Machine and deep learning

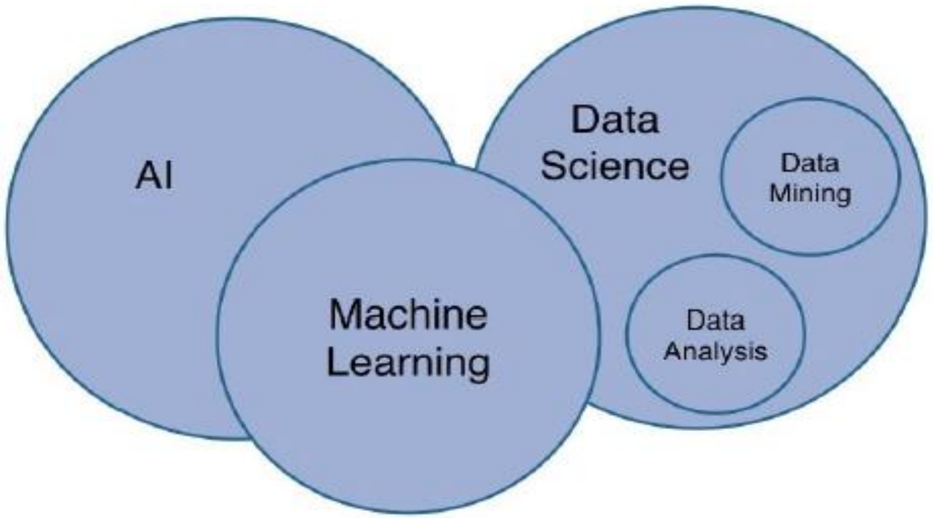
Plan

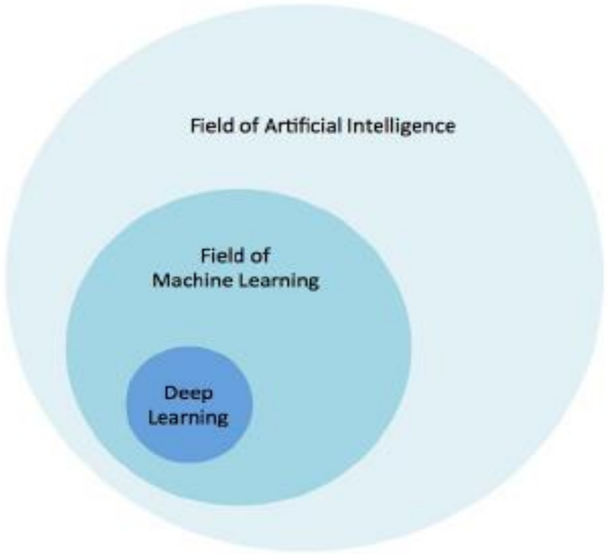
- Basics of machine learning (and libraries)
- Basics of deep learning (and libraries)
- Reinforcement learning
- Deep learning (Transformers, Graph Neural Networks, Decision Transformer)
- Optimization algorithms and libraries
- Machine learning in optimization

Machine learning

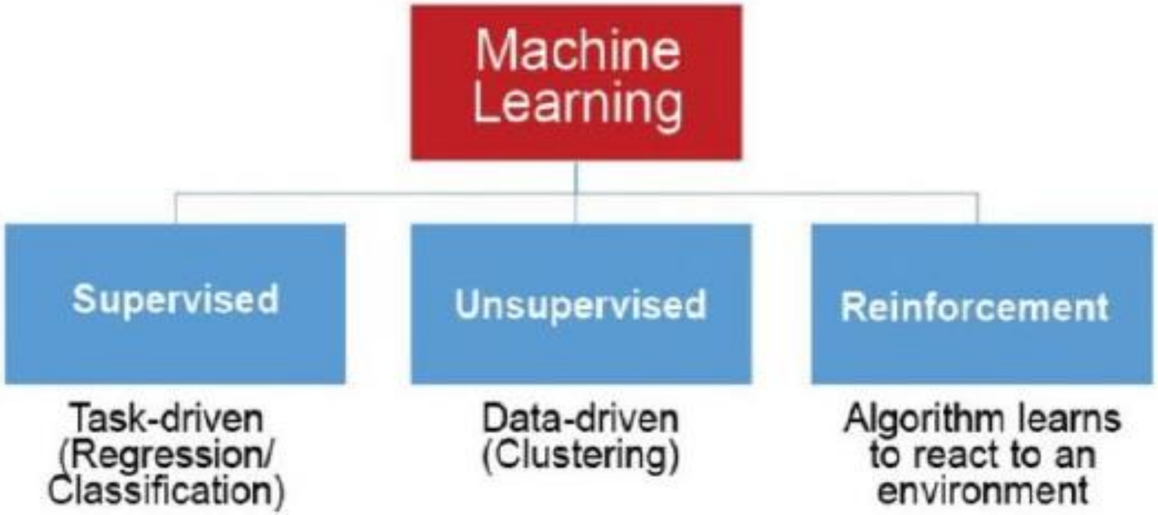
- Introduction
- Standard Methods
- Deep Learning
- Computer Vision
- Natural Language Processing
- Time Series Analysis
- Anomaly detection
- Transformers
- Large Language Models

Machine and deep learning





Machine learning taxonomy



Machine learning

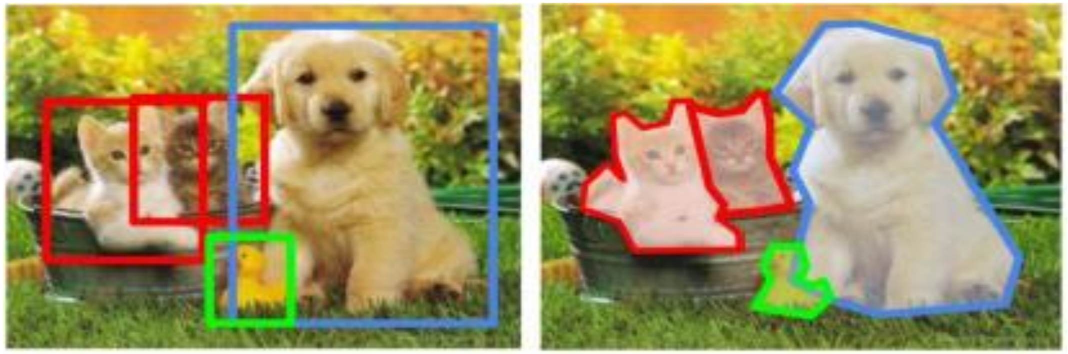
Image processing
- object detection
- classification
- segmentation

Video processing:
- classification
- segmentation

Natural language processing:
- sentiment classification
- question answering
- text summarization
- language generation

Anomaly detection

Time series analysis



NLP and machine learning

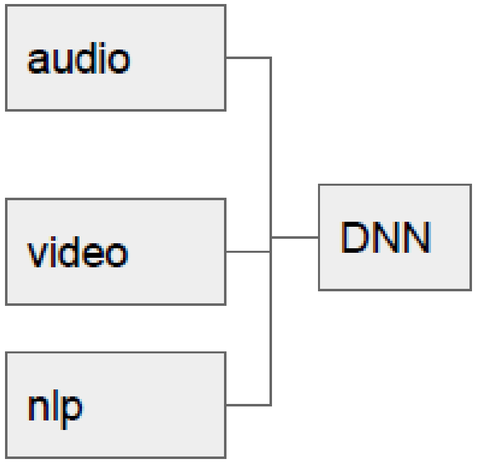
- Natural language processing
- Intersection of lingusitics and AI
- How to represent the text (vector models)
- Word emebeddings
- LLM models

Modality in machine learning

- Multimodal models

The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.



Models are based on probabilistic theory

- Generative models - joint probability $p(x,y)$
- Discriminative models - conditional probability $p(y|x)$

Example:

$(x,y) = \{(0,0), (0,0), (1,0), (1,1), (0,1), (1,1)\}$

$p(x=0,y=0)=1/3$, $p(x=1,y=0)=1/6$, $p(x=0,y=1)=1/6$, $p(x=1,y=1)=1/3$

$p(y=0|x=0)=2/3$, $p(y=0|x=1)=1/3$, $p(y=1|x=0)=1/3$, $p(y=1|x=1)=2/3$

Next taxonomy

- Generative models (Naive bayes, Boltzmann machine, etc.)
- Discriminative models (Neural networks, Logistic regression, SVM, etc.)

Machine learning - repositories and materials

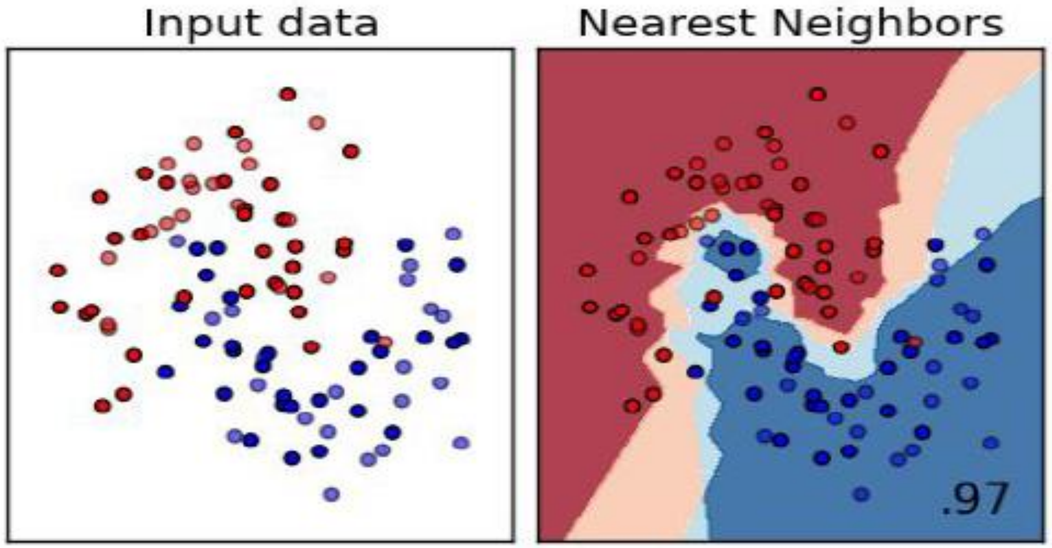
- Peter Harrington, Machine learning in action
  https://github.com/pbharrin/machinelearninginaction3x
- scikit learn - https://scikit-learn.org/stable/

K-NN nearest neighbours

https://scikit-learn.org/dev/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

- distance metric can be a parameter

$$d = \sqrt{(xA_0 - xB_0)^2 + (xA_1 - xB_1)^2}$$

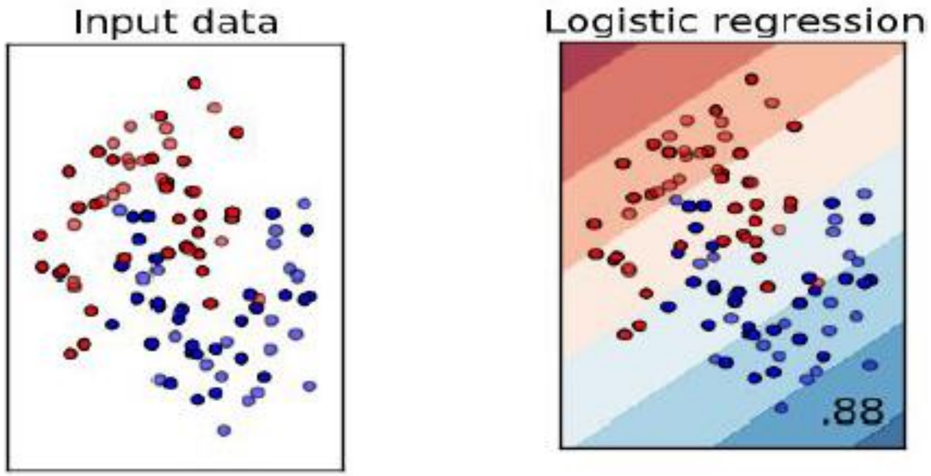


Logistic regression

https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LogisticRegression.html

- logistic/sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad z = w_0 x_0 + w_1 x_1 + w_2 x_2 + \cdots + w_N x_N$$



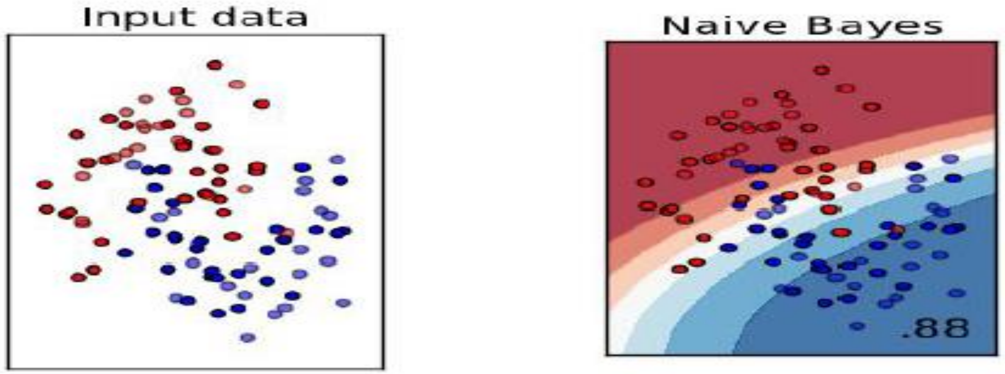
Naive Bayes

https://scikit-learn.org/1.5/modules/naive_bayes.html

Bayes probability based classifier:

- Probability of feature inside class
- Probability of class

$$p(c_i|w) = \frac{p(w|c_i)p(c_i)}{p(w)}$$

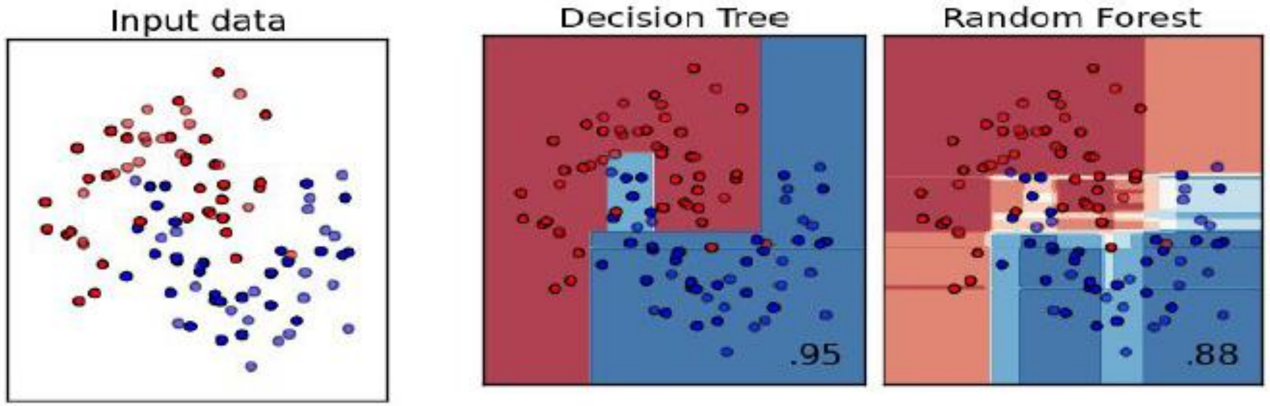


Decision trees

https://scikit-learn.org/1.5/modules/tree.html

Entropy based index to choose the feature for node selection:

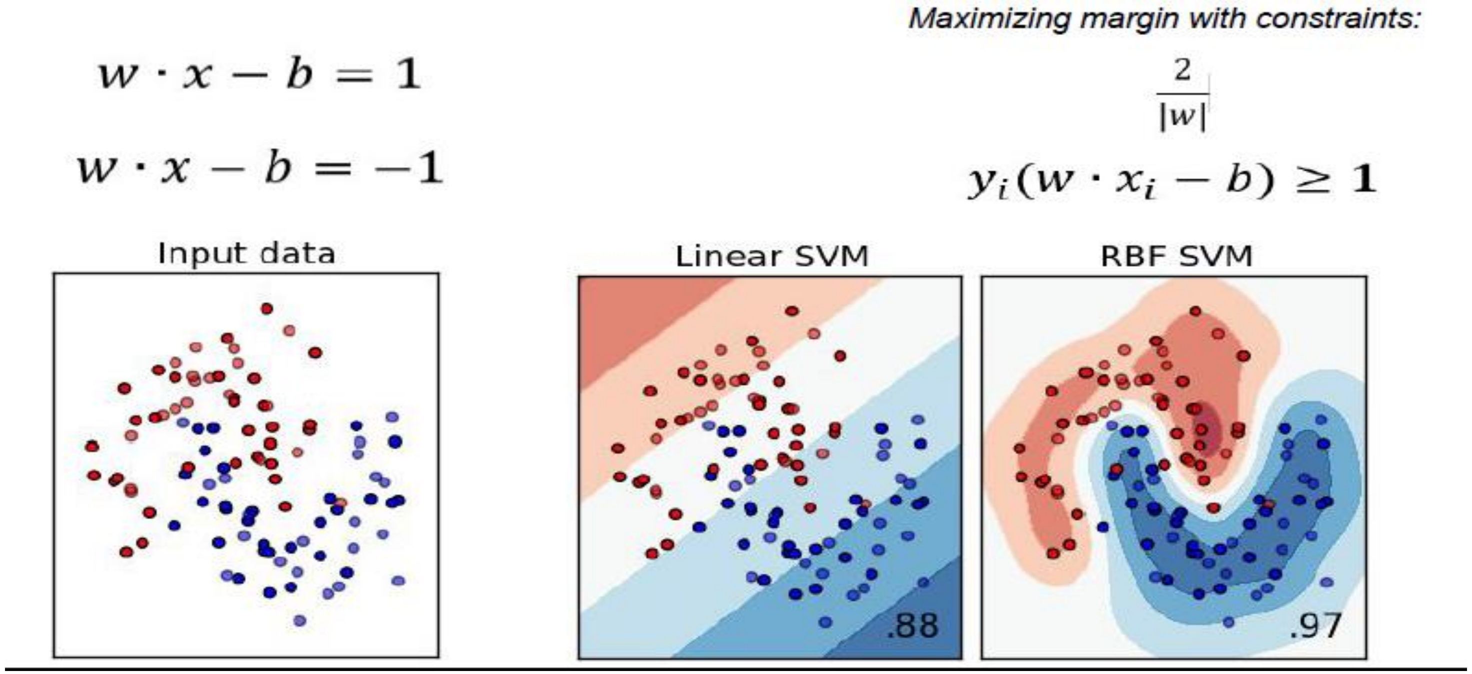
$$H = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$



SVM - support vector machine

https://scikit-learn.org/1.5/modules/svm.html

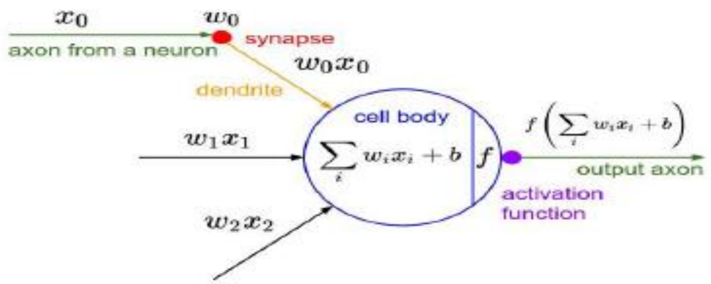
Finding margin which seperate points from different classes:

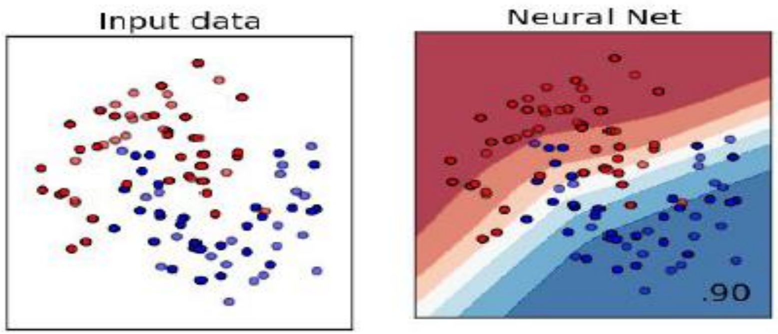


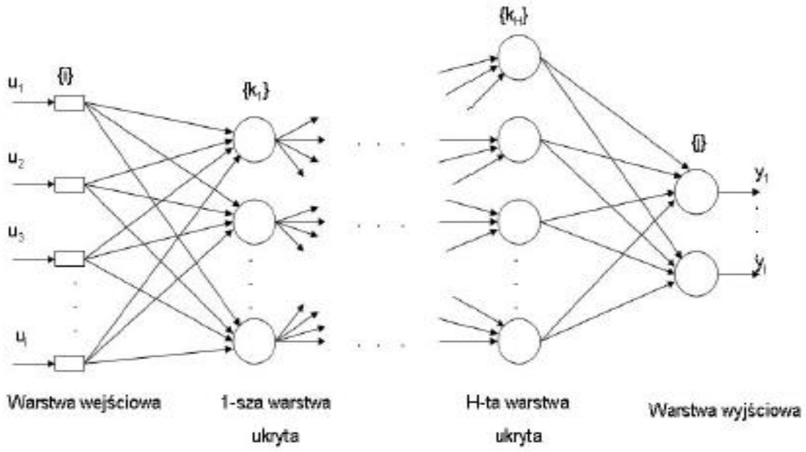
MLP - multi-layer perceptron

https://scikit-learn.org/1.5/modules/neural_networks_supervised.html

- three layer network: input, hidden, output layer
- activation function in each neuron





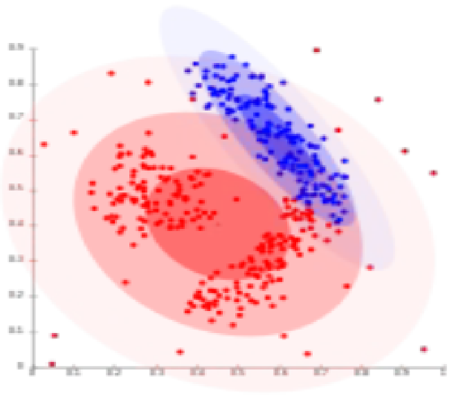


Unsupervised learning - clustering

- https://scikit-learn.org/stable/unsupervised_learning.html

Assigning unseen data to classes

- K-means
- Dbscan
- Hierarchical clustering (c-link)



Models improvements

- Bagging – RandomForest

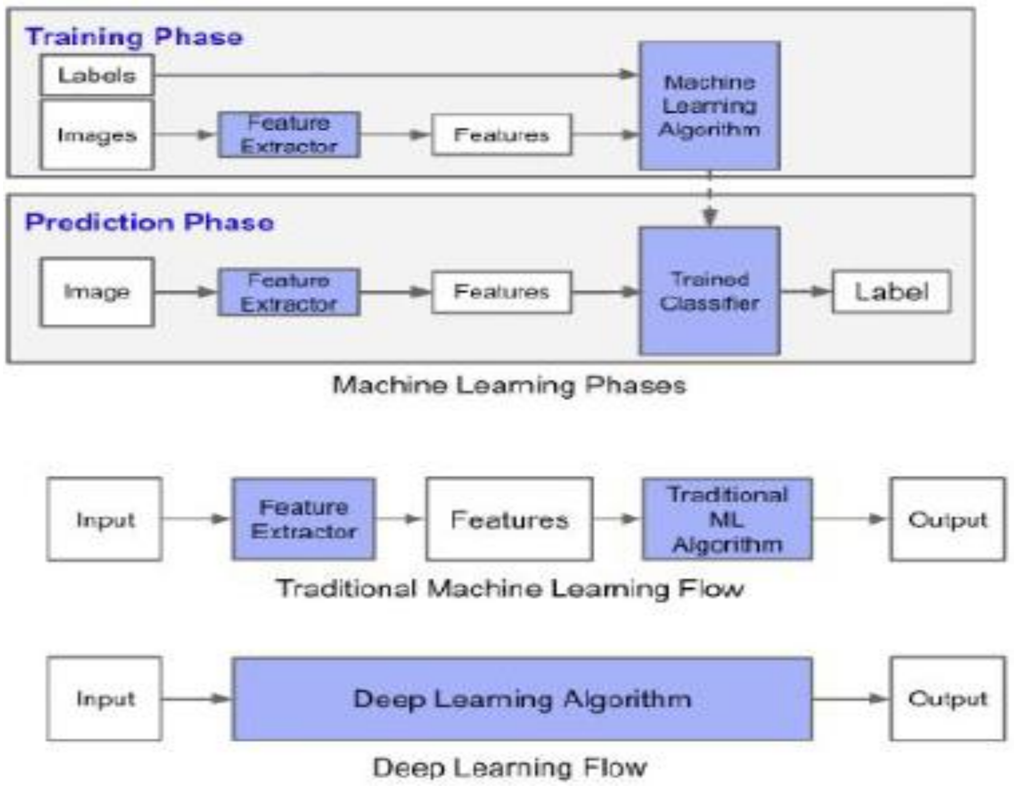
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

- Boosting – AdaBoost

https://scikit-learn.org/dev/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

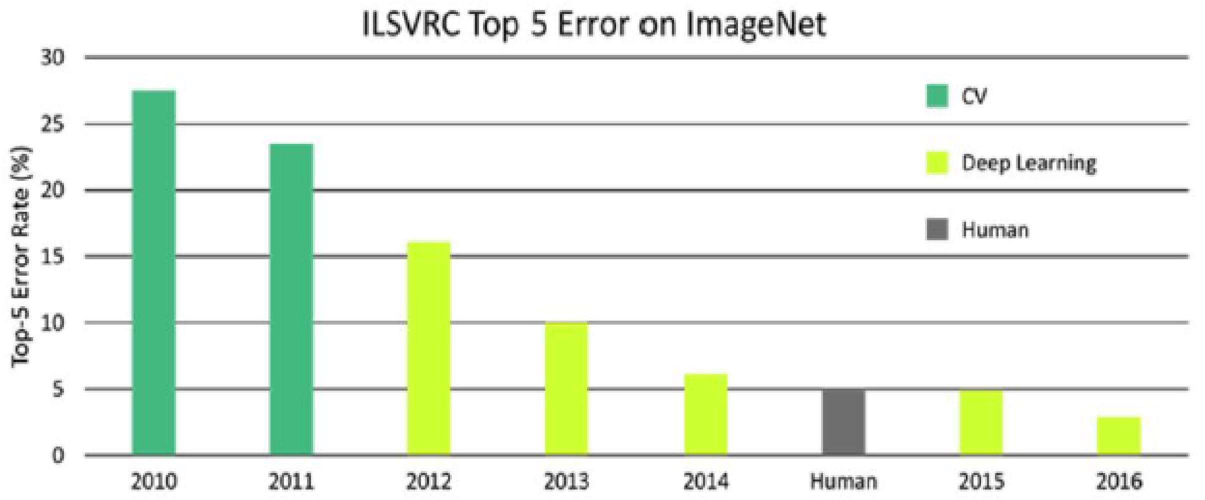
Deep learning

- Number of layers
- Feature extraction
- Complexity
- Static and sequence models



https://cs231n.github.io/convolutional-networks/

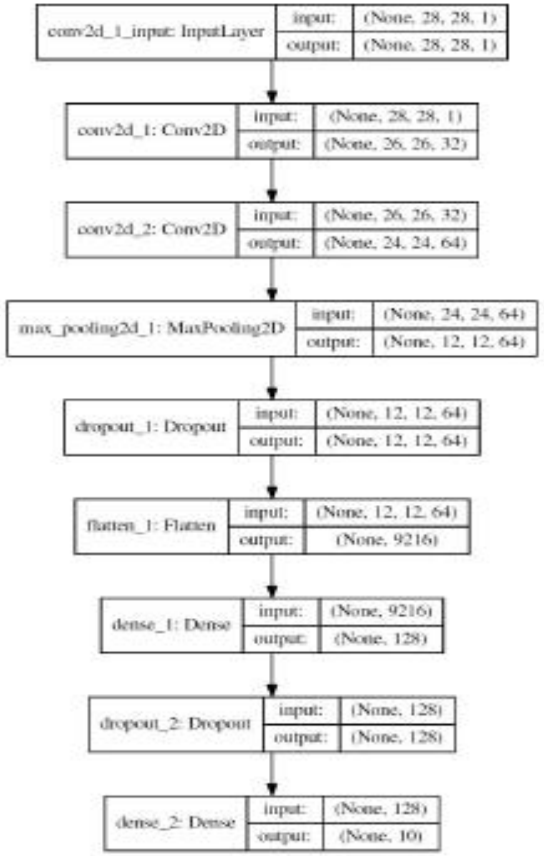
Deep learning vs machine learning



source: https://www.dsiac.org/resources/journals/dsiac/winter-2017-volume-4-number-1/real-time-situ-intelligent-video-analytics

Convolutional model

- Convolution operation - feature extractor
- Fully connected layers
- Non-linear layers
- Pooling
- Normalization layers

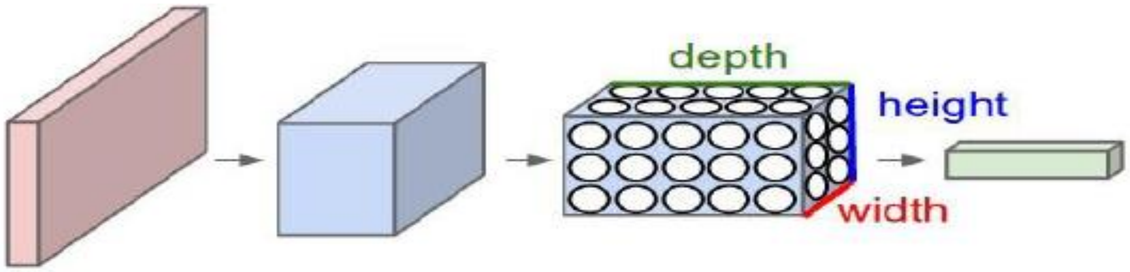


https://cs231n.github.io/convolutional-networks/

Convolutional layers

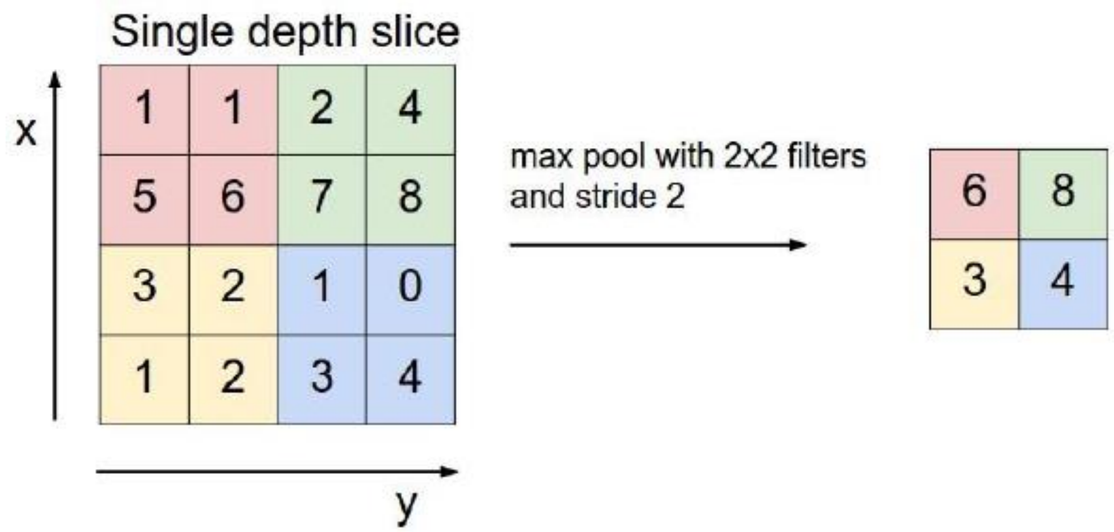
Main features:

- Input depth
- Output depth
- Height
- Width



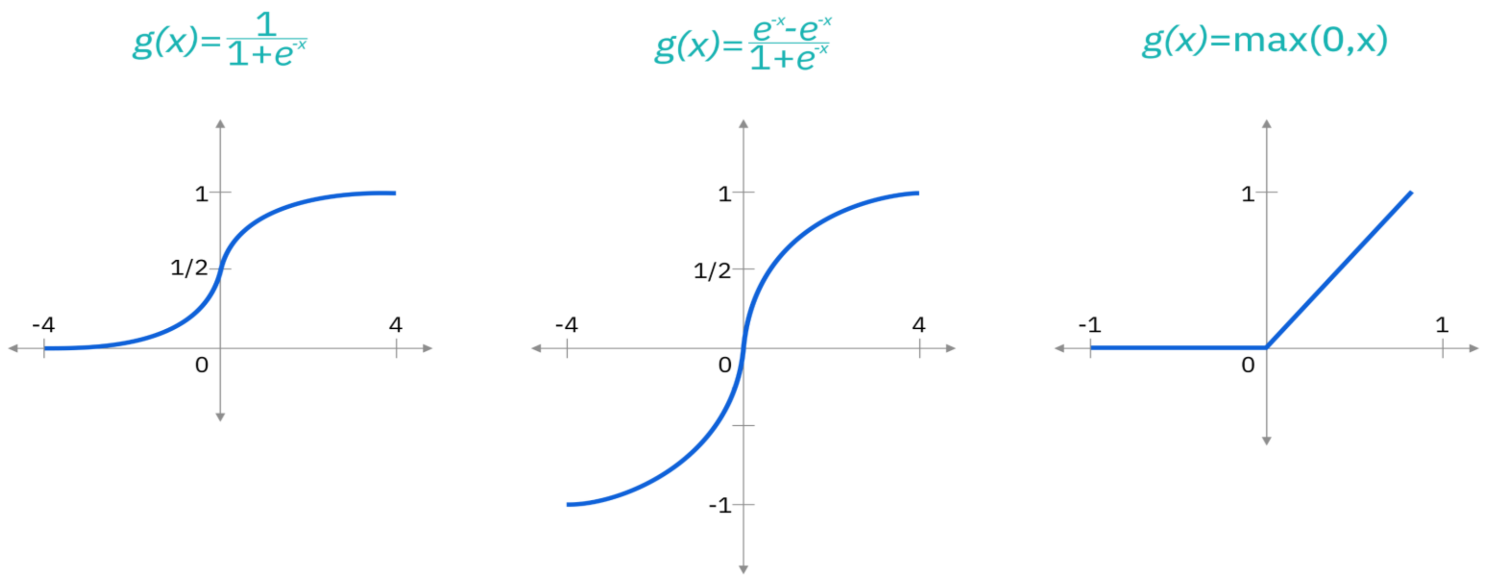
Pooling

- Max pooling, average pooling etc.



Activation functions

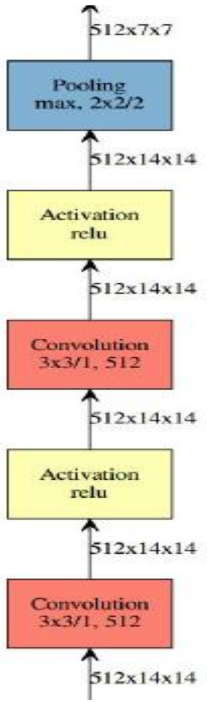
- Sigmoid, ReLU, pReLU, Hardswish etc.

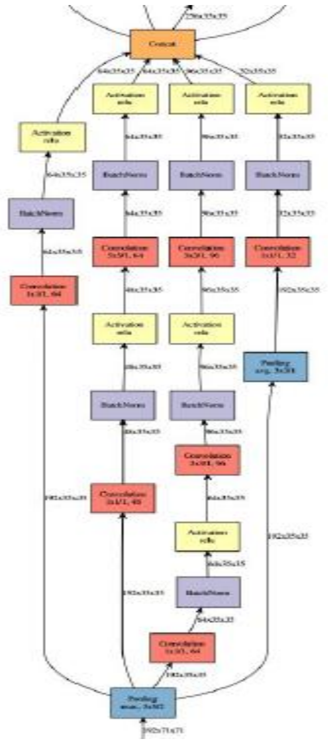


The CNN architectures

Most popular CNN architectures:

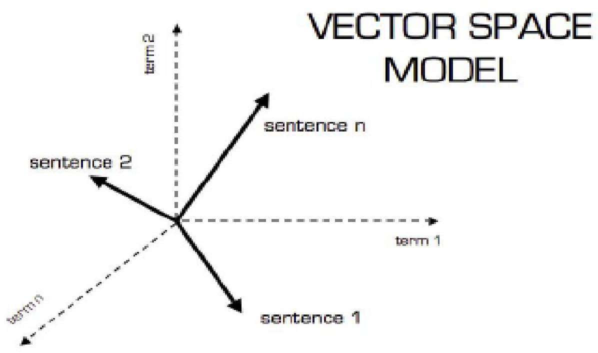
- VGG
- Resnets
- Inception
- EfficientNet





NLP - Natural Language Processing

- Text representation



Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Text representation

- TF-IDF - term frequency - inverse document freqency

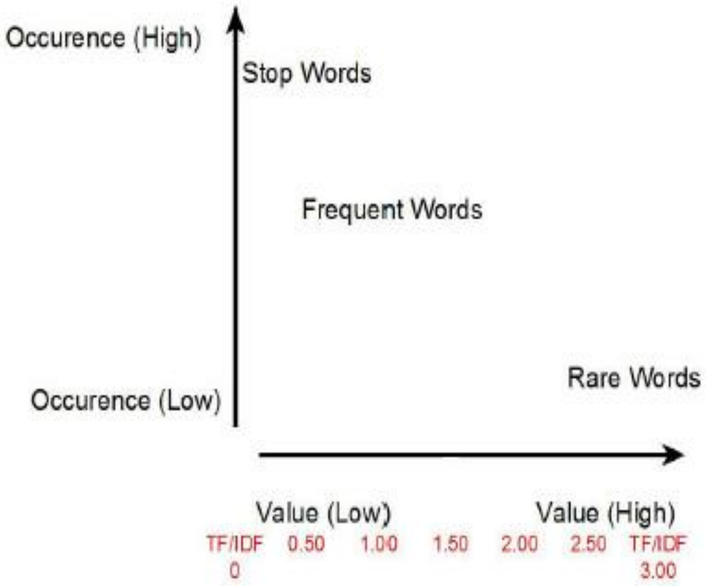
$$\boldsymbol{TF-IDF} = TF_t \times \log\frac{N}{DF_t}$$

IDF

$\boldsymbol{TF_t}$ – **Term Frequency** of the term t. (How many times does the term occur in the document?)

N – Total Number of Documents in the Corpus.

$\boldsymbol{DF_t}$ – **Document Frequency** of t. (How many documents have the term t?)



Examples

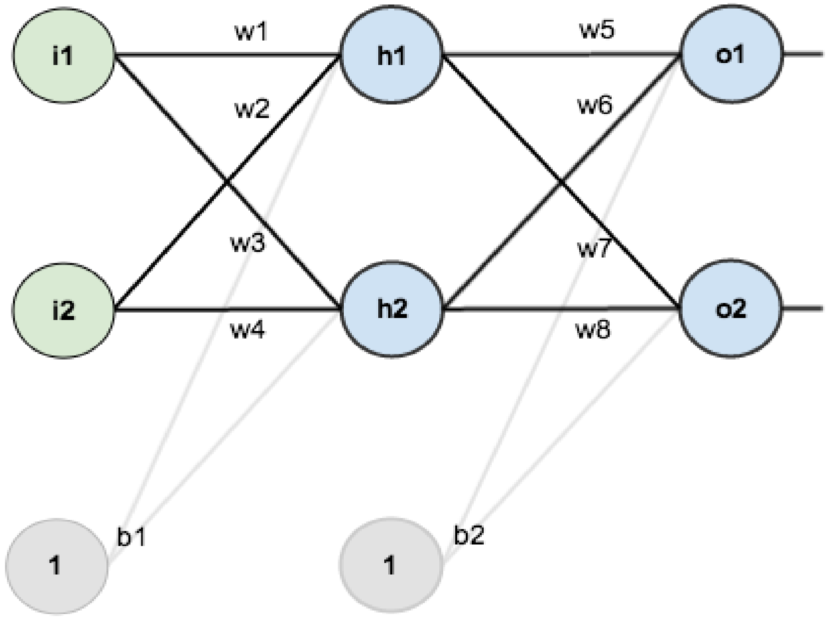
- Text classification: https://scikit-learn.org/1.4/tutorial/text_analytics/working_with_text_data.html

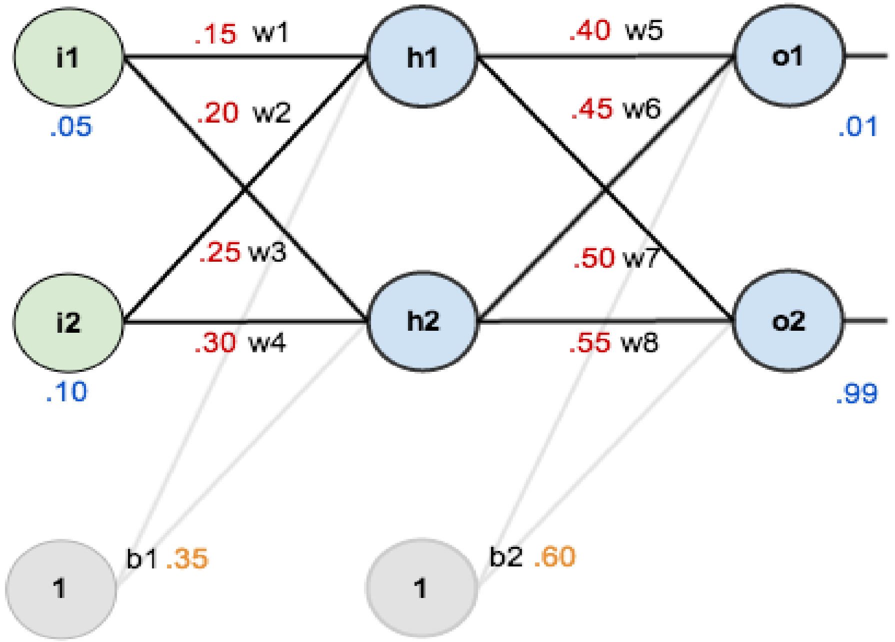
Examples

Setup neural network in pytorch:
https://pytorch.org/tutorials/recipes/recipes/defining_a_neural_network.html

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

Train the model and compute gradient





Train the model and compute gradient

Forward pass (first layer):

$$net_{h_1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h_1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

$$out_{h_1} = \frac{1}{1 + e^{-net_{h_1}}} = \frac{1}{1 + e^{-0.3775}} = 0.593269992$$

$$out_{h_2} = 0.596884378$$

Forward pass (second layer):

$$net_{o_1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$net_{o_1} = 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}} = \frac{1}{1 + e^{-1.105905967}} = 0.75136507$$

Train the model and compute gradient

Calculating the total error:

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

$$E_{total} = \sum \frac{1}{2}(target_{o_1} - output_{o_1})^2 = \frac{1}{2}(0.01 - 0.75136507)^2 = 0.274811083$$

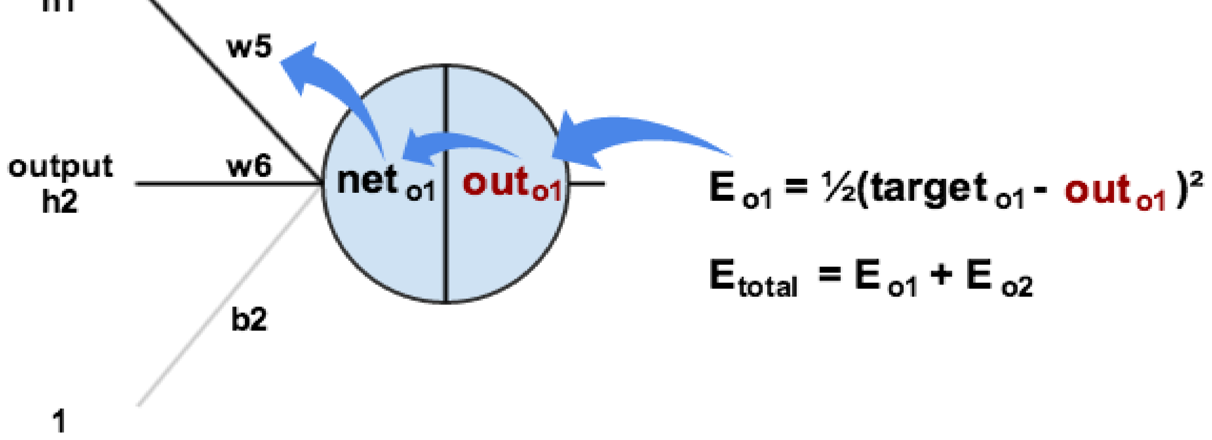
$$E_{o_2} = 0.023560026$$

$$E_{total} = E_{o_2} + E_{o_2} = 0.274811083 + 0.023560026 = 0.298371109$$

Train the model and compute gradient

Backpropagation - gradient computation

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$



Train the model and compute gradient

Backpropagation - gradient computation

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial out_{o_1}} = 2 * \frac{1}{2}(target_{o_1} - out_{o_1})^{2-1} * -1 + 0 \quad \rightarrow \quad \frac{\partial E_{total}}{\partial out_{o_1}} = -\left(target_{o_1} - out_{o_1}\right) = -(0.01 - 0.75136507)\text{=0.741365}$$

$$out_{o_1} = \frac{1}{1 + e^{-net_{o_1}}} \quad \rightarrow \quad \frac{\partial out_{o_1}}{\partial net_{o_1}} = out_{o_1}\left(1 - out_{o_1}\right) = 0.75136507 * (1 - 0.75136507) = 0.186815602$$

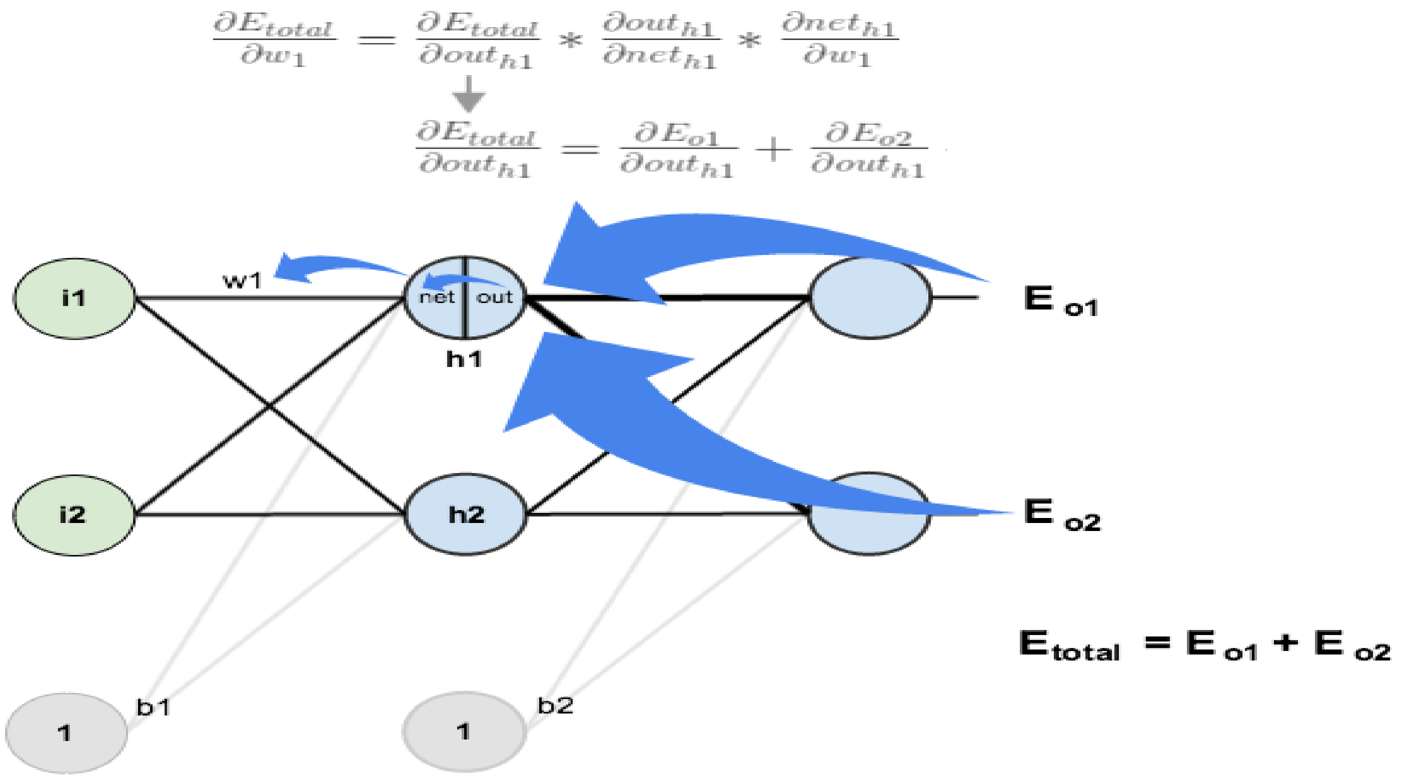
Train the model and compute gradient

$$net_{o_1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1 \quad \rightarrow \quad \frac{\partial net_{o_1}}{\partial w_5} = 1 * out_{h_1} * w_5^{(1-1)} + 0 + 0 = out_{h_1} = 0.593269992$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

$$w_5^{+} = w_5 - \alpha * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

Train the model and compute gradient



Train the model and compute gradient

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial out_{h_1}}$$

$$\frac{\partial E_{o_1}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial out_{h_1}}$$

$$\frac{\partial E_{o_1}}{\partial net_{o_1}} = \frac{\partial E_{o_1}}{\partial out_{o_1}} * \frac{\partial out_{o_1}}{\partial net_{o_1}} = 0.741365 * 0.186815602 = 0.138498562$$

Train the model and compute gradient

$$net_{o_1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$\frac{\partial net_{o_1}}{\partial out_{h_1}} = w_5 = 0.40$$

$$\frac{\partial E_{o_1}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial net_{o_1}} * \frac{\partial net_{o_1}}{\partial out_{h_1}} = 0.138498562 * 0.40 = 0.055399425$$

$$\frac{\partial E_{o_2}}{\partial out_{h_1}} = -0.019049119$$

$$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{o_1}}{\partial out_{h_1}} + \frac{\partial E_{o_2}}{\partial out_{h_1}} = 0.055399425 - 0.019049119$$

Train the model and compute gradient

$$out_{h_1} = \frac{1}{1 + e^{-net_{h_1}}}$$

$$\frac{\partial out_{h_1}}{\partial net_{h_1}} = out_{h_1}\left(1 - out_{h_1}\right) = 0.59326999 * (1 - 0.59326999) = 0.241300709$$

$$net_{h_1} = w_1 * i_1 + w_3 * i_2 + b_1 * 1 \qquad \rightarrow \qquad \frac{\partial net_{h_1}}{\partial w_1} = i_1 = 0.05$$

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} * \frac{\partial out_{h_1}}{\partial net_{h_1}} * \frac{\partial net_{h_1}}{\partial w_1} \qquad \rightarrow \qquad \frac{\partial E_{total}}{\partial w_1} = 0.036350306 * 0.241300709 * 0.05 = 0.000438568$$

$$w_1^+ = w_1 - \alpha * \frac{\partial E_{total}}{\partial w_1} = 0.15 - 0.5 * 0.000438568 = 0.149780716$$

Problems in training the DL models

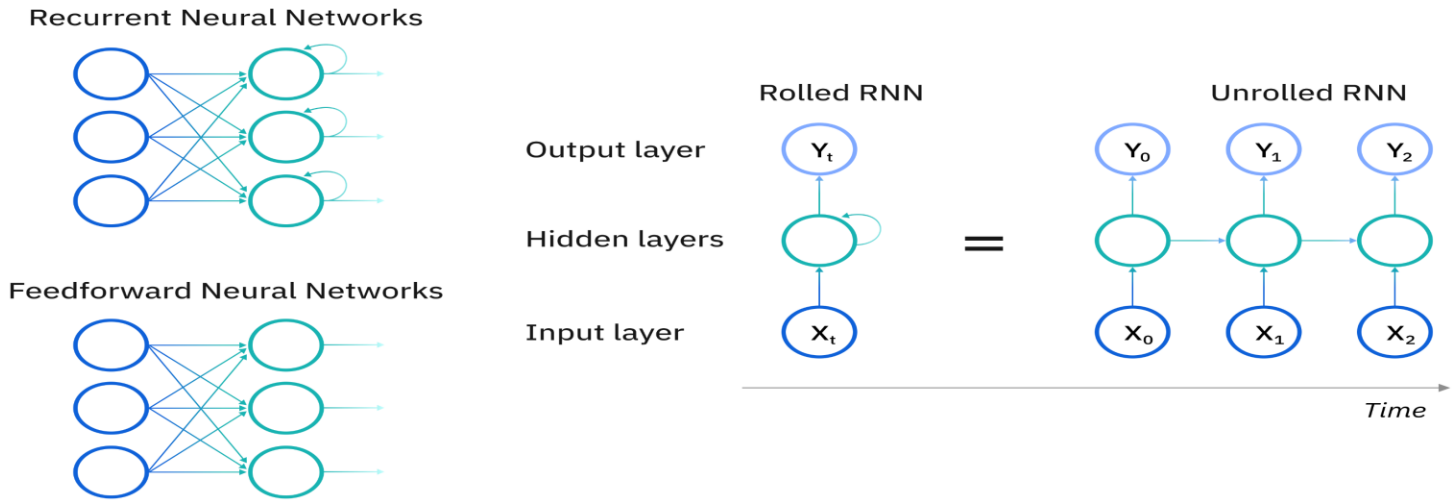
1. **Vanishing Gradient:** Text generation, machine translation, and stock market prediction are just a few examples of the time-dependent and sequential data. The gradient problem makes training difficult.
2. **Exploding Gradient:** An Exploding Gradient occurs when a neural network is being trained and the slope tends to grow exponentially rather than decay. Large error gradients lead to very large updates to the model weights

Recurrent neural network

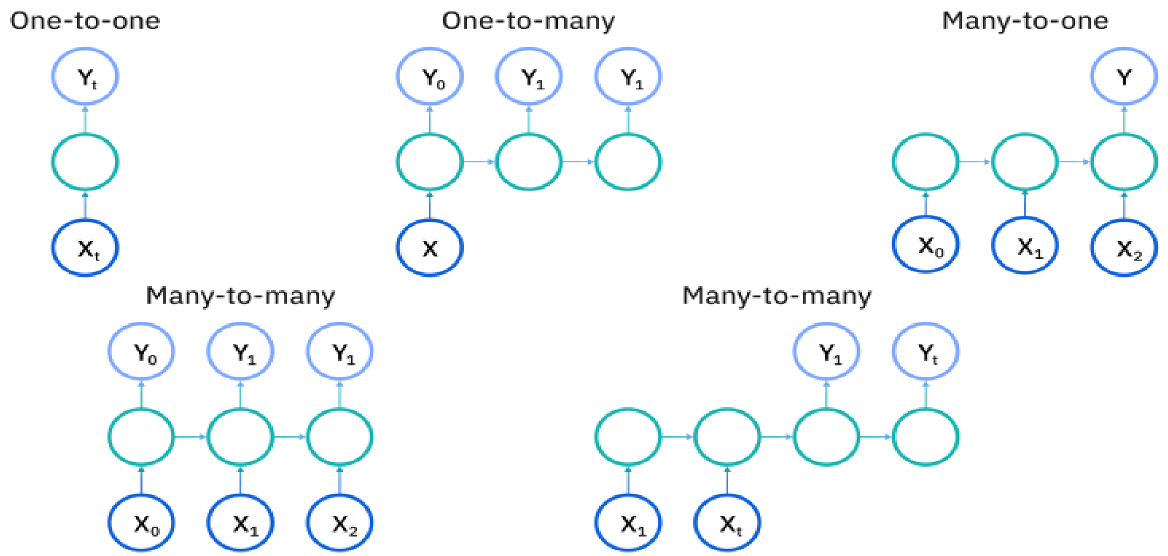
- Temporal data
- Should remember the past data
- It consists of cells
- Cells consist of gates
- Gates form a hidden state

Recurrent neural network

Architecture:



Recurrent neural network - configuration

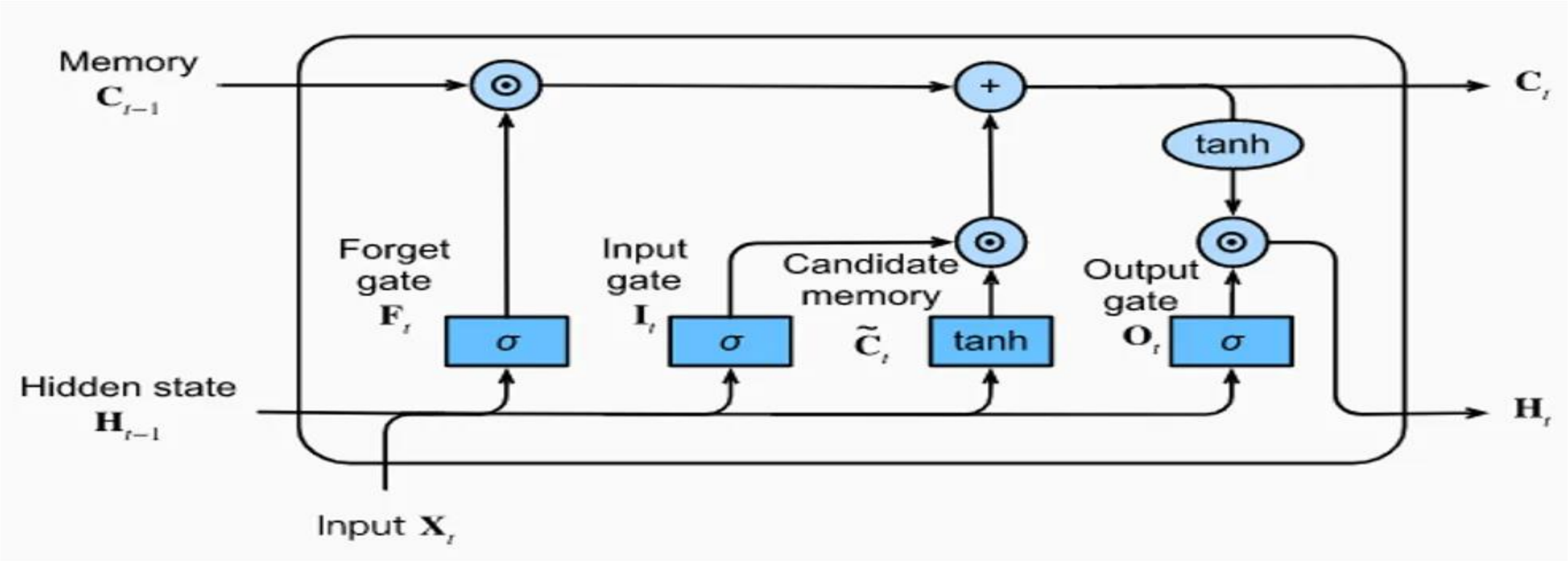


Different types of RNN

LSTM

- **Forget Gate:** LTM goes to forget gate and it forgets information that is not useful.
- **Learn Gate:** Event (current input) and STM are combined together so that necessary information that we have recently learned from STM can be applied to the current input.
- **Remember Gate:** LTM information that we haven’t forget and STM and Event are combined together in Remember gate which works as updated LTM.
- **Use Gate:** This gate also uses LTM, STM, and Event to predict the output of the current event which works as an updated STM.

LSTM architecture



$$h_t = \sigma(W^{hx}x_t + W^{hh}h_{t-1})$$

$$y_t = W^{yh}h_t$$

Training recurrent neural model

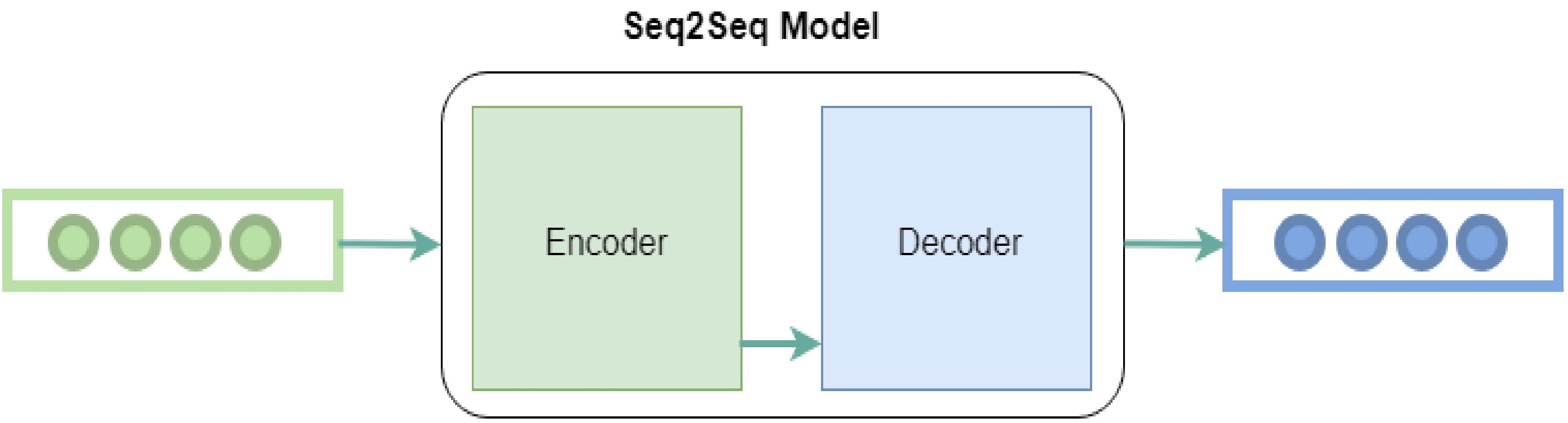
- Backpropagation through time
- Exploding gradient problem

Recurrent Neural Network

Where it can be used:

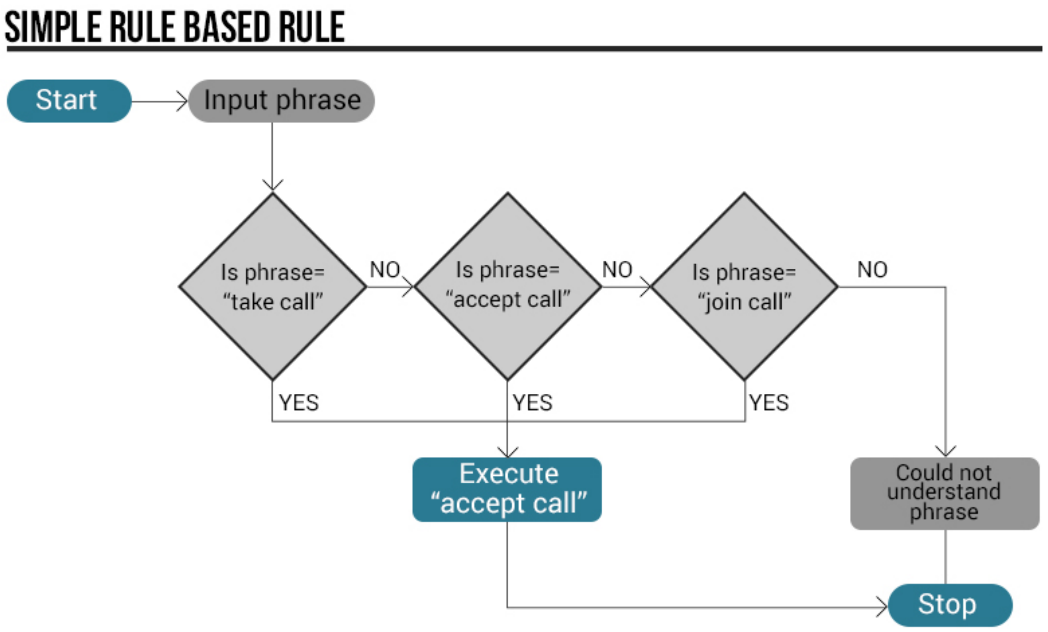
- Language modeling
- Time series prediction
- Language translation
- Video analysis

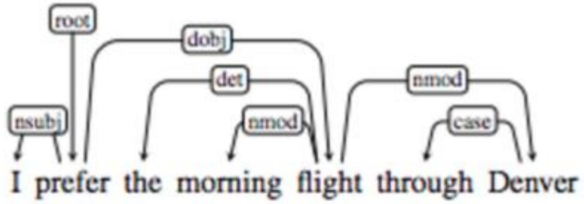
Sequence-to-sequence model

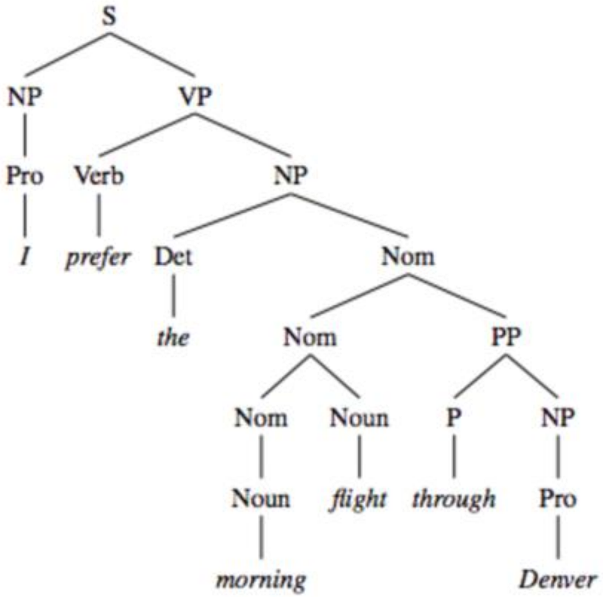


Linguistic Foundations

- Rule-based approaches
- Semantic parsing
- Analyzing linguistic structure and grammars of text

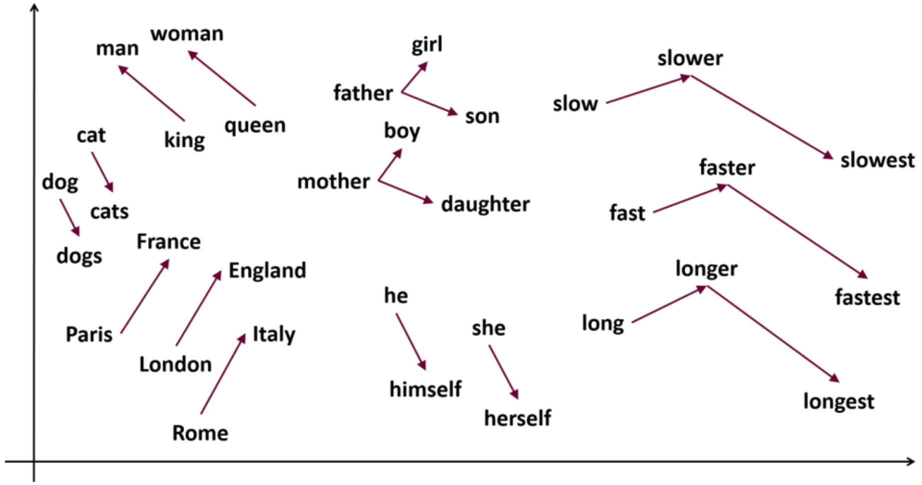




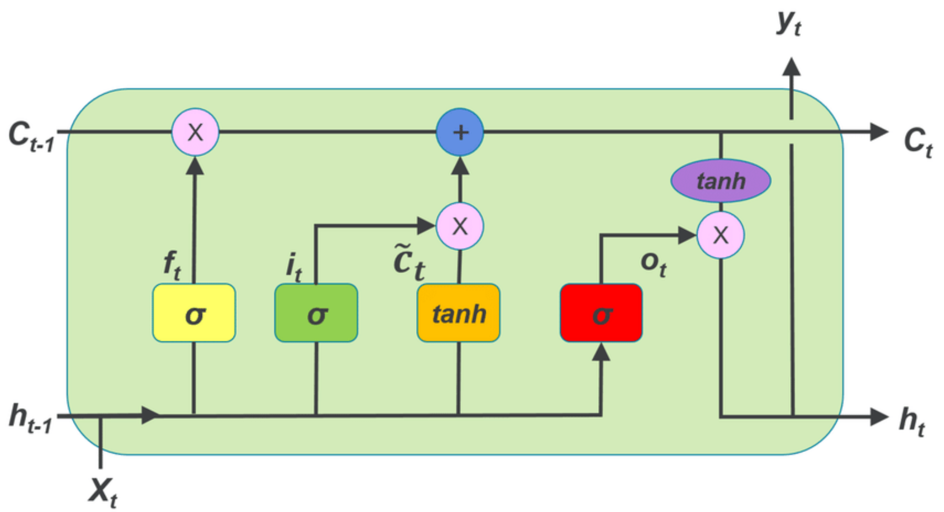


Word Embeddings

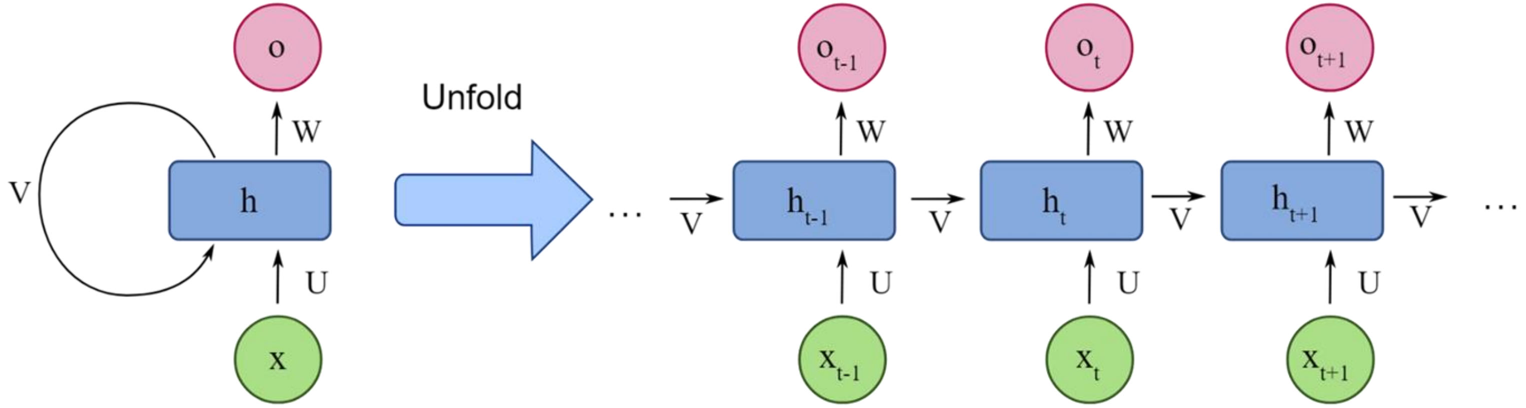
- Represent each word as a "vector" of numbers
- Converts a "discrete" representation to "continuous", allowing for:
  - More "fine-grained" representations of words
  - Useful computations such as cosine/eucl distance
  - Visualization and mapping of words onto a semantic space
- Examples:
  - Word2Vec (2013), GloVe, BERT, ELMo



Seq2seq Models



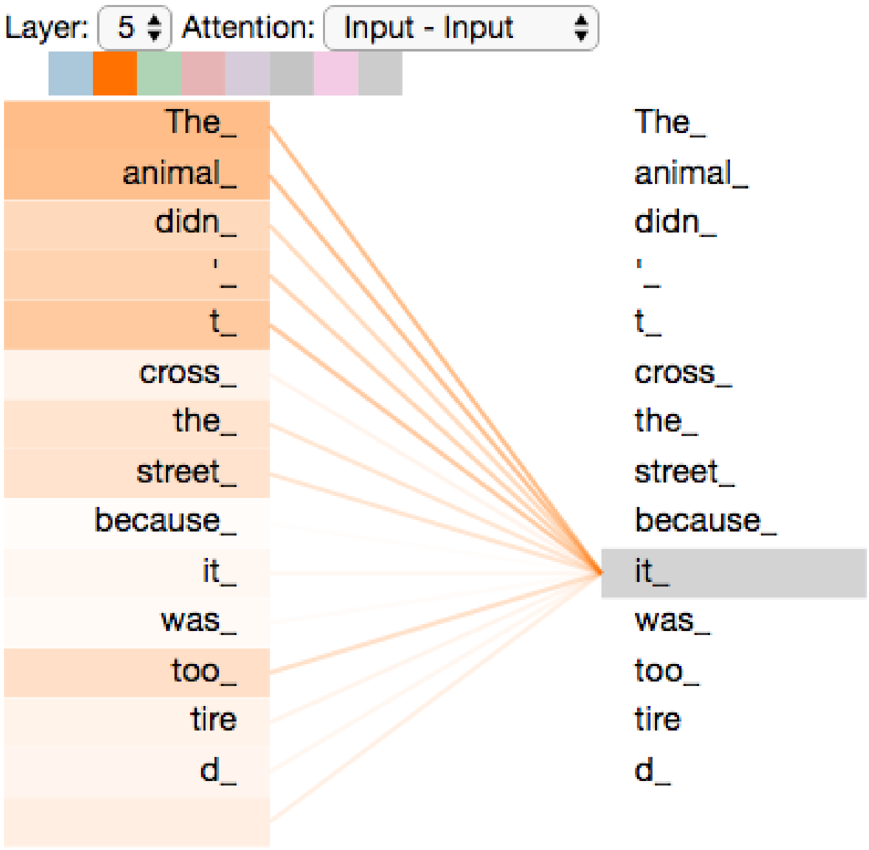
- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory Networks (LSTMs)
- "Dependency" and info between tokens
- Gates to "control memory" and flow of information



Attention and Transformers

- Allows to “focus attention” on particular aspects of the input text
- Done by using a set of parameters, called "weights," that determine how much attention should be paid to each input at each time step
- These weights are computed using a combination of the input and the current hidden state of the model
- Attention weights are computed (dot product of the query, key and value matrix), then a softmax function is applied to the dot product

$$\mathrm{attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



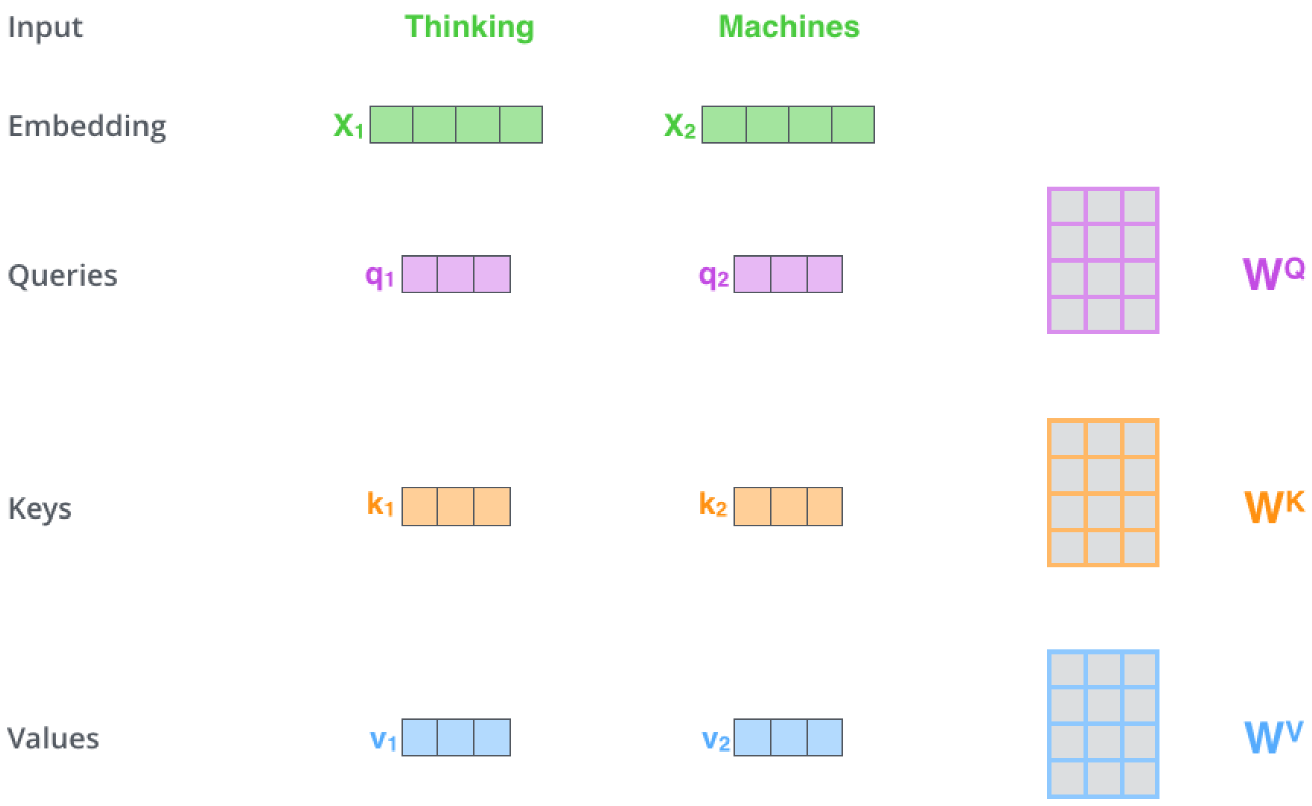
https://arxiv.org/abs/1706.03762
https://jalammar.github.io/illustrated-transformer/

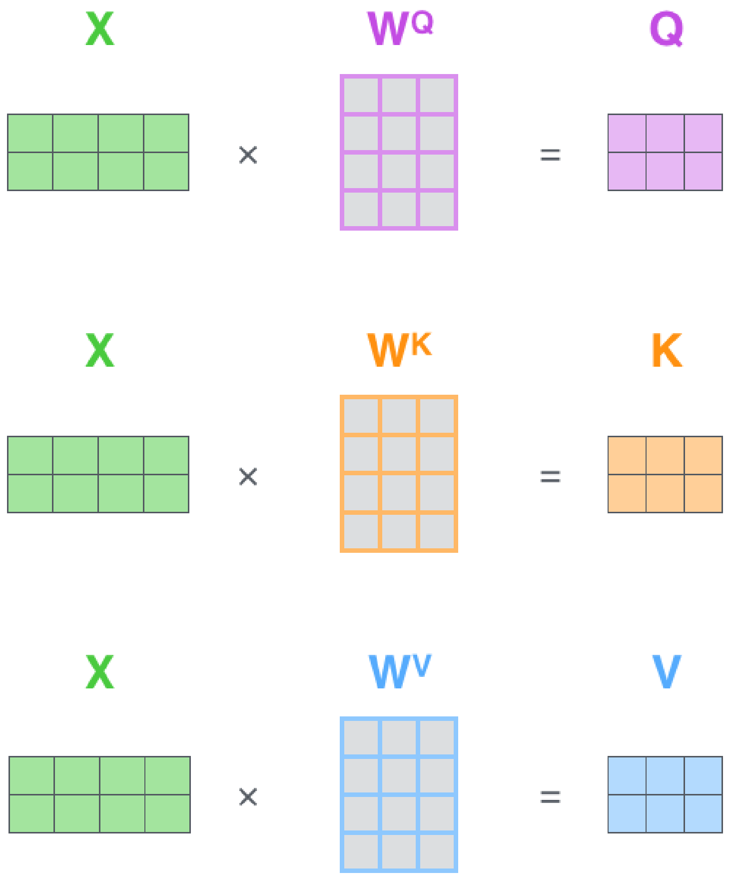
Analogy for Q, K, V

- Library system
- Imagine you're looking for information on a specific topic (query)
- Each book in the library has a summary (key) that helps identify if it contains the information you're looking for
- Once you find a match between your query and a summary, you access the book to get the detailed information (value) you need
- Here, in Attention, we do a “soft match” across multiple values, e.g. get info from multiple books (“book 1 is most relevant, then book 2, then book 3, etc.”)

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Self-Attention





https://jalammar.github.io/illustrated-transformer/

Transformer & Multi-Head Attention

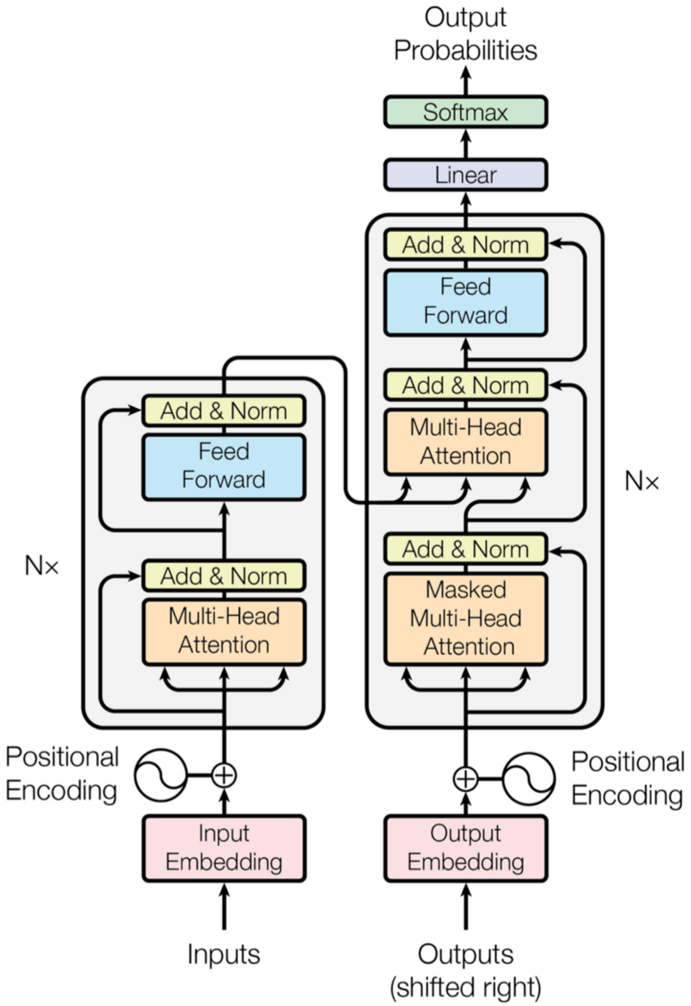


Figure 1: The Transformer - model architecture.

"Attention Is All You Need"
https://arxiv.org/abs/1706.03762

Multi-Head Attention

1) This is our input sentence*

2) We embed each word*

3) Split into 8 heads. We multiply X or R with weight matrices

4) Calculate attention using the resulting Q/K/V matrices

5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer