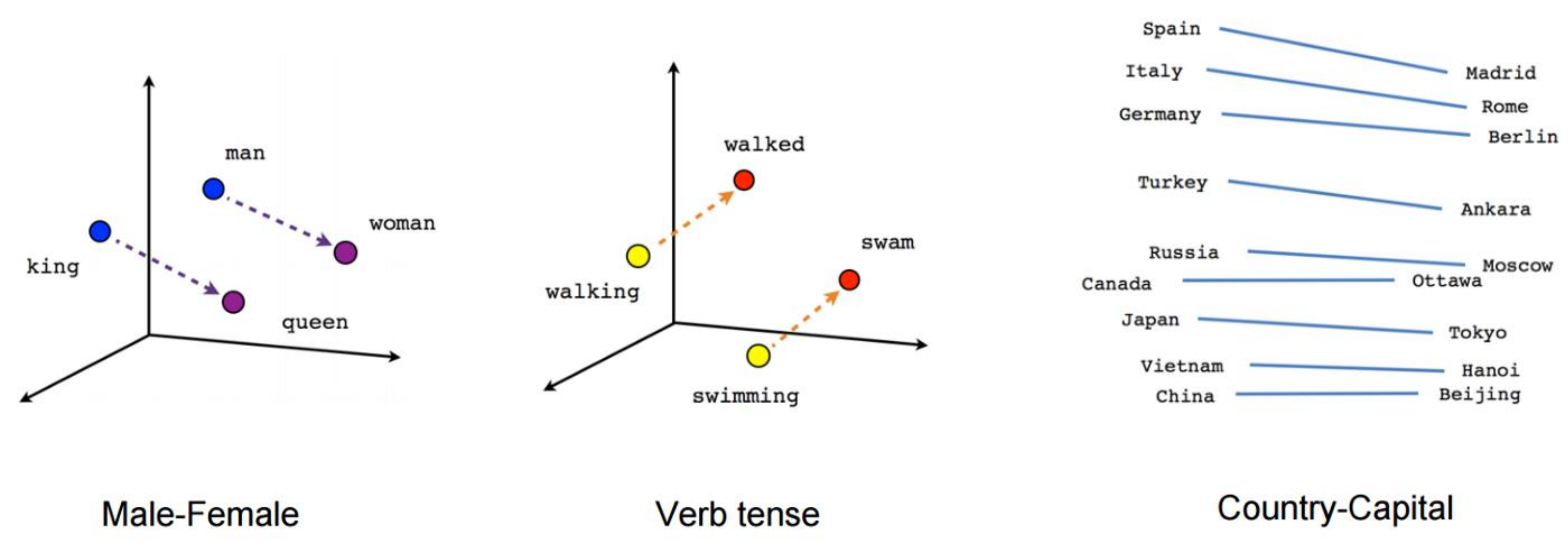


> We call the vectors in the matrix between the input and hidden layer *word vectors* (also known as word embeddings)

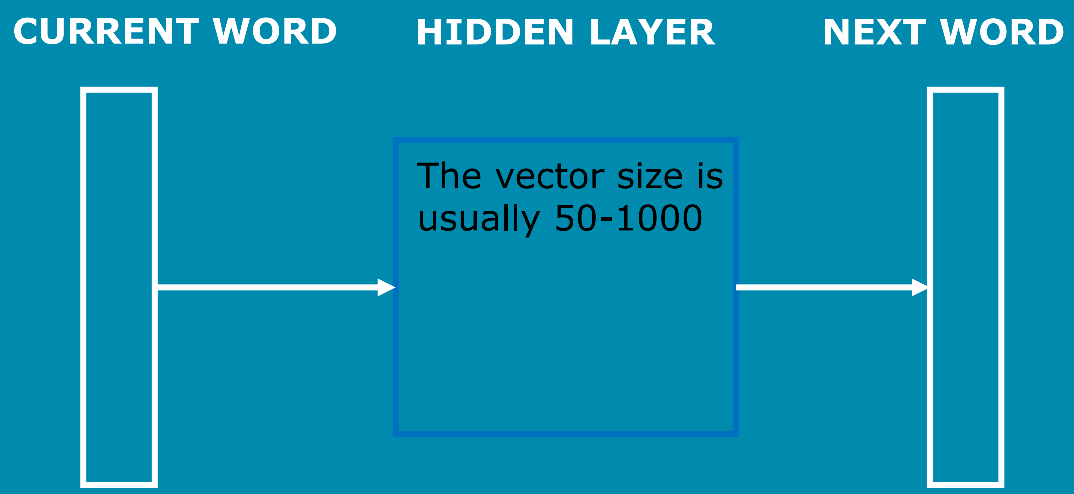
> The word vectors have similar properties to word classes (similar words have similar vector representations)

Word Vectors – Semantic Properties

The distance can be calculated with L2-distance or cosine similarity function.



Word Vectors



- The word vectors can be used as feature-inputs for many NLP tasks.
- Word vectors are trained in a ***semi-supervised*** way

Recommended Read: [The Illustrated Word2vec – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)]

Word Embedding - Demo

- Embedding projector - visualization of high-dimensional data (tensorflow.org)

Word2Vec

> A variation of the word2vec is actually matrix factorization (SVD)
  - Levy & Goldberg – 2014
  Neural Word Embedding as Implicit Matrix Factorization (nips.cc)

Word Vectors

This is only the beginning.

- Other frameworks for Word Vectors:
  - FastText – character based
  - BytePair Embedding (BPE) – frequent sub-words (letters that often appear together)

- Contextual Embedding:
  - ELMo
  - ULMFiT
  - BERT
  - RoBERTa
  - GPT

Language Model - Revisited

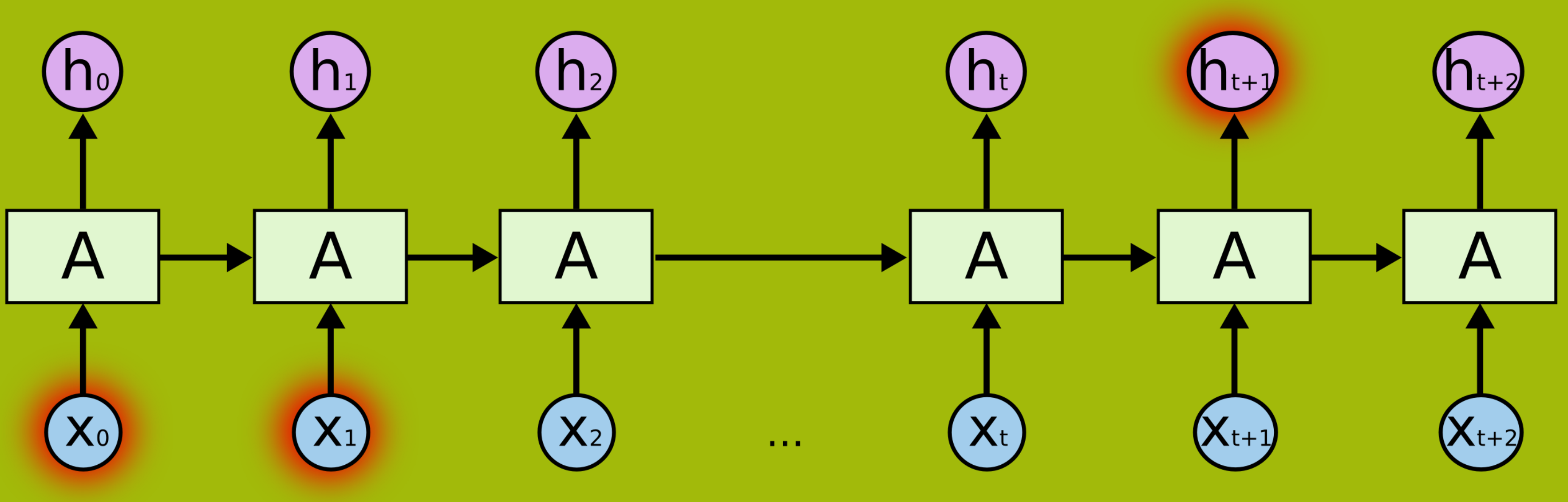
- In the original word2vec / Language Modeling we predicted:
  - Which token would come after a context
  - Which token would come in a filled <mask> (cloze style)
  - Which tokens are the context of a given token

- What other methods can you think of?

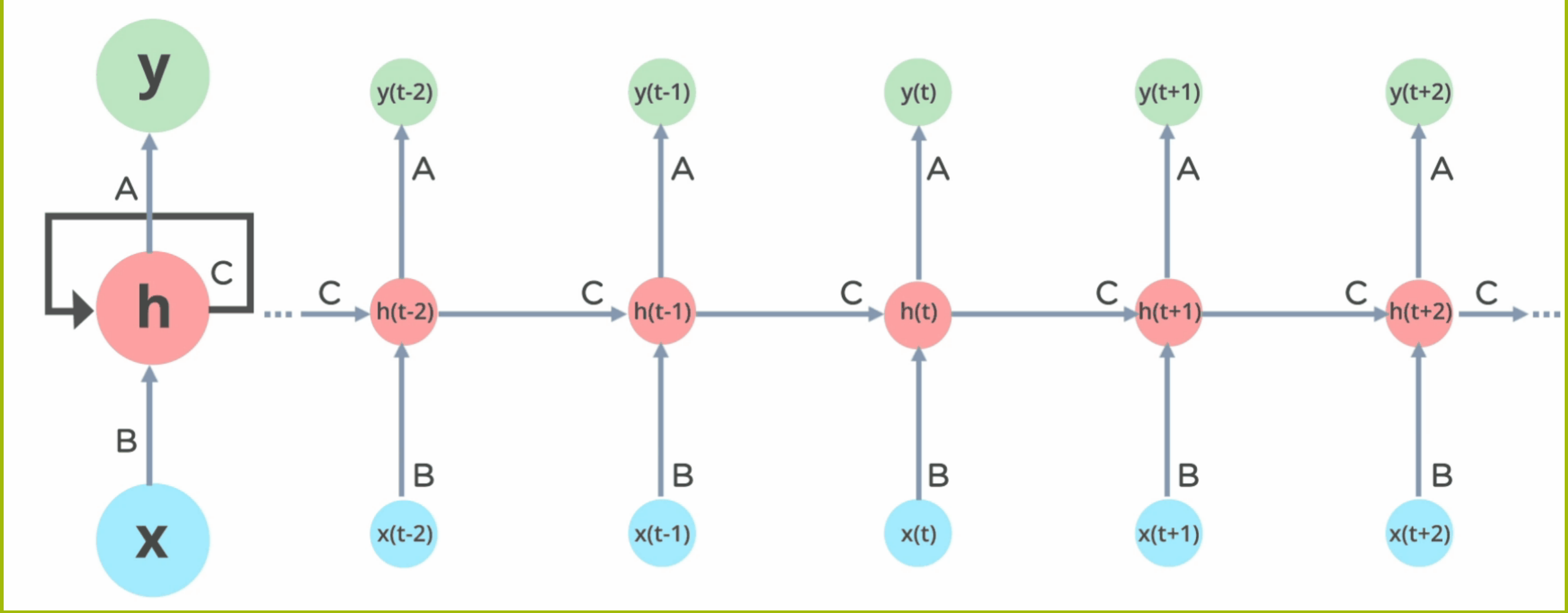
FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Advanced Deep Learning

- For NLP, a common architecture is called RNN or LSTM
- In every step, the network output is re-used as an additional input for the next step.



Advanced Deep Learning – RNN/LSTM



RNN/LSTM

- > Feed-Forward Neural Networks:
  - > Fixed input
  - > Fixed output
- > LSTM/RNN
  - > Variable length input
  - > Variable length output

- > When should we use FFNN / LSTM?
  For which cases?

NN Usage

- Topic classification
- Sentence tagging
- Translation
- Question Answering
- Intent classification
- Summarization
- Search Engine

Additional Resources

- https://lena-voita.github.io/nlp_course.html
- Animated Explanation of Feed Forward Neural Network Architecture - MLK - Machine Learning Knowledge
- Word Embedding Demo: Tutorial (cmu.edu)
- WebVectors: distributional semantic models online (nlpl.eu)

- Embedding projector - visualization of high-dimensional data (tensorflow.org)

Take aways

> Neural Networks are very variable
> Many different architectures already exists
> Is today’s #1 method of performing NLP (when there are enough resources)

> Will be the focus of M4