COMP7180: Quantitative Methods for Data Analytics and Artificial Intelligence

Lecture 11: Mathematics in Deep Learning

Jun Qi

Research Assistant Professor in Computer Science @ Hong Kong Baptist University

Affiliated Associate Professor in Electronic Engineering @ Fudan University

香港浸會大學
HONG KONG BAPTIST UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

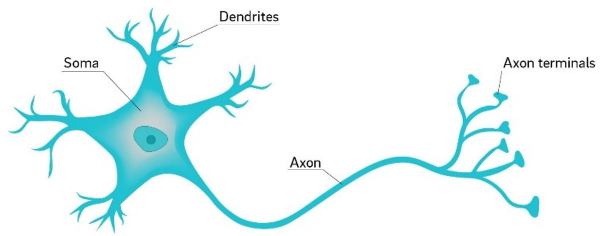
03/12/2024

Neural Network

- A computational model that can learn.
- A model with parameters.
- Learns the parameters from the data.

Round Shape
Red color
Small Size

Neuron



APPLE

How Neural Networks Learn

Human

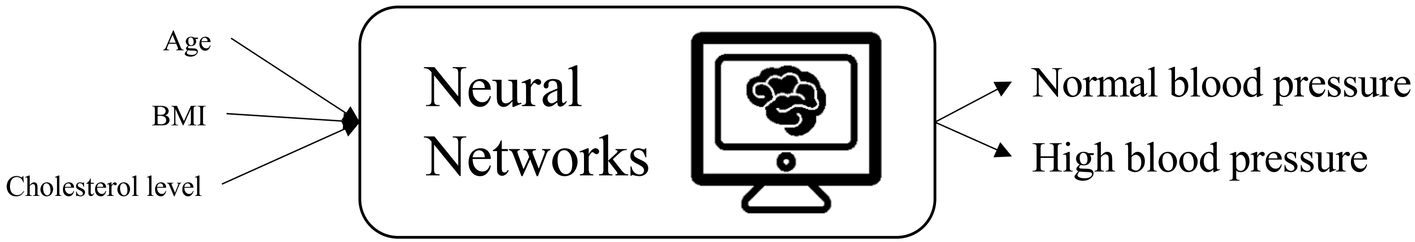
Information

Computer

- Reading
- Past Experience

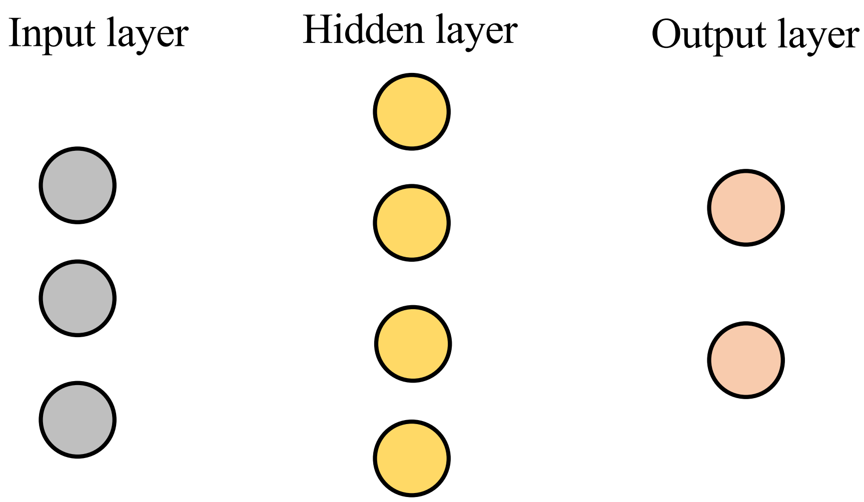
Neural Networks

- Data:
  - Input (Age, BMI, Cholesterol level)
  - Output (Normal / High Blood Pressure)
- We teach the model using the data to make accurate prediction by optimizing parameters



Neural Networks Basics

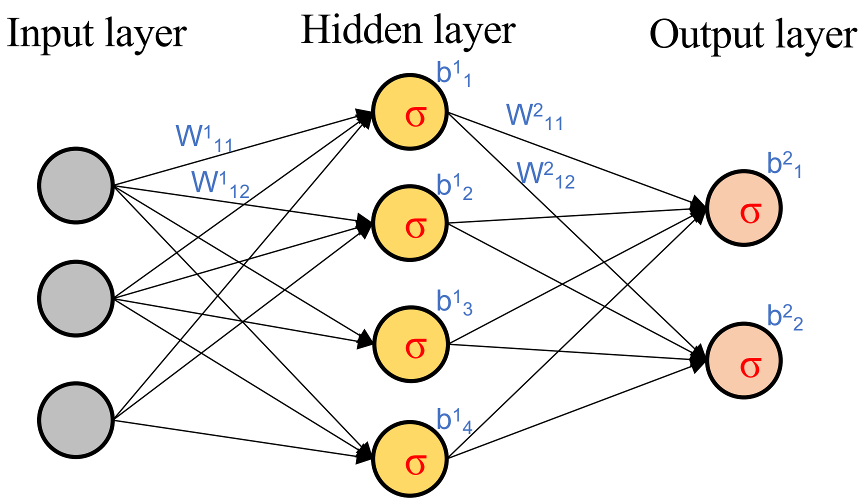
- Made up of <u>multiple layers of nodes</u>



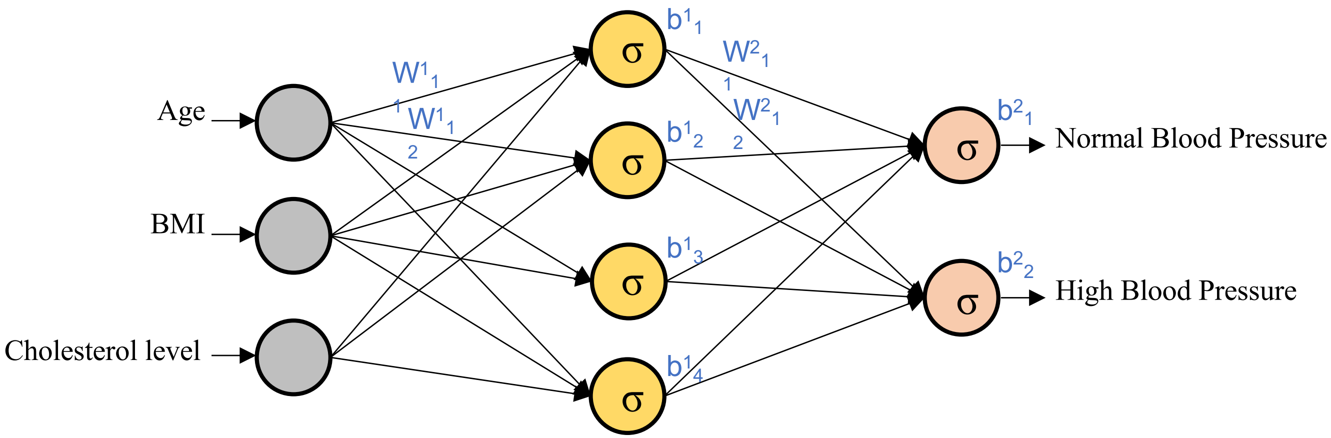
Neural Networks Basics

- Made up of multiple layers of nodes.
- Each layer make simple decisions using different weights, w, bias, b and activation function, σ.

Learnable parameters

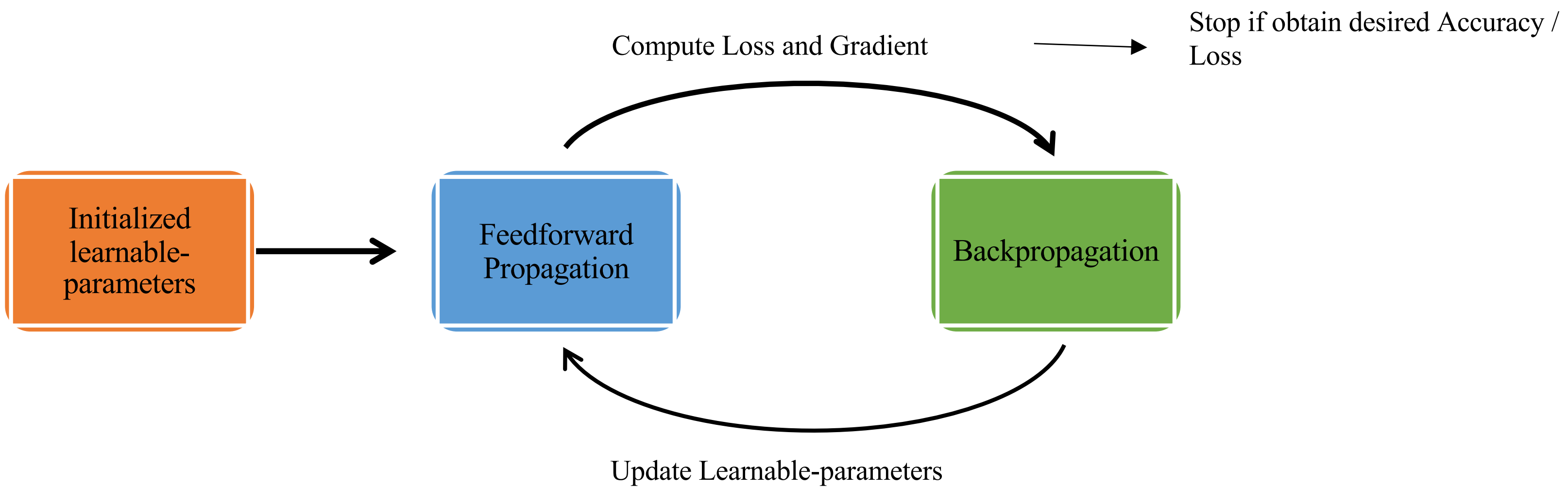


Neural Networks Basics

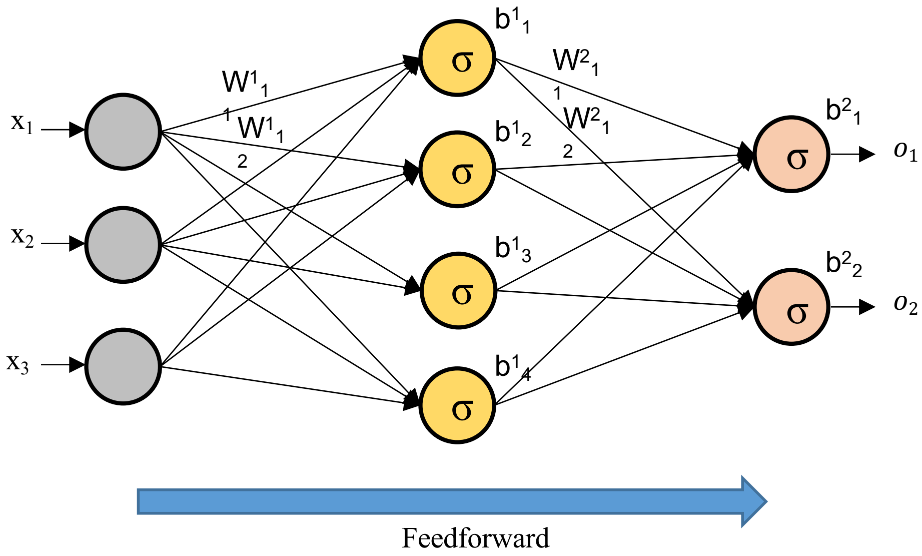


1. Given the diagram, which of the following are learnable parameters?
   A) Age
   B) σ
   C) W^1_{11}
   D) All of the above

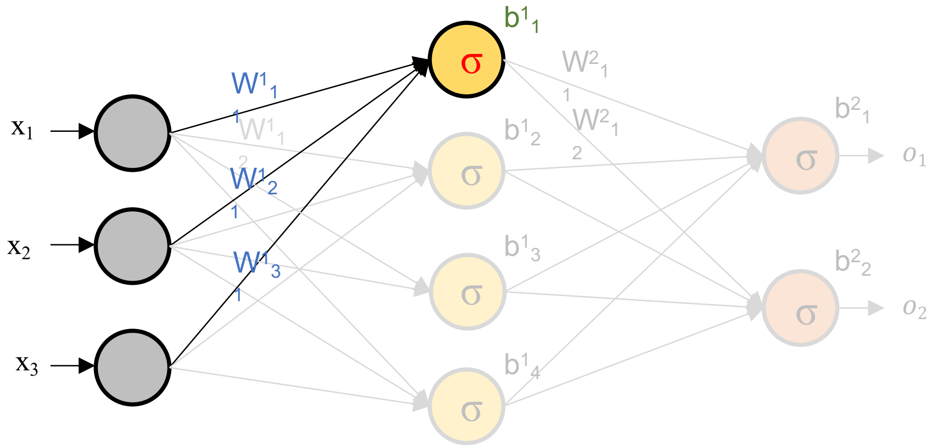
Neural Networks Training Process



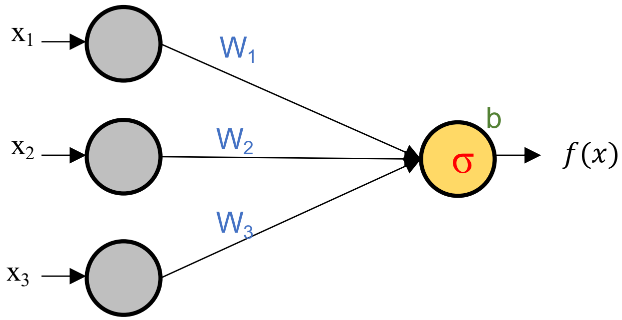
Feedforward Propagation



Feedforward Propagation



Feedforward Propagation (One Perceptron)



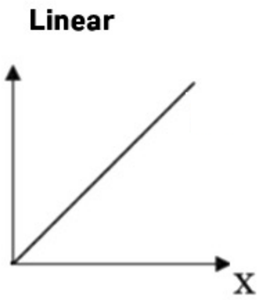
Input x Weight + Bias $(x_1 w_1 + x_2 w_2 + x_3 w_3 + b)$

Apply Activation Function $\sigma(x_1 w_1 + x_2 w_2 + x_3 w_3 + b)$

Obtain output $f(x) = \sigma(x_1 w_1 + x_2 w_2 + x_3 w_3 + b)$

Activation Function

- A mathematics function that determines the output of each perceptron in the neural network



$\sigma(x) = x, for\ all\ x$

Y=X

$$f(x) = \sigma(x_1 w_1 + x_2 w_2 + x_3 w_3 + b)$$

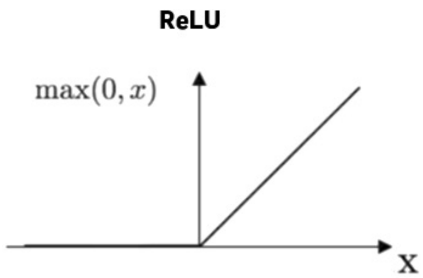
$$= \sigma(20 \cdot (-3) + 23 \cdot 2 + 4 \cdot 9 + 8)$$

$$= \sigma(-60 + 46 + 36 + 8)$$

$$= \sigma(30)$$

$$= 30$$

Activation Function



$$\boxed{\sigma(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}}$$

1. Given ReLU activation, what is the output of $\sigma(x)$ if x = 13? +ve

   A) 13 B) 0 C) -9 D) 21

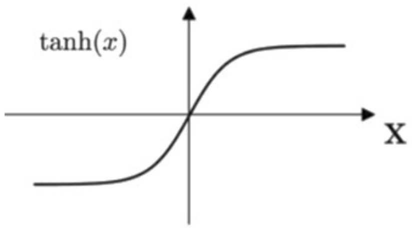
2. Given ReLU activation, what is the output of $\sigma(x)$ if x = −13?

   A) 13 B) 0 C) -9 D) 21

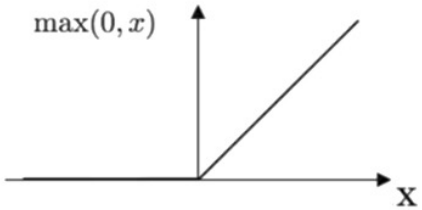
Activation Function

- Common activation functions:

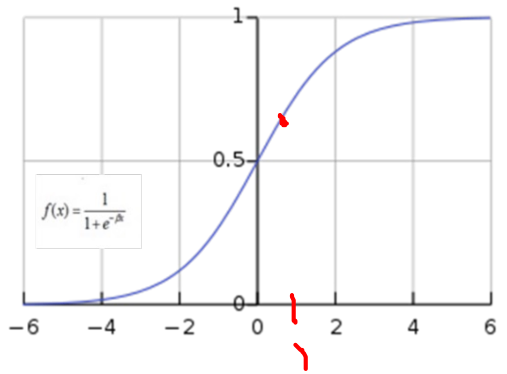
Tanh



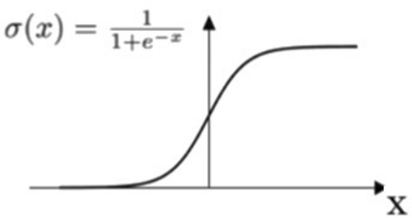
ReLU



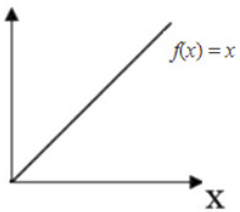
Softmax



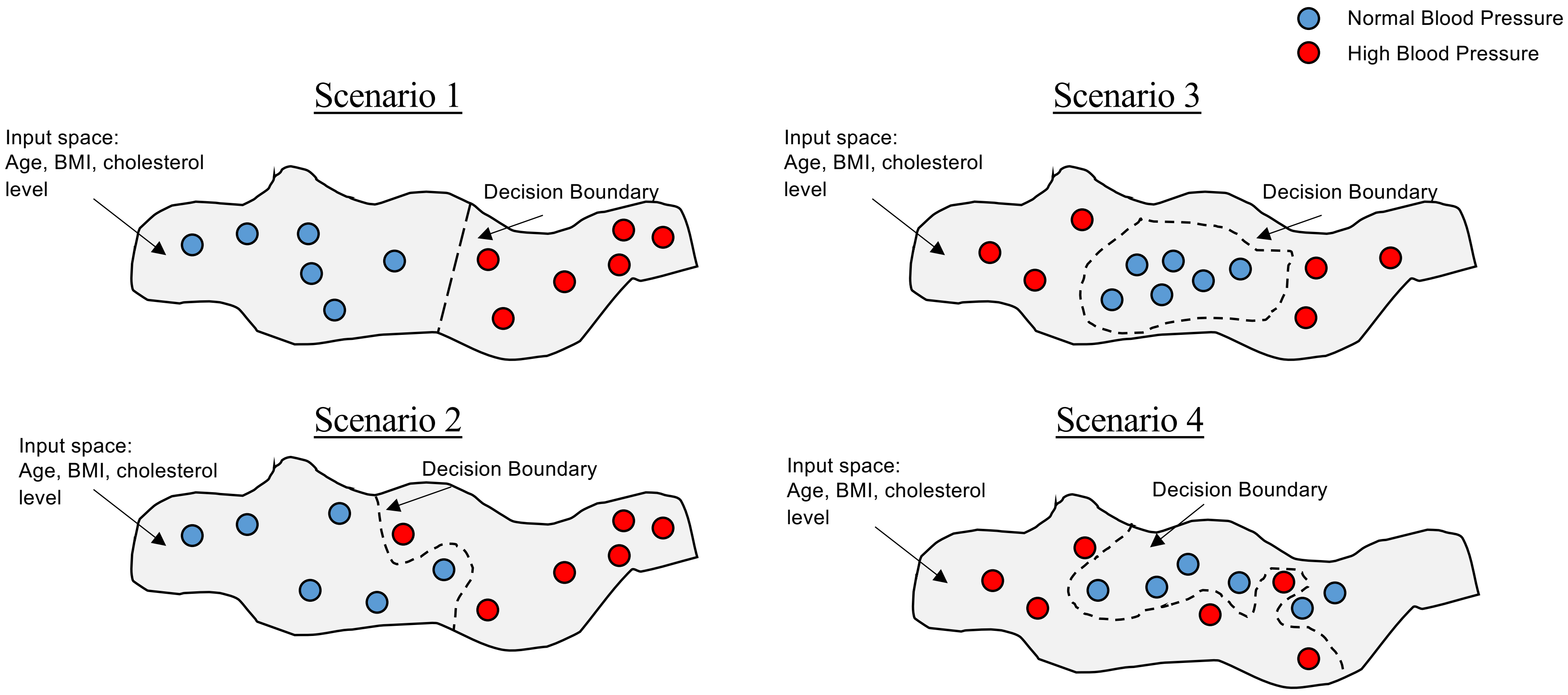
Sigmoid



Linear

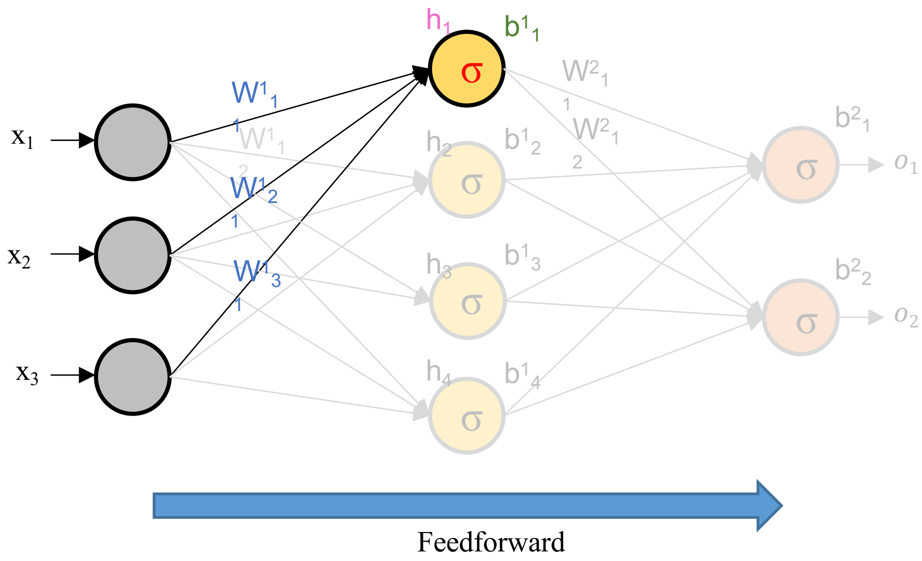


Why Different Activation Functions?



Feedforward All Perceptrons

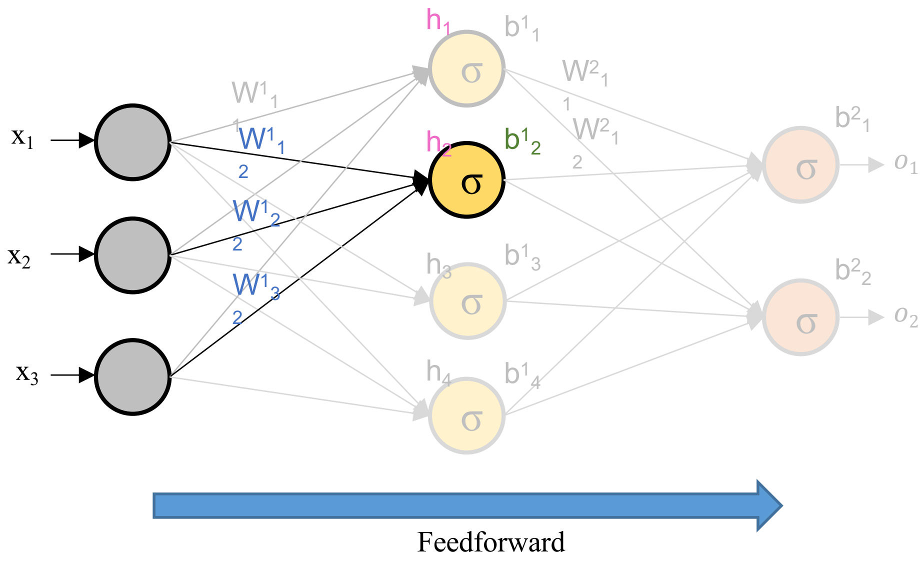
Feedforward Neural Network



$$h_1 = \sigma(\overbrace{x_1 w_{11}^1 + x_2 w_{21}^1 + x_3 w_{31}^1 + b_1^1}^{\text{Input x Weight + Bias}})$$

Activation function

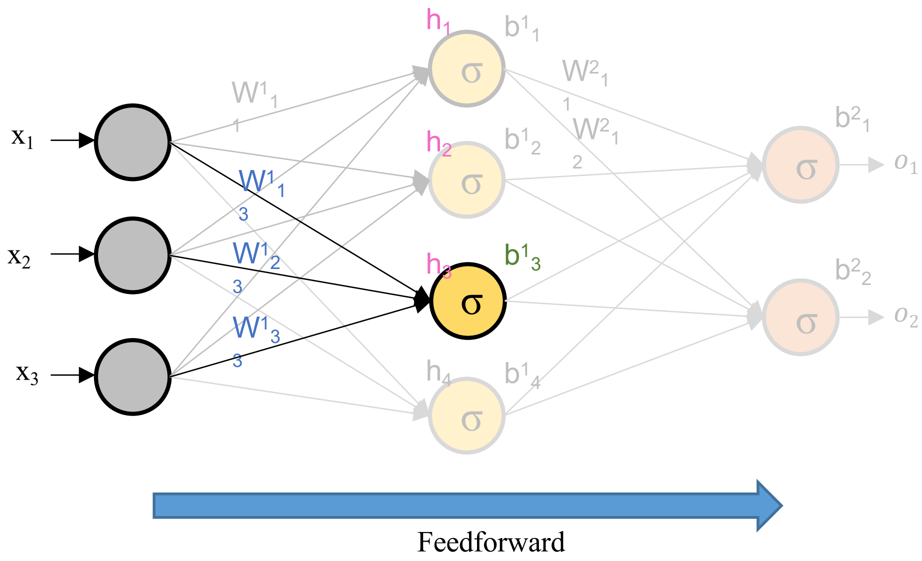
Feedforward All Perceptrons



$$h_1 = \sigma(x_1 w_{11}^1 + x_2 w_{21}^1 + x_3 w_{31}^1 + b_1^1)$$

$$h_2 = \sigma(x_1 w_{12}^1 + x_2 w_{22}^1 + x_3 w_{32}^1 + b_2^1)$$

Feedforward All Perceptrons

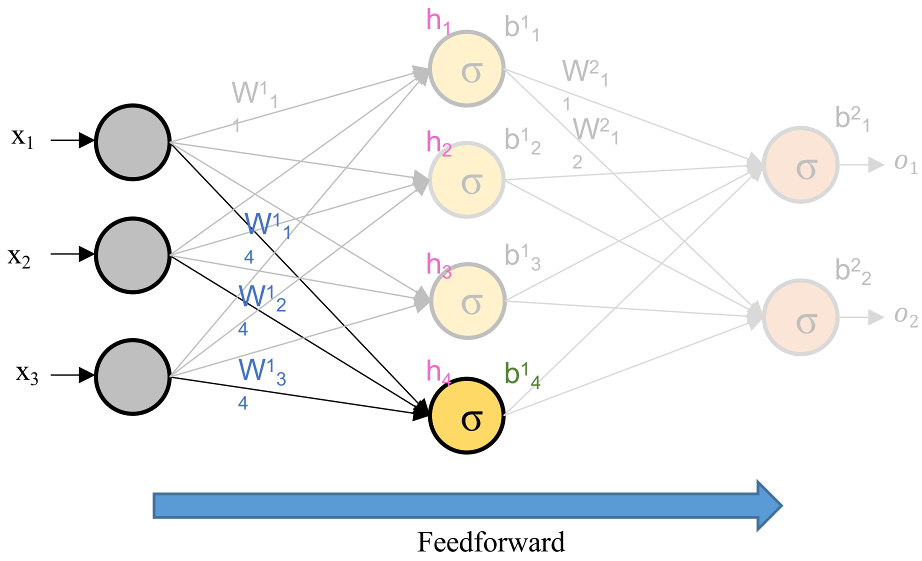


$$h_1 = \sigma(x_1 w_{11}^1 + x_2 w_{21}^1 + x_3 w_{31}^1 + b_1^1)$$

$$h_2 = \sigma(x_1 w_{12}^1 + x_2 w_{22}^1 + x_3 w_{32}^1 + b_2^1)$$

$$h_3 = \sigma(x_1 w_{13}^1 + x_2 w_{23}^1 + x_3 w_{33}^1 + b_3^1)$$

Feedforward All Perceptrons



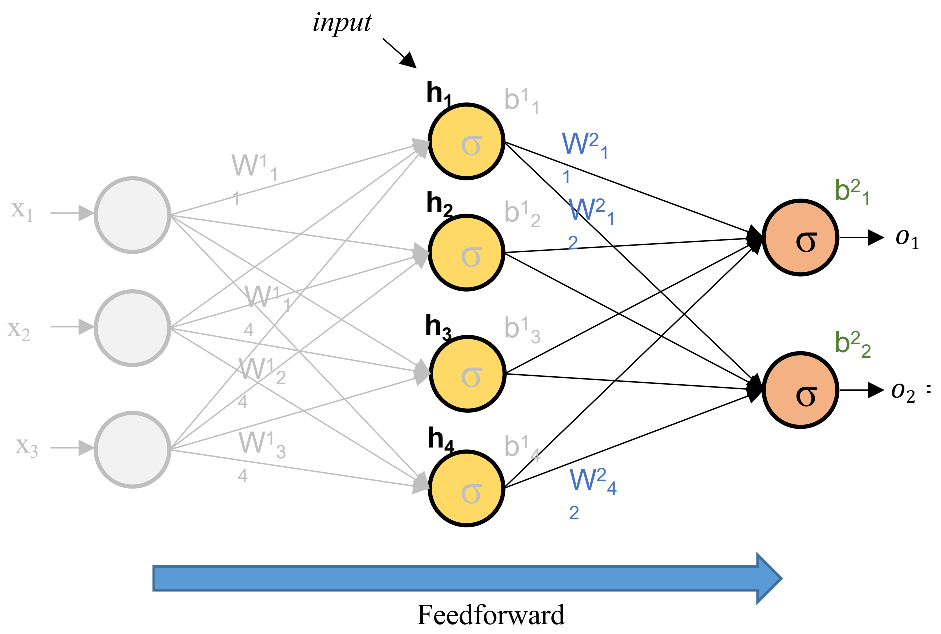
$$h_1 = \sigma(x_1 w_{11}^1 + x_2 w_{21}^1 + x_3 w_{31}^1 + b_1^1)$$

$$h_2 = \sigma(x_1 w_{12}^1 + x_2 w_{22}^1 + x_3 w_{32}^1 + b_2^1)$$

$$h_3 = \sigma(x_1 w_{13}^1 + x_2 w_{23}^1 + x_3 w_{33}^1 + b_3^1)$$

$$h_4 = \sigma(x_1 w_{14}^1 + x_2 w_{24}^1 + x_3 w_{34}^1 + b_4^1)$$

Feedforward All Perceptrons



$$o_1 = \sigma(h_1 w_{11}^2 + h_2 w_{21}^2 + h_3 w_{31}^2 + h_4 w_{41}^2 + b_1^2)$$

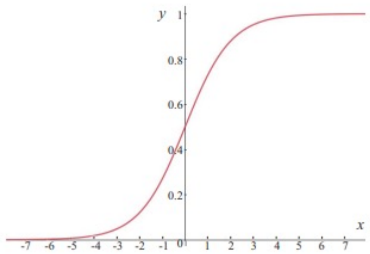
$$o_2 = \sigma(h_1 w_{12}^2 + h_2 w_{22}^2 + h_3 w_{32}^2 + h_4 w_{42}^2 + b_2^2)$$

Loss Functions

$o_1 = 0.4$

$o_2 = 0.2$

→ Sigmoid function → Output Probability → Prediction, $\hat{y}$



Prediction, $\hat{y}$

$> 0.5 \rightarrow 1$

$< 0.5 \rightarrow 0$

Ground Truth

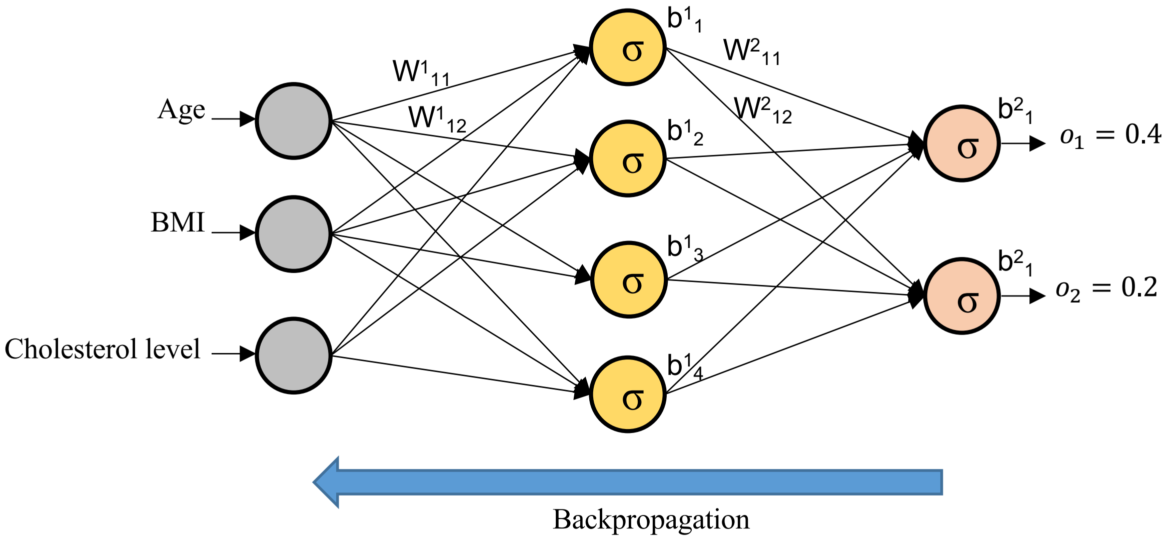
Normal blood pressure, y = 0

High blood pressure, y=1

$$Loss = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \text{ (Ground Truth - Prediction)}$$

Backpropagations to Update Weight

- Compute Gradients, i.e: $\frac{\partial L}{\partial w_{ij}^{n}}, \frac{\partial L}{\partial b_{i}^{n}}$
- Backpropagate the gradient to updates the weights



Gradient Descent

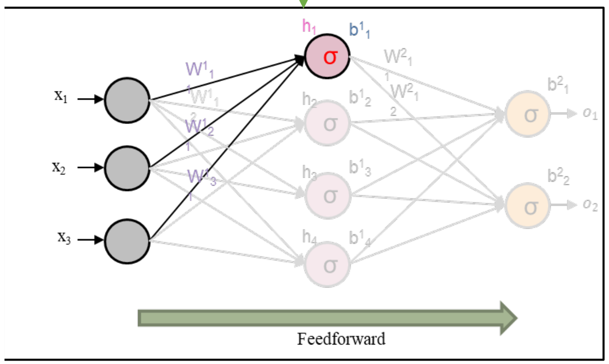
$$\theta_{t+1} = \theta_t + \rho_t d_t$$

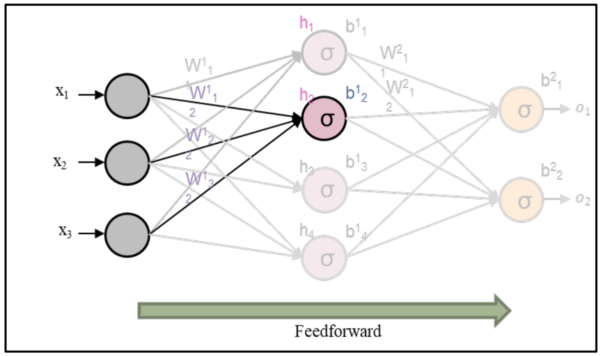
Updated parameter (weight, bias)

Step size

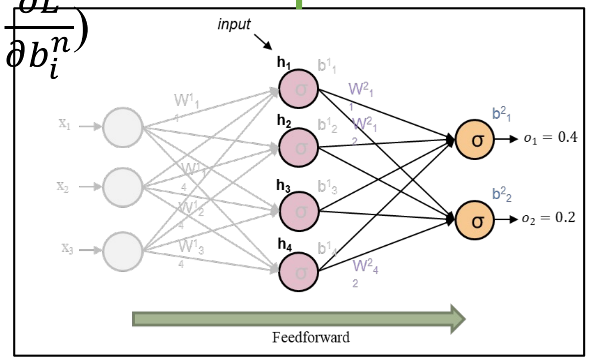
Descent direction (Gradient: $\frac{\partial L}{\partial w_{ij}^n}$, $\frac{\partial L}{\partial b_i^n}$)

Compute Gradients, i.e: $\frac{\partial L}{\partial w_{ij}^n}$, $\frac{\partial L}{\partial b_i^n}$

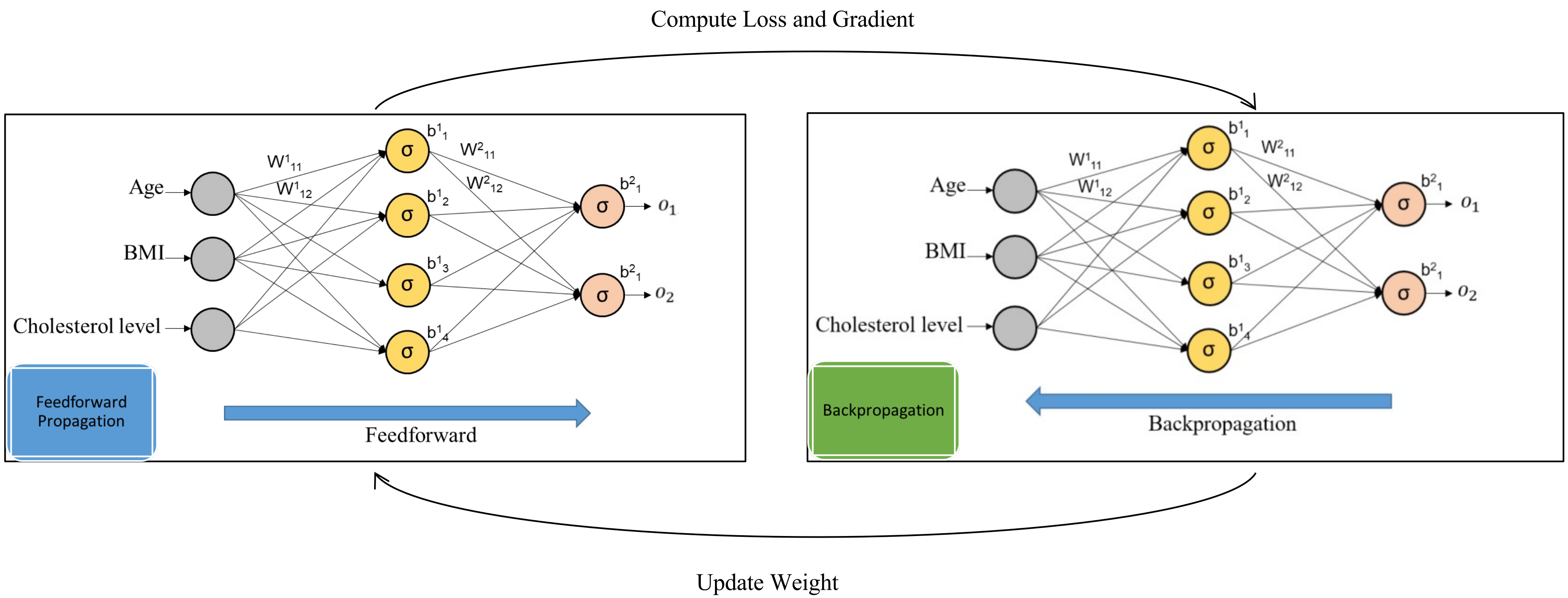




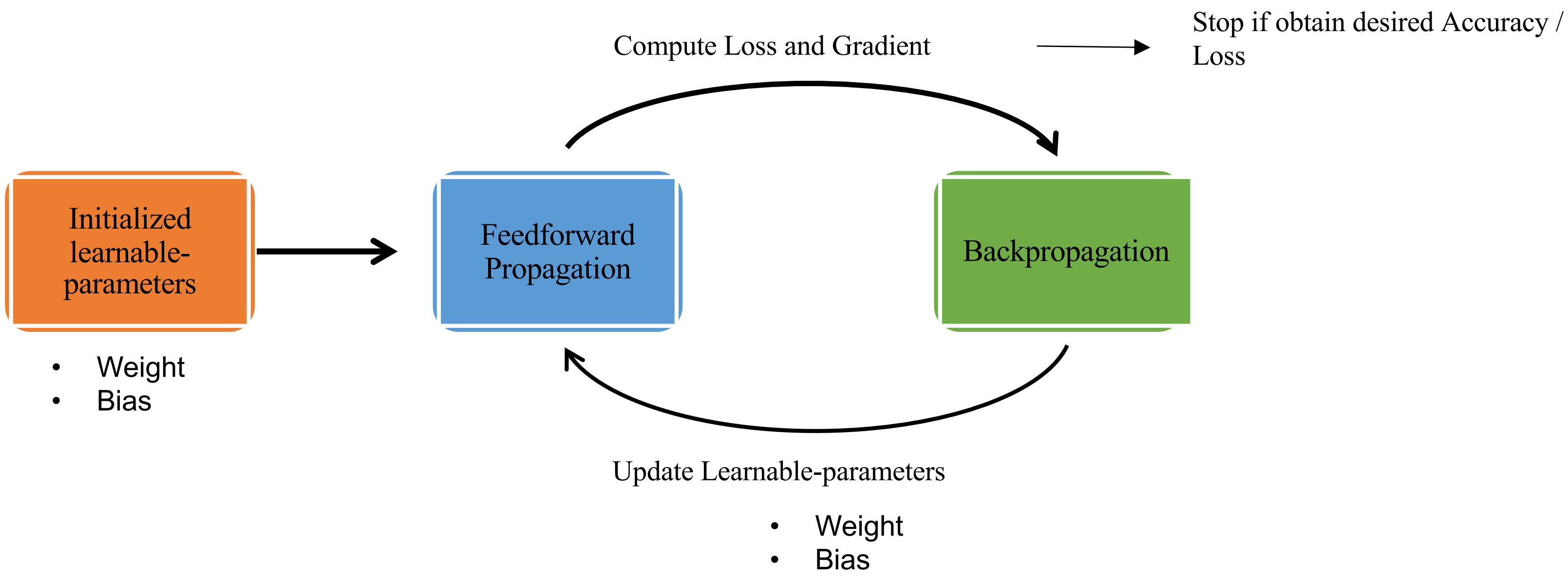
…



Training Iteratively Until Loss ~ 0

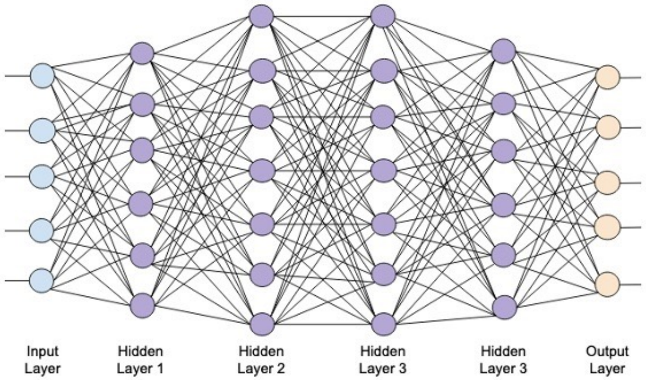


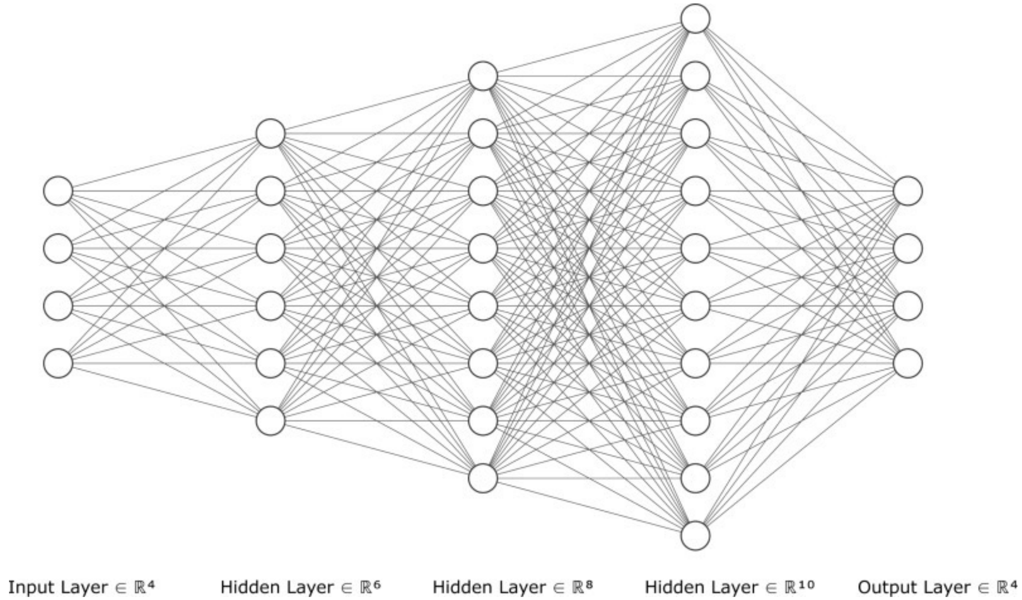
Summary: Neural Networks Training Process



Larger Network

- Increase number of node
- Increase number of hidden layer





Lager Network

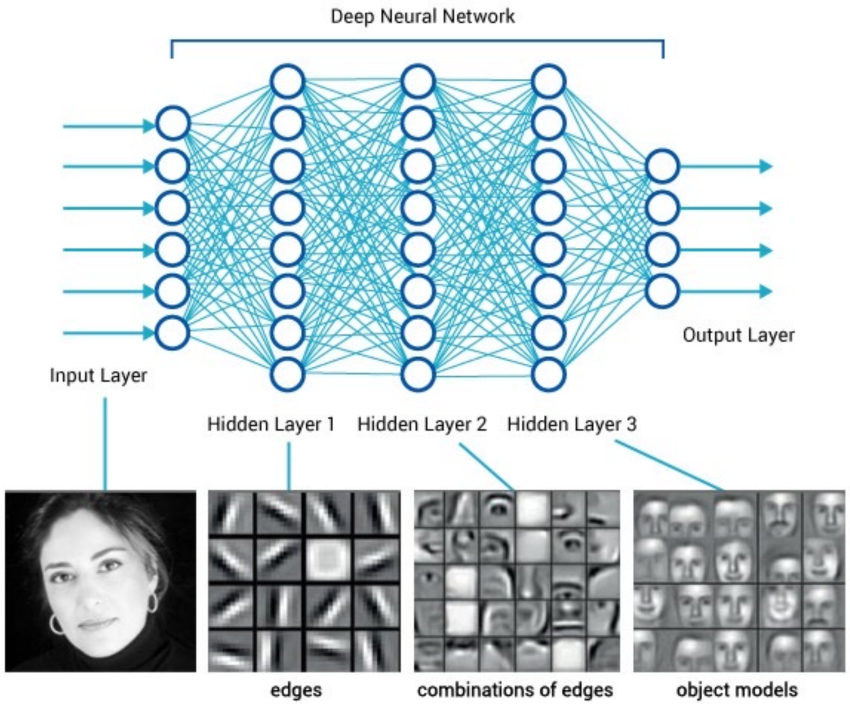
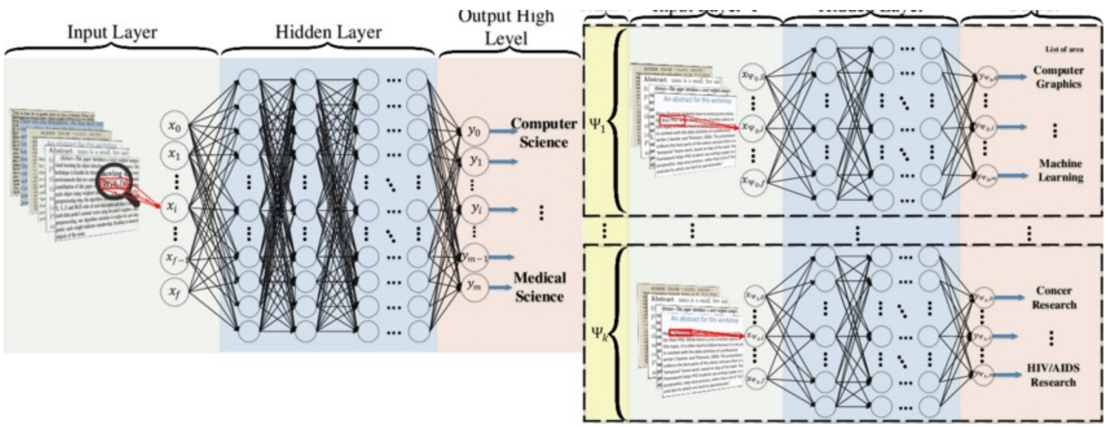


Figure from https://medium.com/diaryofawannapreneur/deep-learning-for-computer-vision-for-the-average-person-861661d8aa61



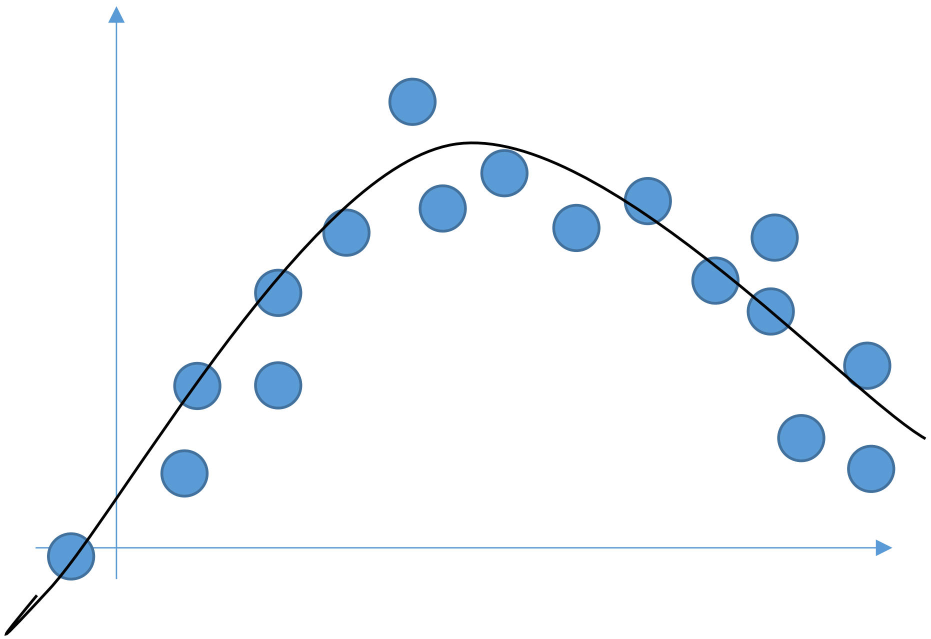
Kowsari et al. HDLTex: Hierarchical Deep Learning for Text Classification

Larger Network

- The larger the better???
- No → overfitting (layman term – memorize training data only, poor performance when testing data is used)
- What the reason of overfitting??? One of the reason is we have too many parameters
- Why do I say so?

Overfitting

If we have the following graph



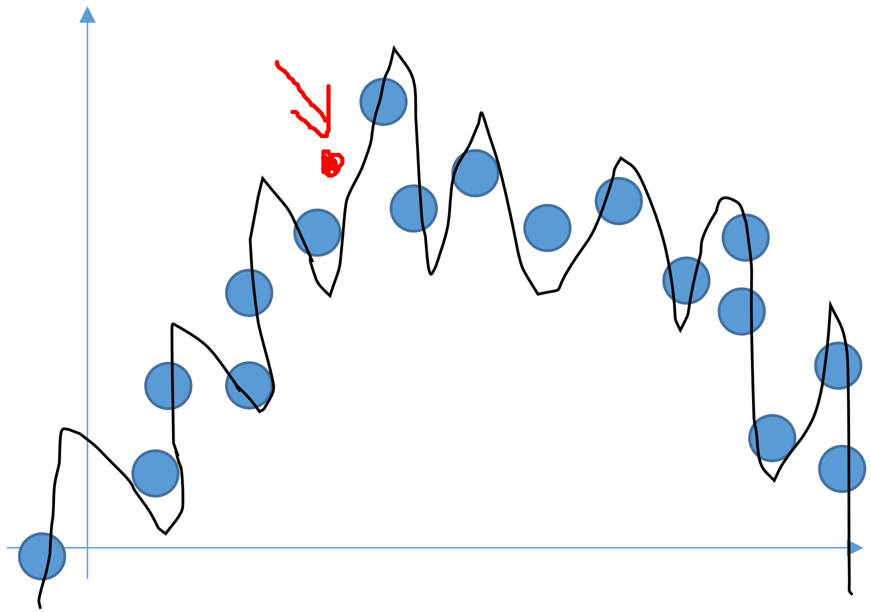
- We know it should be a quadratic graph
- Let's assume the best fit graph would be $y = -3x^2 + x + 0.5$

DEPARTMENT OF COMPUTER SCIENCE 計算機科學系

香港浸會大學 HONG KONG BAPTIST UNIVERSITY

Overfitting

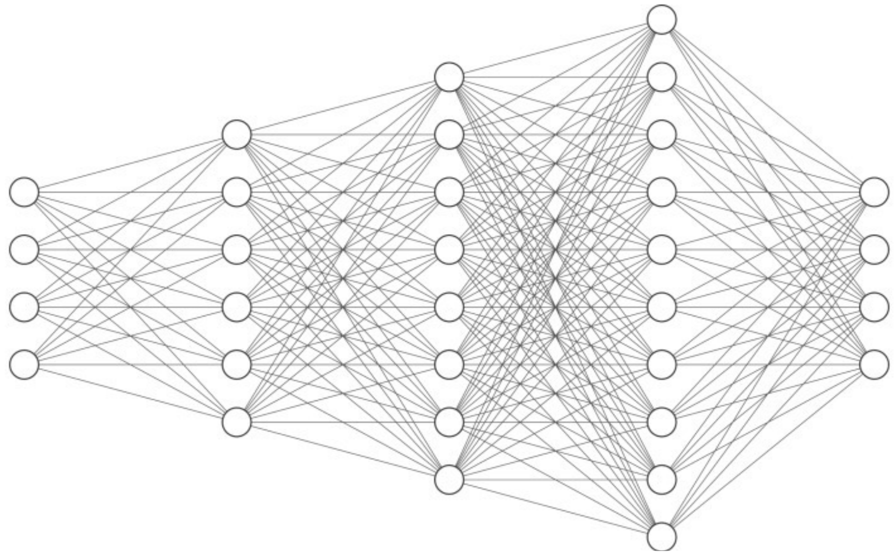
If we have the following graph



- We know it should be a quadratic graph
- Let's assume the best fit graph would be $y = -3x^2 + x + 0.5$
- If we fit with a higher order arbitrary polynomial function, $y = 0.4x^8 + 1.9x^7 - 1.4x^6 + \cdots + 1.9$ → overfit
- Have too many parameters
  - Quadratic – 3 parameters, m1, m2, b
  - 8th order – 9 parameters, m1, m2, …, m8, b

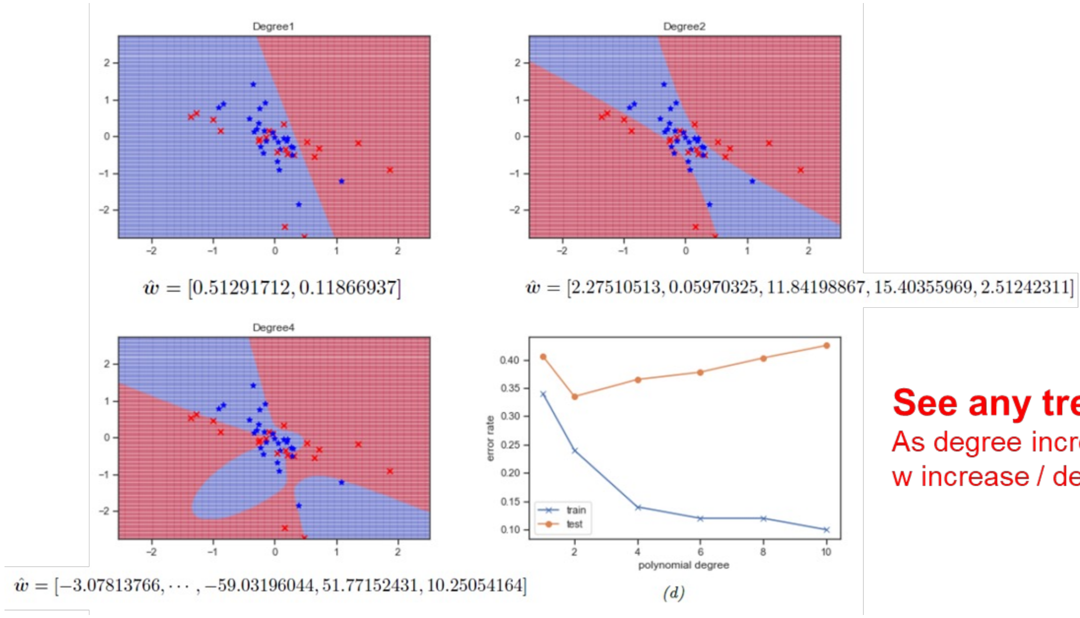
Overfitting

- Reason: Have too many parameters
- How to solve???
- Reduce the model size → number of node / hidden layer



Larger Network

- Another reason of overfitting is the weight value are too big.
- Remember previous slide (week4)? We observed that the value of weight increase in overfitting model.



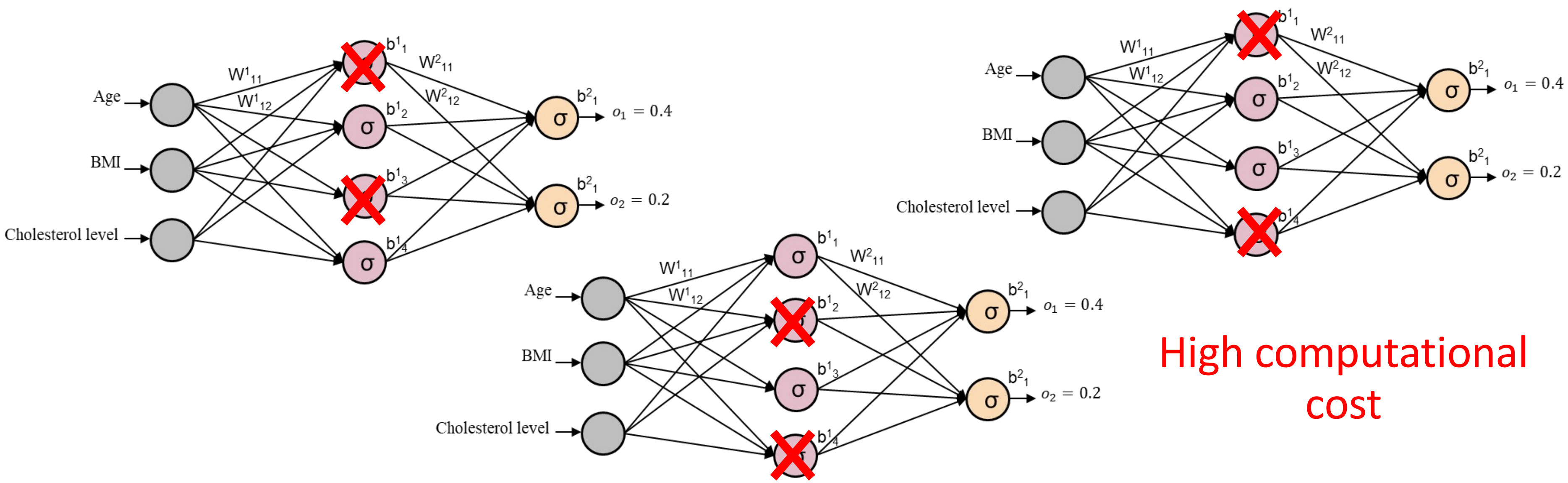
See any trend?
As degree increases,
w increase / decrease?

Larger Network

- Another reason of overfitting is the weight value are too big.
- Remember previous slide (week4)? We observed that the value of weight increase in overfitting model.
- To solve this, we regularize the weight value – don't let it grow too large via Batch Normalization
- Batch Normalization → normalize each layer to z-score (0 mean, 1 stdev)

Overfitting

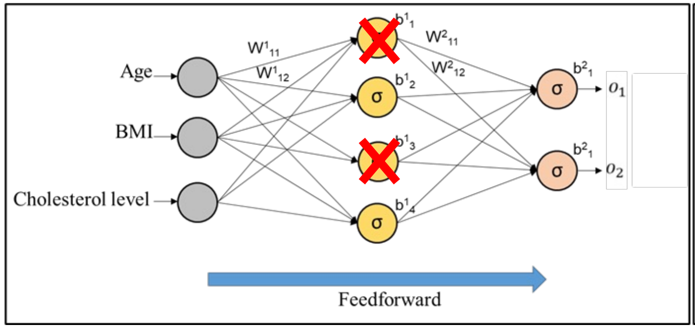
- Another way to solve: Use the idea of ensemble model
- Ensemble model → train multiple model then combine all of them.



Dropout

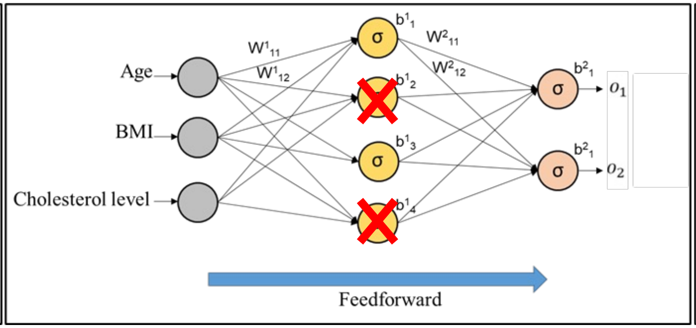
- Randomly drop the node

1st iteration (Model1)



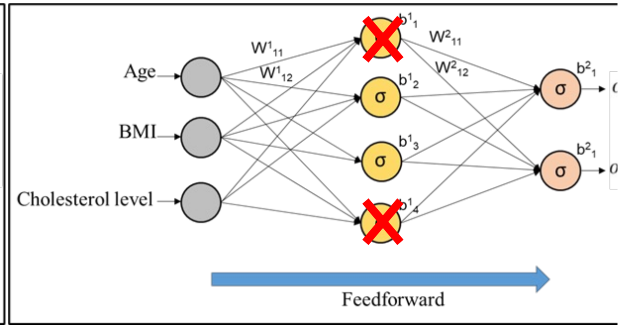
1. Forward
2. Compute gradient
3. Update weight

2nd iteration (Model2)



1. Forward
2. Compute gradient
3. Update weight

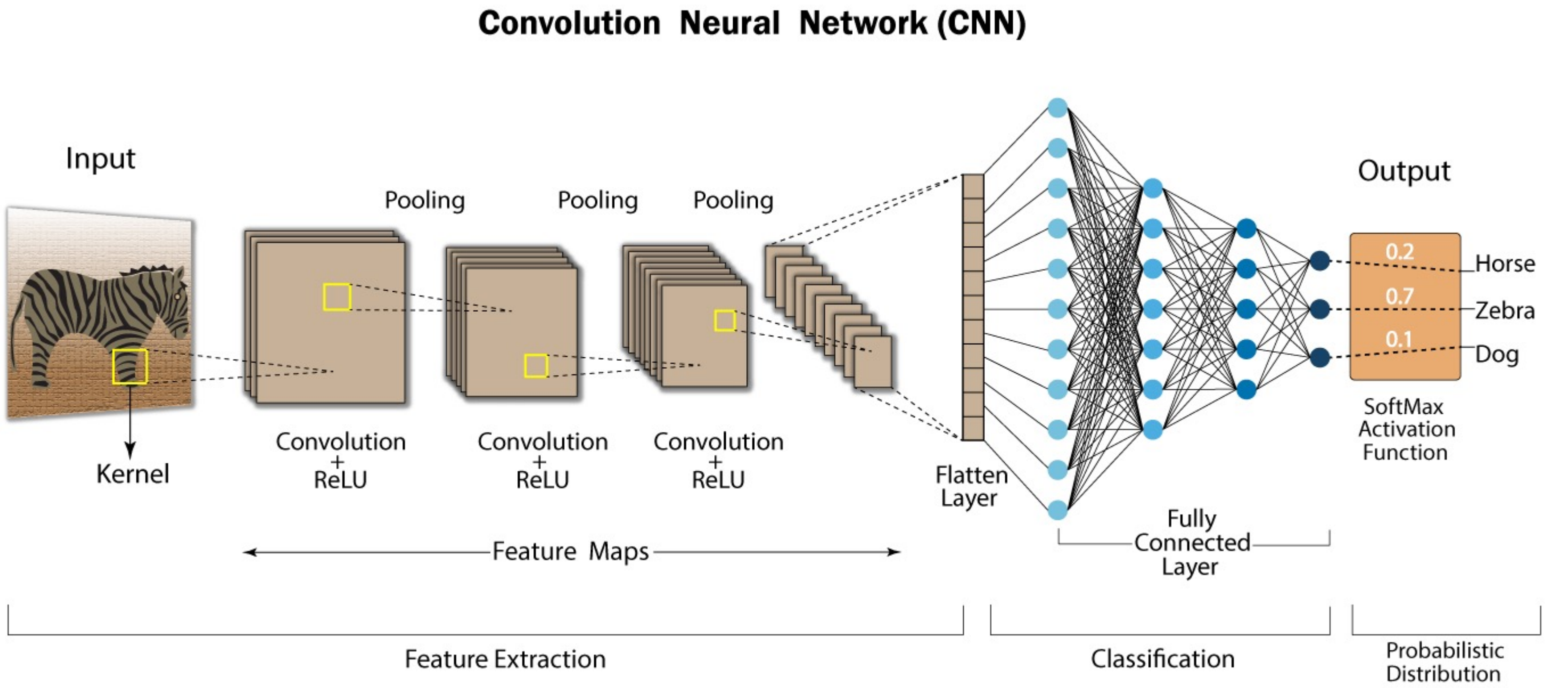
3rd iteration (Model3)



1. Forward
2. Compute gradient
3. Update weight

Idea: Train different model at every iteration.

Convolutional Neural Network (CNN)



From https://developersbreach.com/convolution-neural-network-deep-learning/

Convolutional Neural Network (CNN)

CNN

1	0	1	1	0	1
1	1	1	1	1	0
0	0	0	1	1	1
0	0	1	1	1	0
1	1	1	1	0	0
0	1	0	1	0	0

$$\begin{bmatrix}1 & 0 & 1\\ 1 & 1 & 1\\ 0 & 0 & 0\end{bmatrix} \cdot \begin{bmatrix}1 & 0 & 1\\ 1 & 0 & 1\\ 1 & 0 & 1\end{bmatrix} = 4$$

Convolution

3x3 kernel

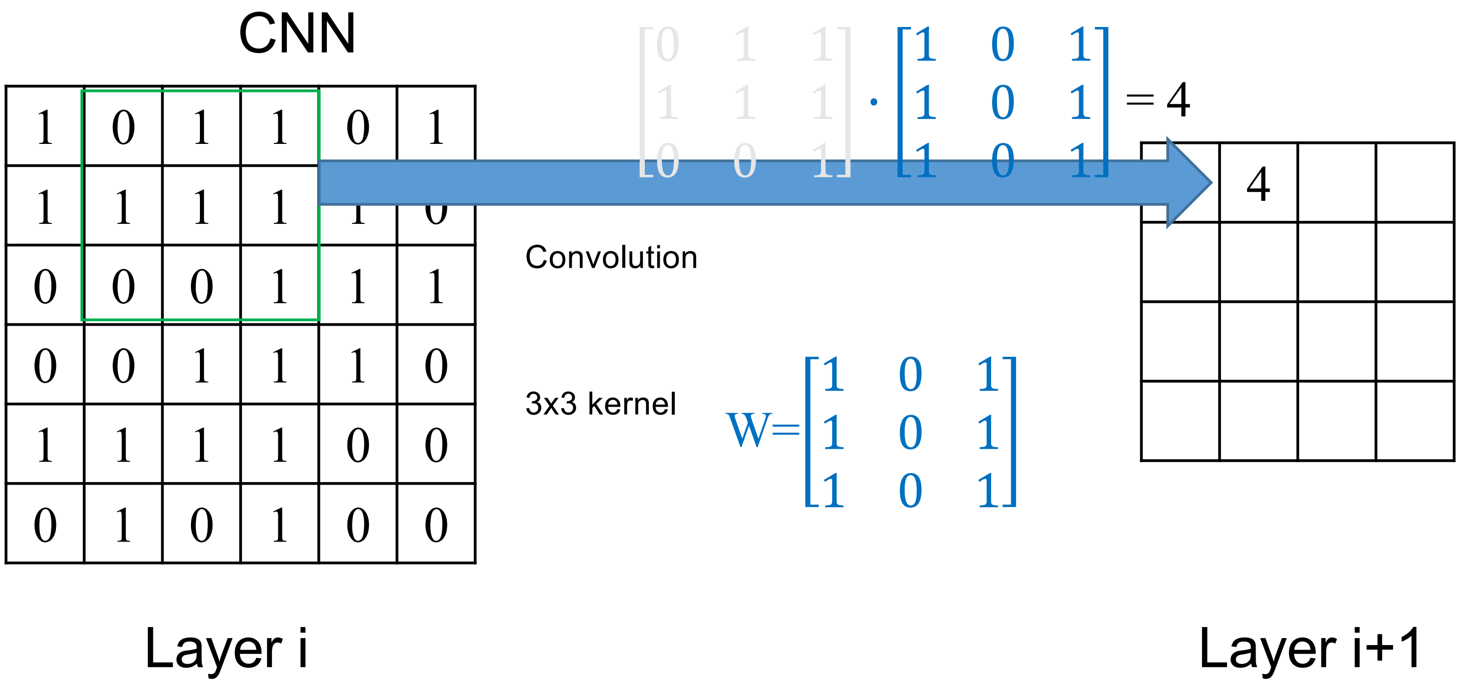
$$W=\begin{bmatrix}1 & 0 & 1\\ 1 & 0 & 1\\ 1 & 0 & 1\end{bmatrix}$$

4			

Layer i

Layer i+1

Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN)

CNN

1	0	1	1	0	1
1	1	1	1	1	0
0	0	0	1	1	1
0	0	1	1	1	0
1	1	1	1	0	0
0	1	0	1	0	0

Layer i

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} = 4$$

Convolution

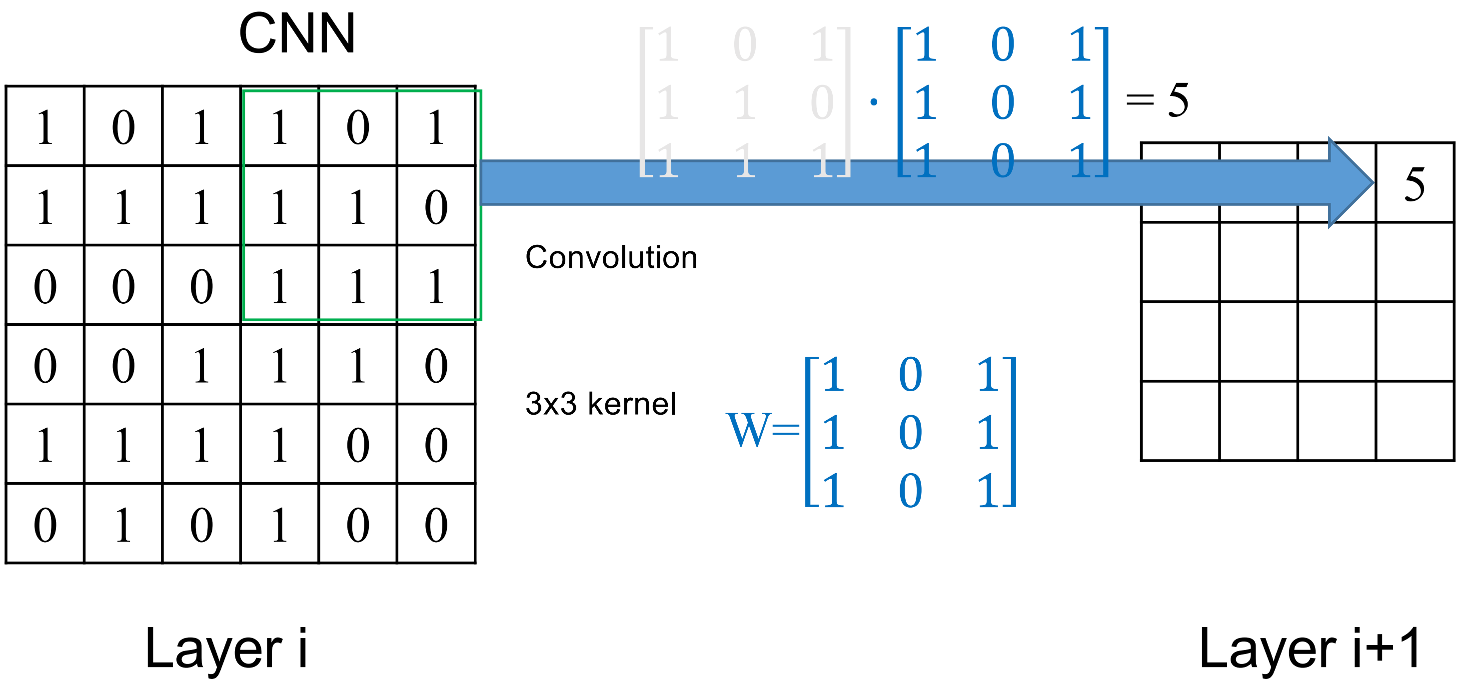
3x3 kernel

$$W=\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

		4	

Layer i+1

Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN)

CNN

1	0	1	1	0	1
1	1	1	1	1	0
0	0	0	1	1	1
0	0	1	1	1	0
1	1	1	1	0	0
0	1	0	1	0	0

Layer i

Repeat the rest

Convolution

3x3 kernel

$$W=\begin{bmatrix}1 & 0 & 1\\ 1 & 0 & 1\\ 1 & 0 & 1\end{bmatrix}$$

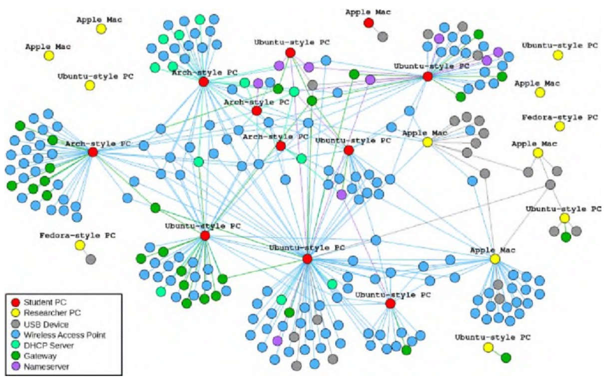
Feature extraction

4	4	4	5
4	4	5	4
3	3	4	4
3	5	3	3

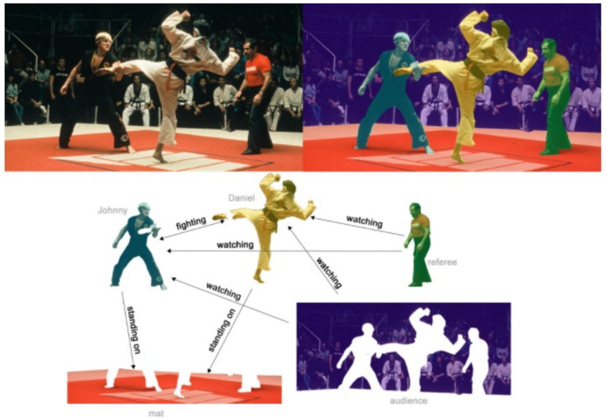
Layer i+1

Representation of Real-World Data as A Graph

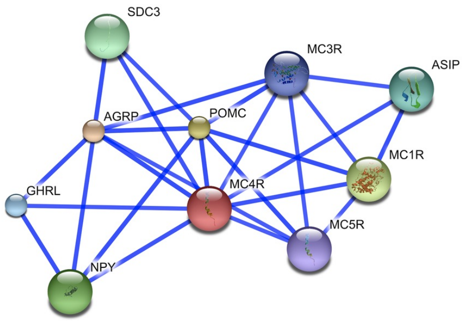
Social Network Graph



Scene Graph

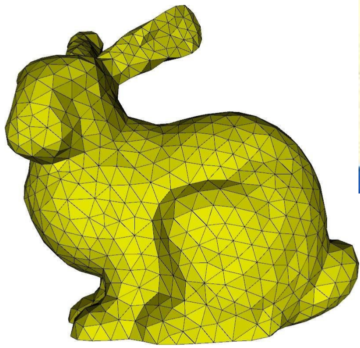


Transportation Network Graph



Molecule Network Graph

Geometric Network Graph



Graph Neural Network

- Graph is made up of nodes (vertices) and edges
- Graph is defined as: $G\ (V, E)$
- Edge can be undirected or directed edge.
- Direction of the edges indicate the dependencies: unidirected or bidirected or.
- A graph is often represented using adjacent matrix, A
- If a graph has n nodes, A has a dimension of $n \times n$
- Flexible but also mean it is harder to build

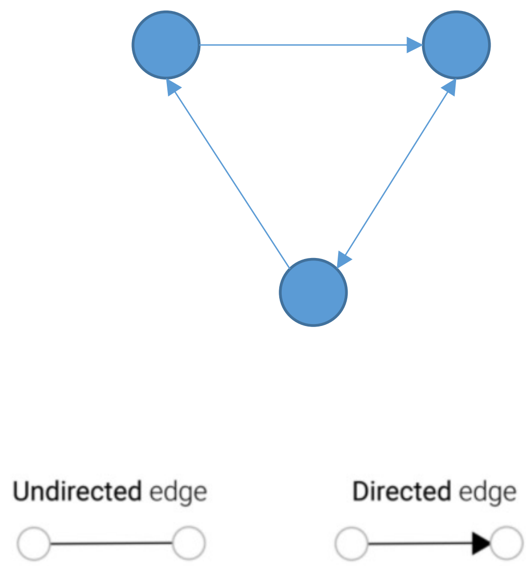
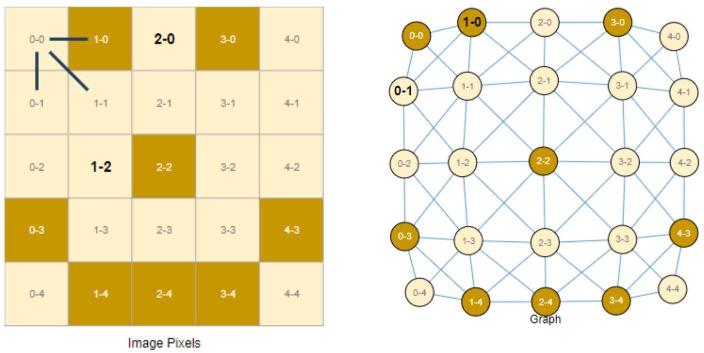


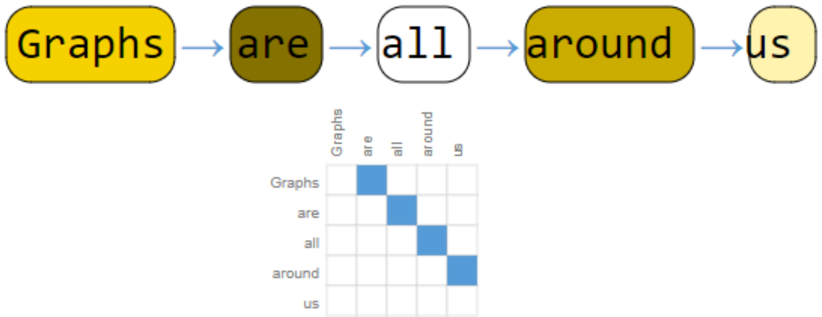
Figure form https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications

Graph Neural Network

Image as Graph



Text as Graph



Molecules as Graph

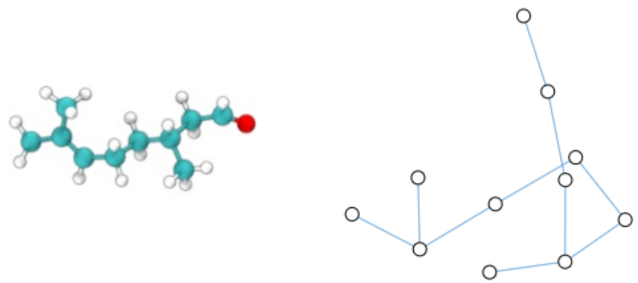
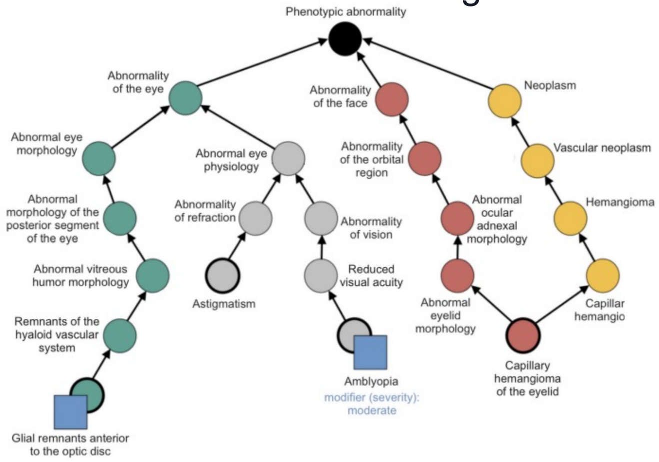


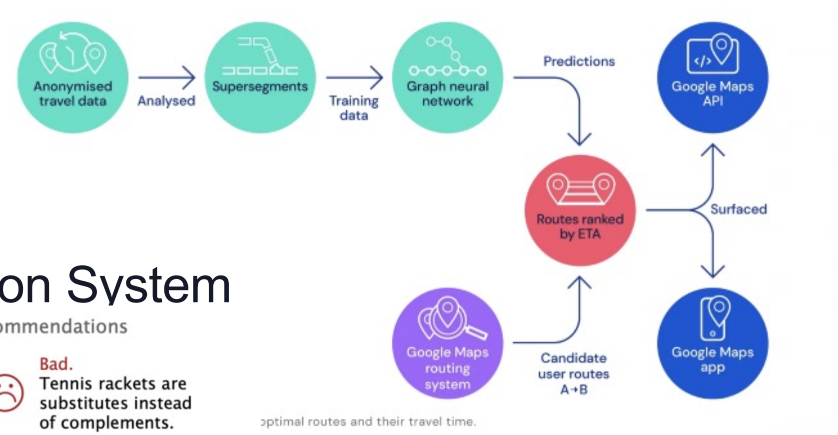
Figure from https://distill.pub/2021/gnn-intro/

Applications of Graph Neural Network

Medical Diagnosis & Electronic Health Records Modeling



Traffic forecasting



Recommendation System

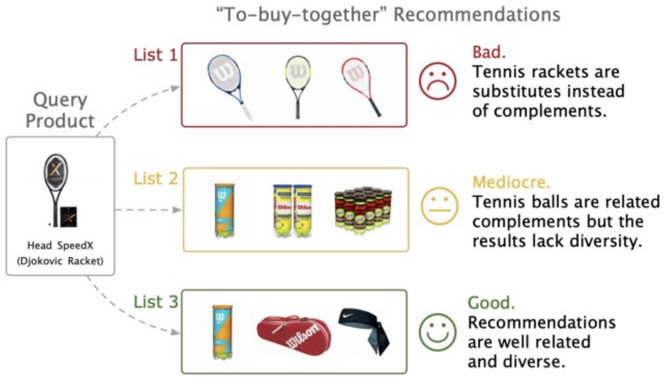


Figure form https://jonathan-hui.medium.com/applications-of-graph-neural-networks-gnn-d487fd5ed17d

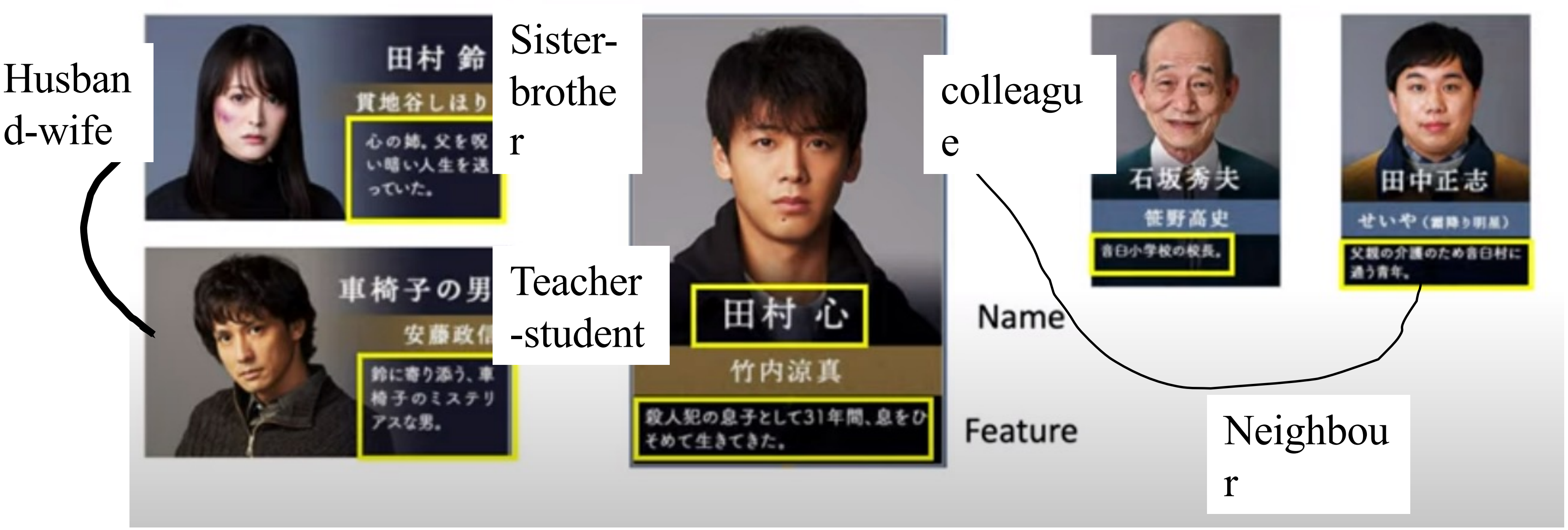
Basic Ideas of Classifiers

Features → Classifier → Who is the murder?

From https://www.youtube.com/watch?v=eybCCtNKwzA&ab_channel=Hung-yiLee

Graph Neural Networks (GNN)

Features → Classifier → Who is the murder?



From https://www.youtube.com/watch?v=eybCCtNKwzA&ab_channel=Hung-yiLee

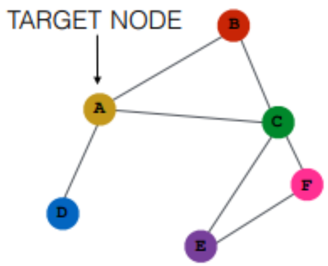
Graph Neural Network

- Approach1: Spatial-based convolution (same idea as CNN)
- Approach2: Spectral-based convolution (same idea as convolution in signal processing)

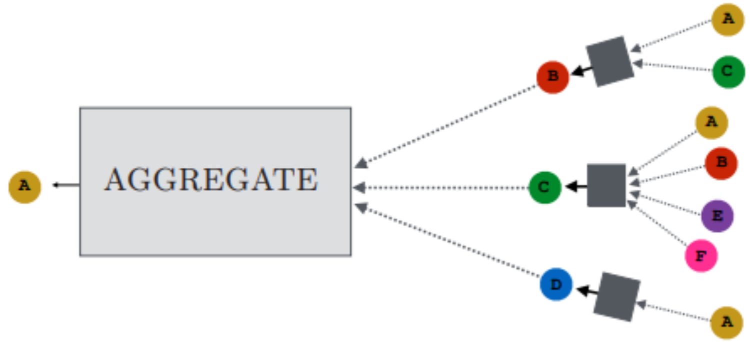
Spatial-based Convolution

Aggregate local network neighbourhoods:

- Sum
- Mean
- Weighted sum
- LSTM
- Max pooling



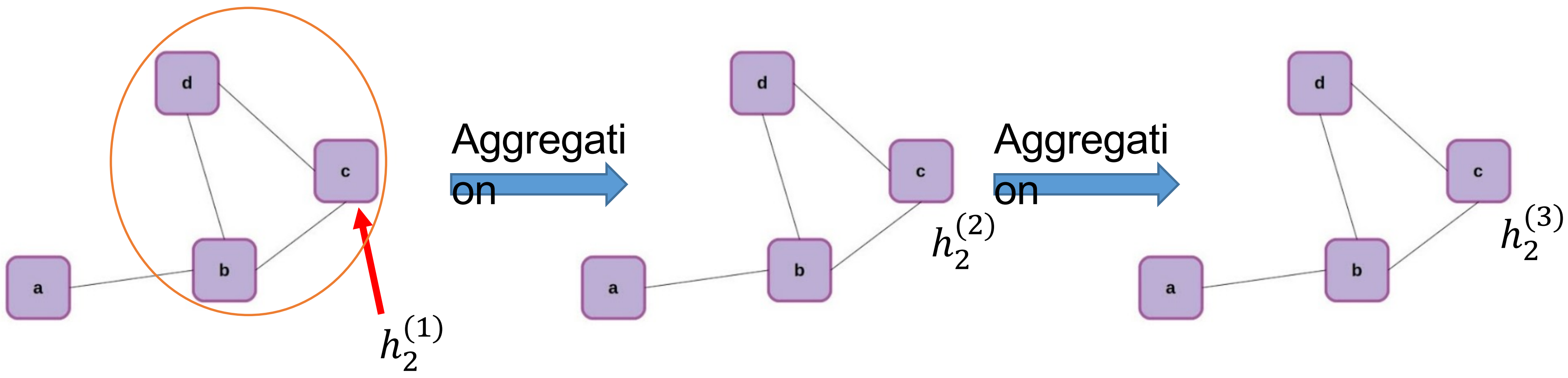
INPUT GRAPH



Message Passing

Figure from https://www.cs.mcgill.ca/~wlh/grl_book/files/GRL_Book-Chapter_5-GNNs.pdf

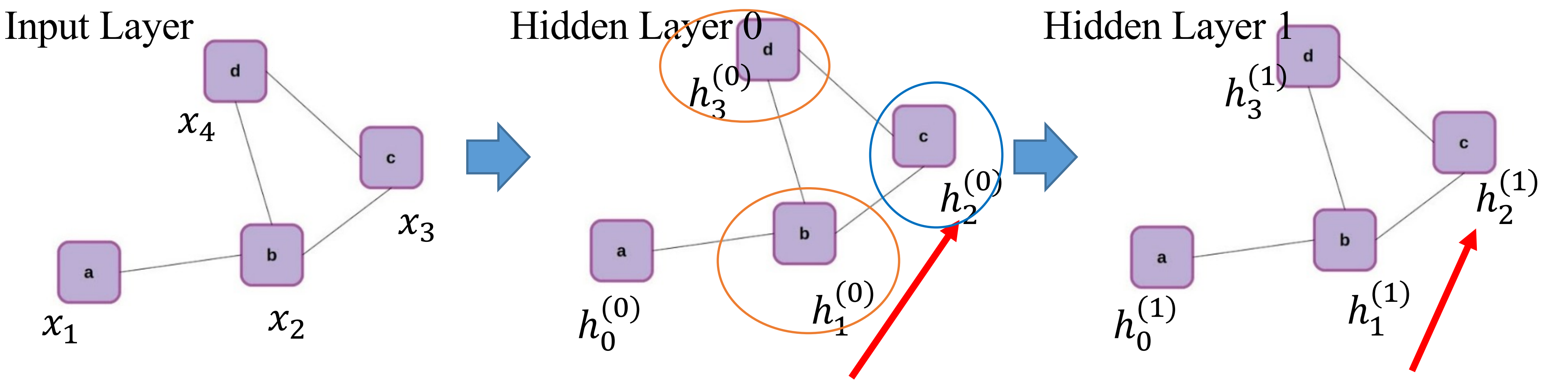
Spatial-based Convolution



Every node consists a hidden embedding, $h_u^{(k)}$

Aggregate: perform some operation with neighbour features and update the next state

Aggregation: Sum



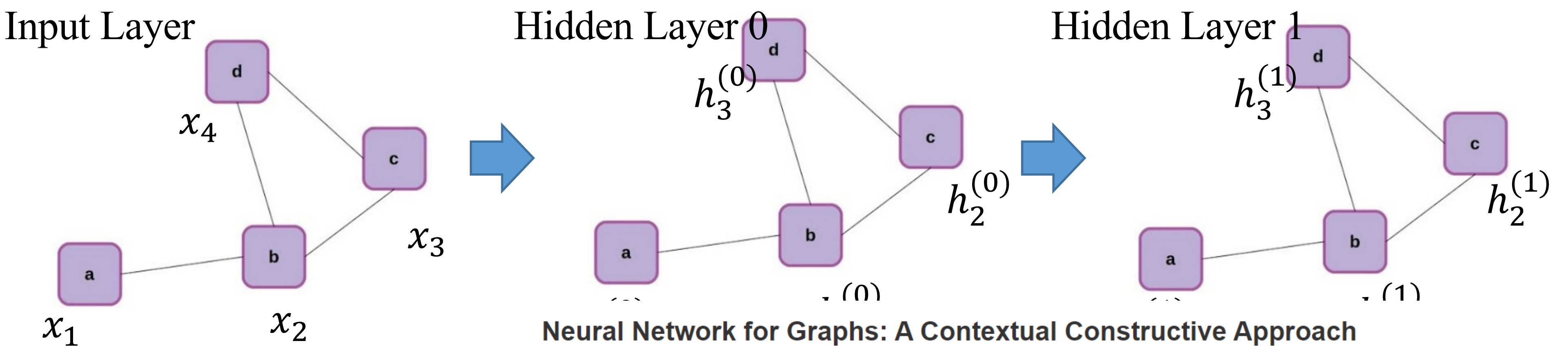
$$h_2^{(0)} = \bar{w}_0(x_2)$$

$$h_2^{(1)} = \hat{w}_{1,0}(h_1^0 + h_3^0) + \bar{w}_0(x_2)$$

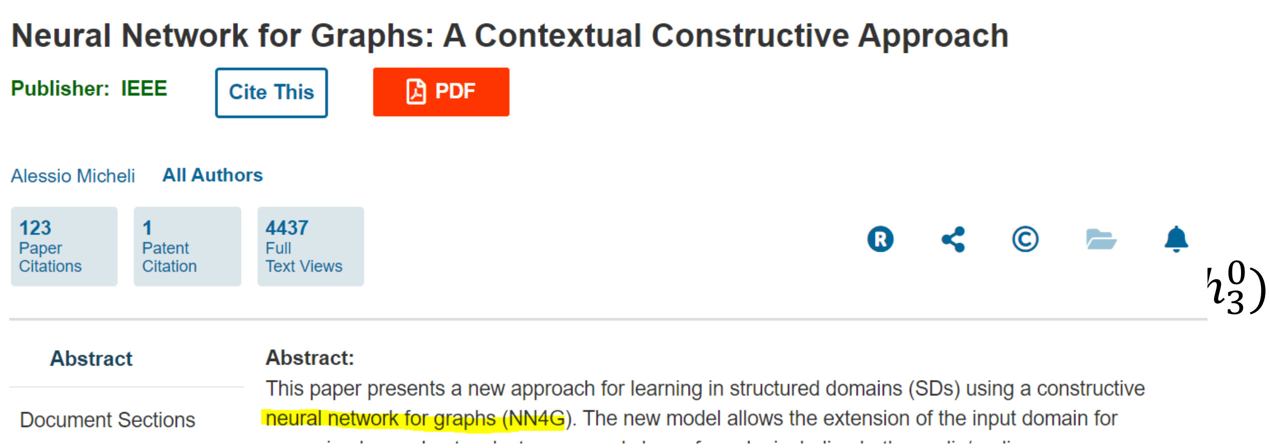
$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}(\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$
$$= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right),$$

m = message passing

Aggregation: Sum



NN4G (Neural Network for Graph)
https://ieeexplore.ieee.org/document/4773279


Neural Network for Graphs: A Contextual Constructive Approach

Publisher: IEEE | Cite This | PDF

Alessio Micheli | All Authors

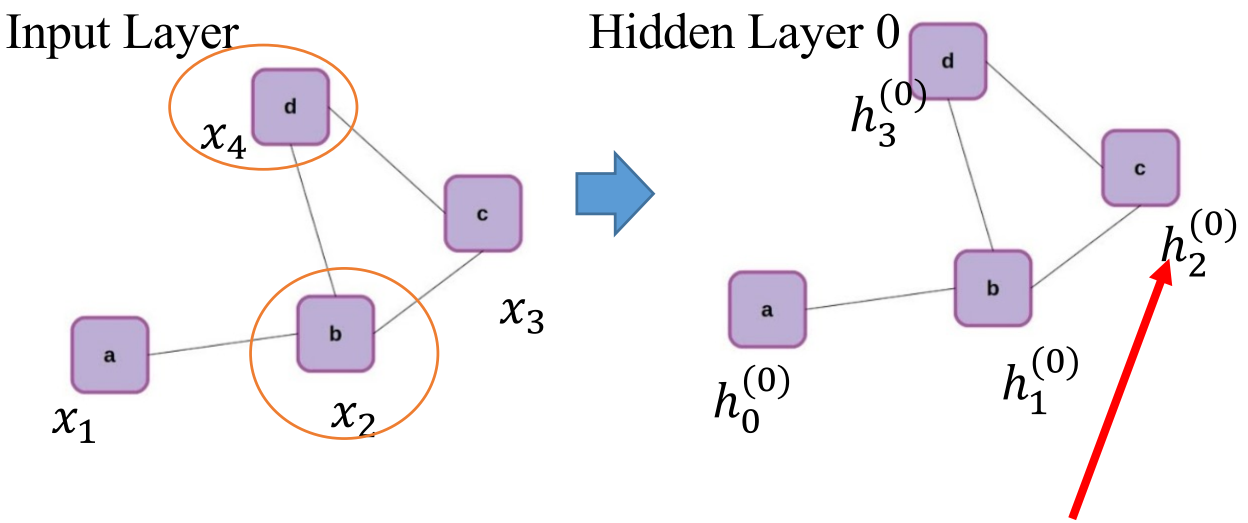
123 Paper Citations | 1 Patent Citation | 4437 Full Text Views

Abstract

Document Sections

Abstract:
This paper presents a new approach for learning in structured domains (SDs) using a constructive neural network for graphs (NN4G). The new model allows the extension of the input domain for

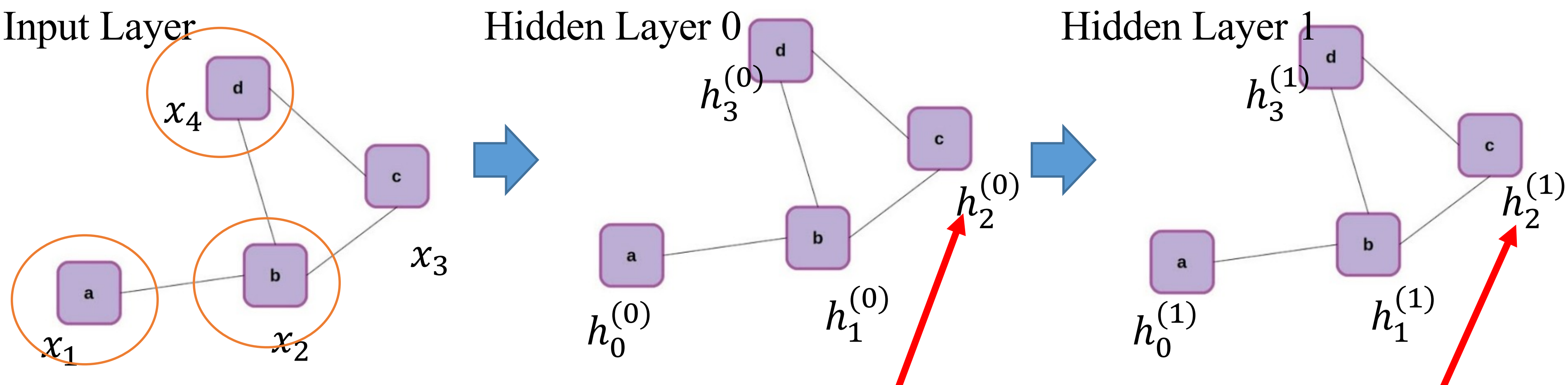
Aggregation: Mean



$$h_1^{(0)} = w_3^0 MEAN(d(3,:) = 1)$$

d = distance

Aggregation: Mean



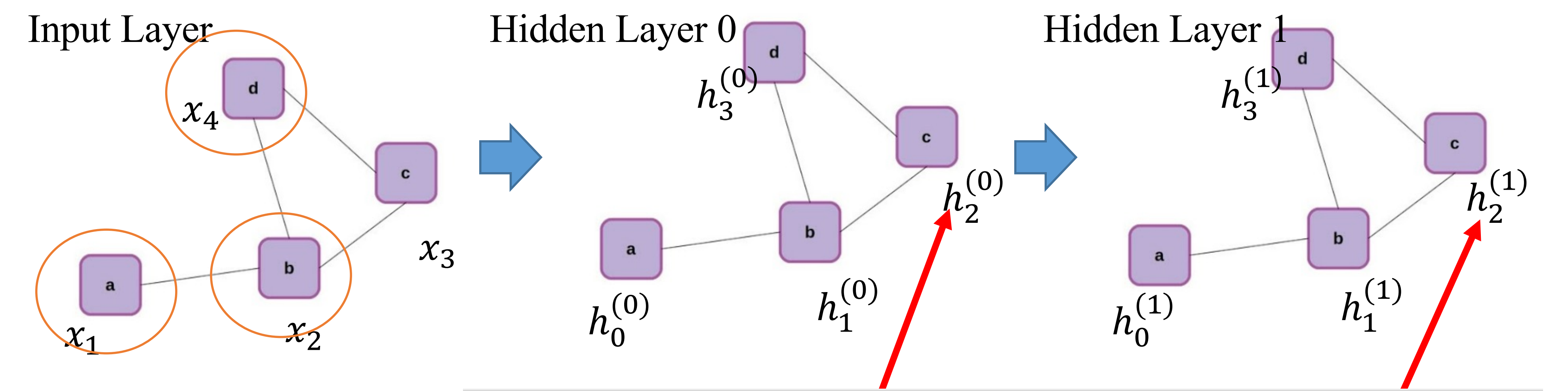
$h_1^{(1)} = w_3^1 MEAN(d(3,:) = 2)$

$h_1^{(0)} = w_3^0 MEAN(d(3,:) = 1)$

By having many hidden layer, we can extract the relationship between nodes from far.

d = distance

Aggregation: Mean



DCNN (Diffusion-Convolutional Neural Network)

https://arxiv.org/abs/1511.02136

[Submitted on 6 Nov 2015 (v1), last revised 8 Jul 2016 (this version, v6)]

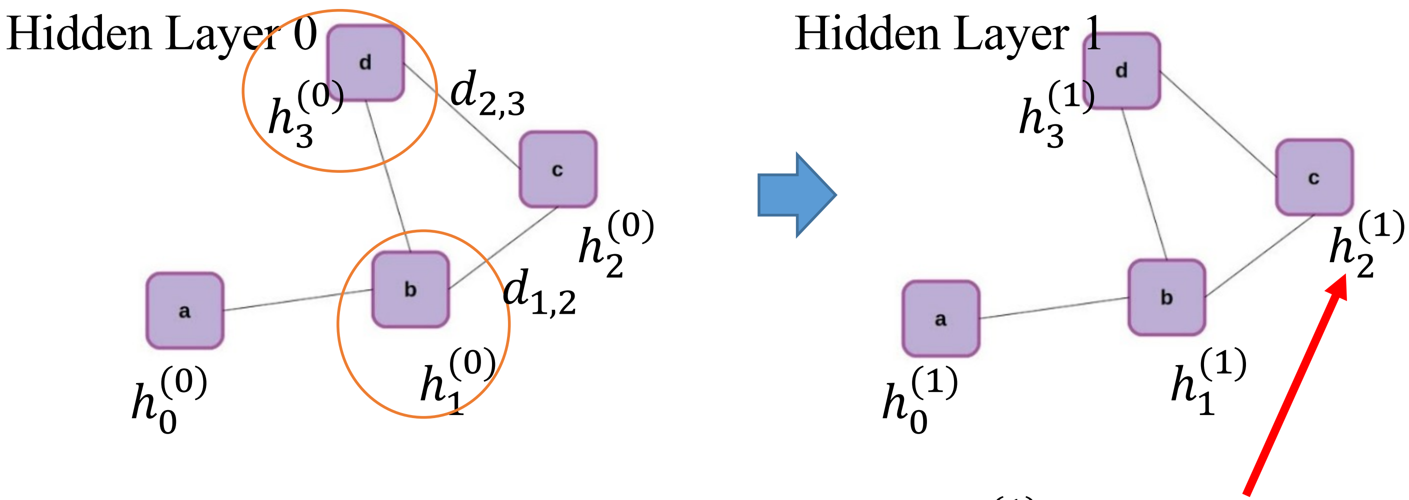
Diffusion-Convolutional Neural Networks

James Atwood, Don Towsley

We present diffusion-convolutional neural networks (DCNNs), a new model for graph-structured data. Through the introduction of a diffusion-convolution operation, we show how diffusion-based representations can be learned from graph-structured data and used as an effective basis for node classification. DCNNs have several attractive qualities, including a latent representation for graphical data that is invariant under isomorphism, as well as polynomial-time prediction and learning that can be represented as tensor operations and efficiently implemented on the GPU. Through several experiments with real structured datasets, we demonstrate that DCNNs are able to outperform probabilistic relational models and kernel-on-graph methods at relational node classification tasks.

Aggregation: Weighted Sum

Take into account the interaction of the neighbour's nodes

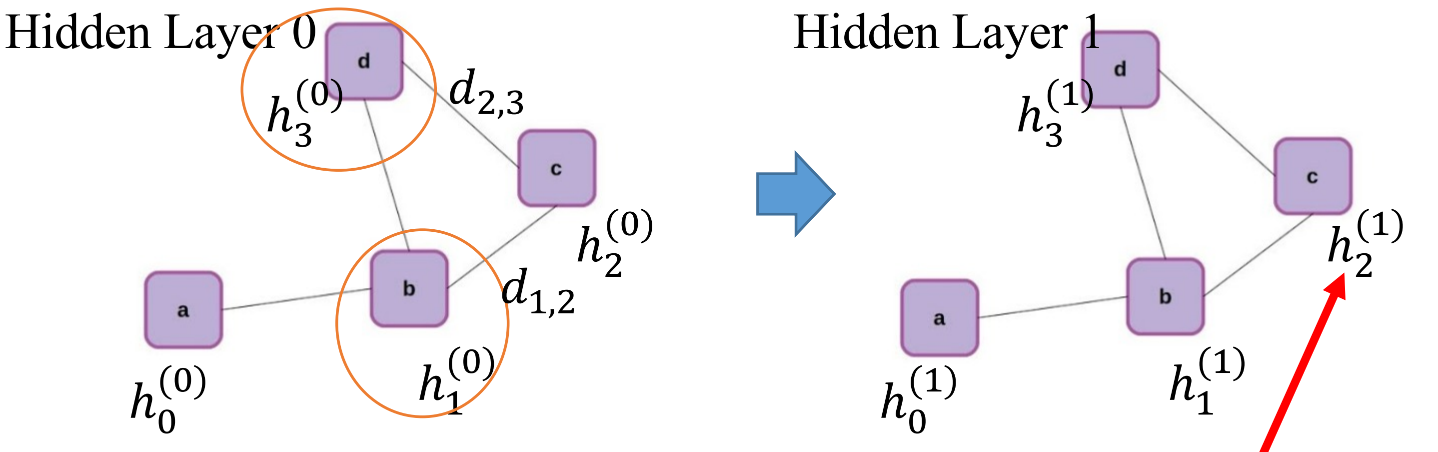


$$d = \text{distance} = d_{x,y} = \left(\frac{1}{\sqrt{degree(x)}}, \frac{1}{\sqrt{degree(y)}}\right)^T$$

degree = degree of node (connect to how many node)

$$h_1^{(1)} = w(d_{1,2}) \times h_1^{(0)} + w(d_{2,3}) \times h_3^{(0)}$$

Aggregation: Weighted Sum



$d_{x,y}$

MoNET (Mixture Model Network)

[https://arxiv.org/pdf/1611.0 8402.pdf](https://arxiv.org/pdf/1611.08402.pdf)

node)

Geometric deep learning on graphs and manifolds using mixture model CNNs

Federico Monti[1*] Davide Boscaini[1*] Jonathan Masci[1,4]
Emanuele Rodolà[1] Jan Svoboda[1] Michael M. Bronstein[1,2,3]
[1]USI Lugano [2]Tel Aviv University [3]Intel Perceptual Computing [4]Nnaisense

Aggregation: Weighted Sum

[Submitted on 30 Oct 2017 (v1), last revised 4 Feb 2018 (this version, v3)]

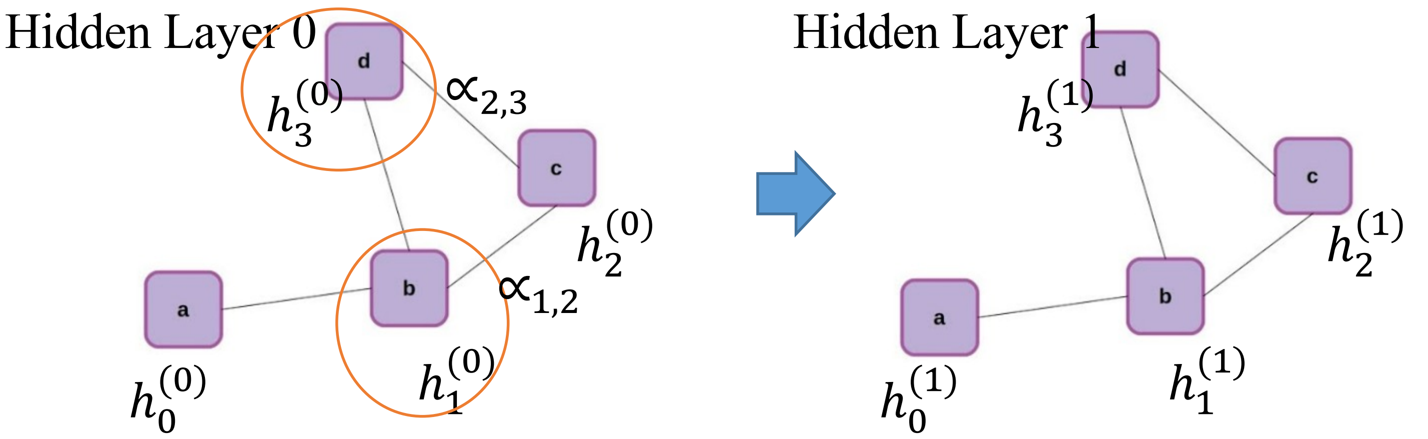
Graph Attention Networks

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio

We present graph attention networks (GATs), novel neural network architectures that operate on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations. By stacking layers in which nodes are able to attend over their neighborhoods' features, we enable (implicitly) specifying different weights to different nodes in a neighborhood, without requiring any kind of costly matrix operation (such as inversion) or depending on knowing the graph structure upfront. In this way, we address several key challenges of spectral-based graph neural networks simultaneously, and make our model readily applicable to inductive as well as transductive problems. Our GAT models have achieved or matched state-of-the-art results across four established transductive and inductive graph benchmarks: the Cora, Citeseer and Pubmed citation network datasets, as well as a protein-protein interaction dataset (wherein test graphs remain unseen during training).

Useful strategy for increasing the representational capacity of a GNN model, especially in cases where you have prior knowledge to indicate that some neighbors might be more informative than others

Aggregation: Weighted Sum

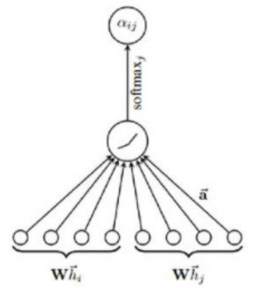


$$h_2^{(1)} = \overline{w}_0(x_2) + \propto_{1,2}\left(h_1^{(0)}\right) + \propto_{2,3}\left(h_3^{(0)}\right)$$

Parameter for feature Transformation

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T[\boxed{\mathbf{W}}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\boxed{\vec{a}^T}[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Parameter of Attention mechanism



Attention Mechanism in GAT

$\propto$ = attention on neighbour

Spatial-based Convolution

Aggregate local network neighbourhoods:

- Sum
- Mean
- Weighted sum
- LSTM
- Max pooling

SNAP

SNAP Code

GraphSAGE: Inductive Representation Learning on Large Graphs

GraphSAGE is a framework for inductive representation learning on large graphs. GraphSAGE is used to generate low-dimensional vector representations for nodes, and is especially useful for graphs that have rich node attribute information.

[Submitted on 30 Oct 2017 (v1), last revised 4 Feb 2018 (this version, v3)]

Graph Attention Networks

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio

We present graph attention networks (GATs), novel neural network architectures that operate on graph-structured data, leveraging masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations. By stacking layers in which nodes are able to attend over their neighborhoods' features, we enable (implicitly) specifying different weights to different nodes in a neighborhood, without requiring any kind of costly matrix operation (such as inversion) or depending on knowing the graph structure upfront. In this way, we address several key challenges of spectral-based graph neural networks simultaneously, and make our model readily applicable to inductive as well as transductive problems. Our GAT models have achieved or matched state-of-the-art results across four established transductive and inductive graph benchmarks: the Cora, Citeseer and Pubmed citation network datasets, as well as a protein-protein interaction dataset (wherein test graphs remain unseen during training).

Thank You