



Learn Machine Learning in an hour

- Intuitions
- Hand-on examples
- Resources for next steps

Alex Pang July 2023
For C.-R. Ji research group

Outline

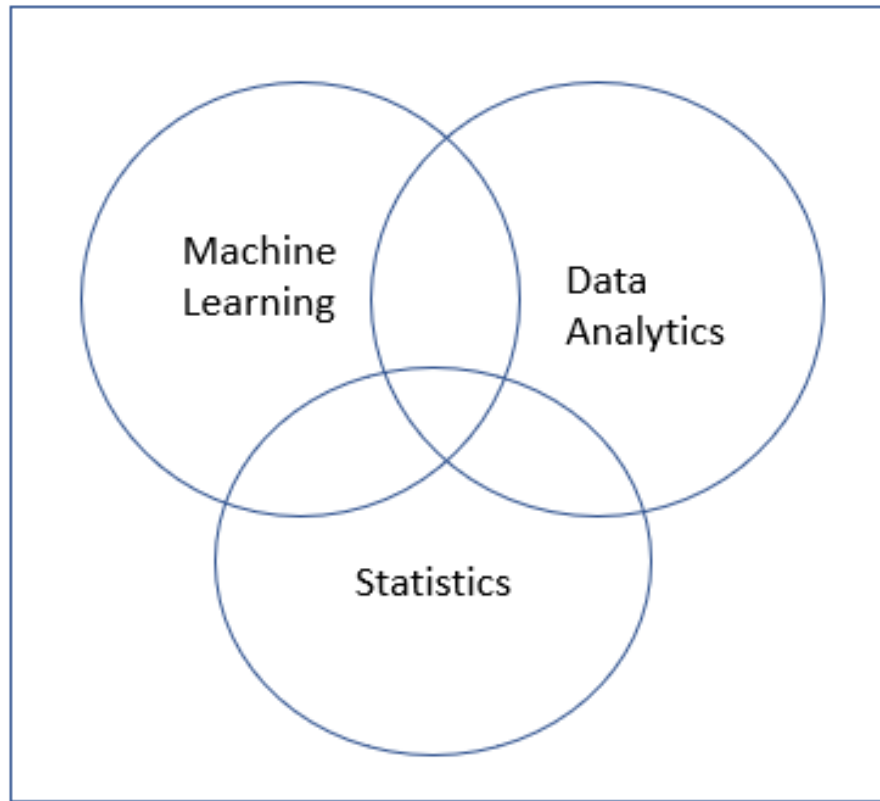
- **Overview**
- Classical Machine Learning
- Deep Learning
- Generative AI
- List of Resources
- Relation to Physics (if time allows)

Warnings

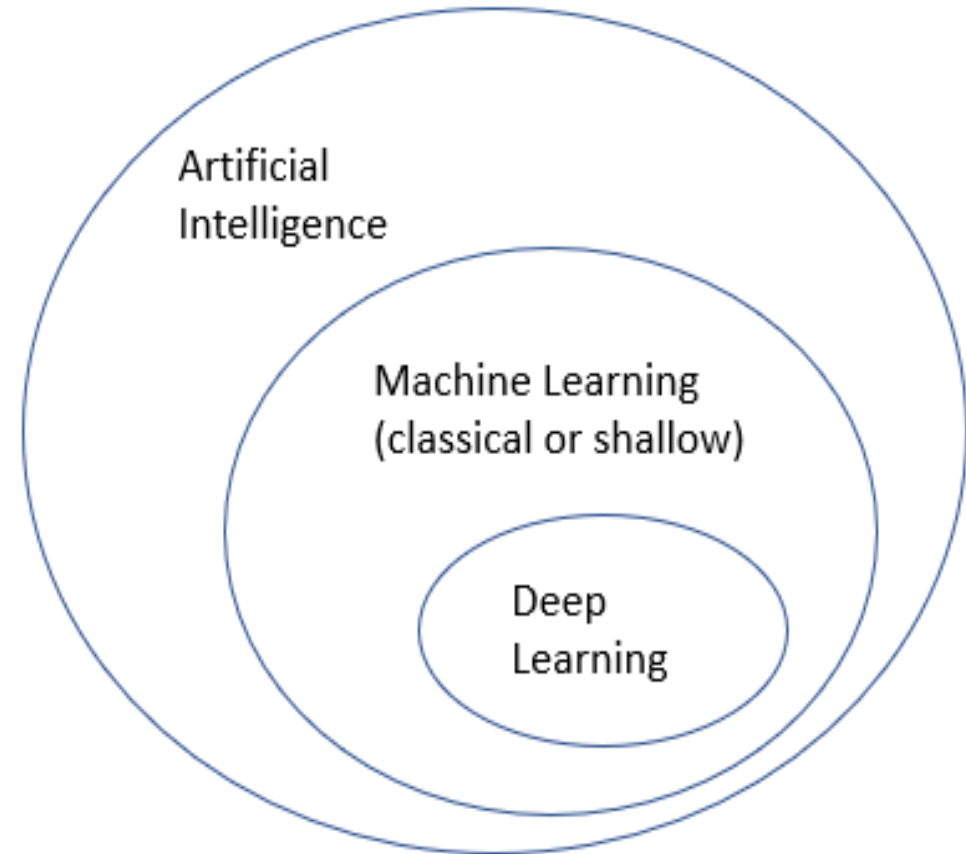
Basically, I will go over what are classical mechanics, electrodynamics, quantum physics and general relativity under one hour

Focus on **important concepts** and **intuitions**

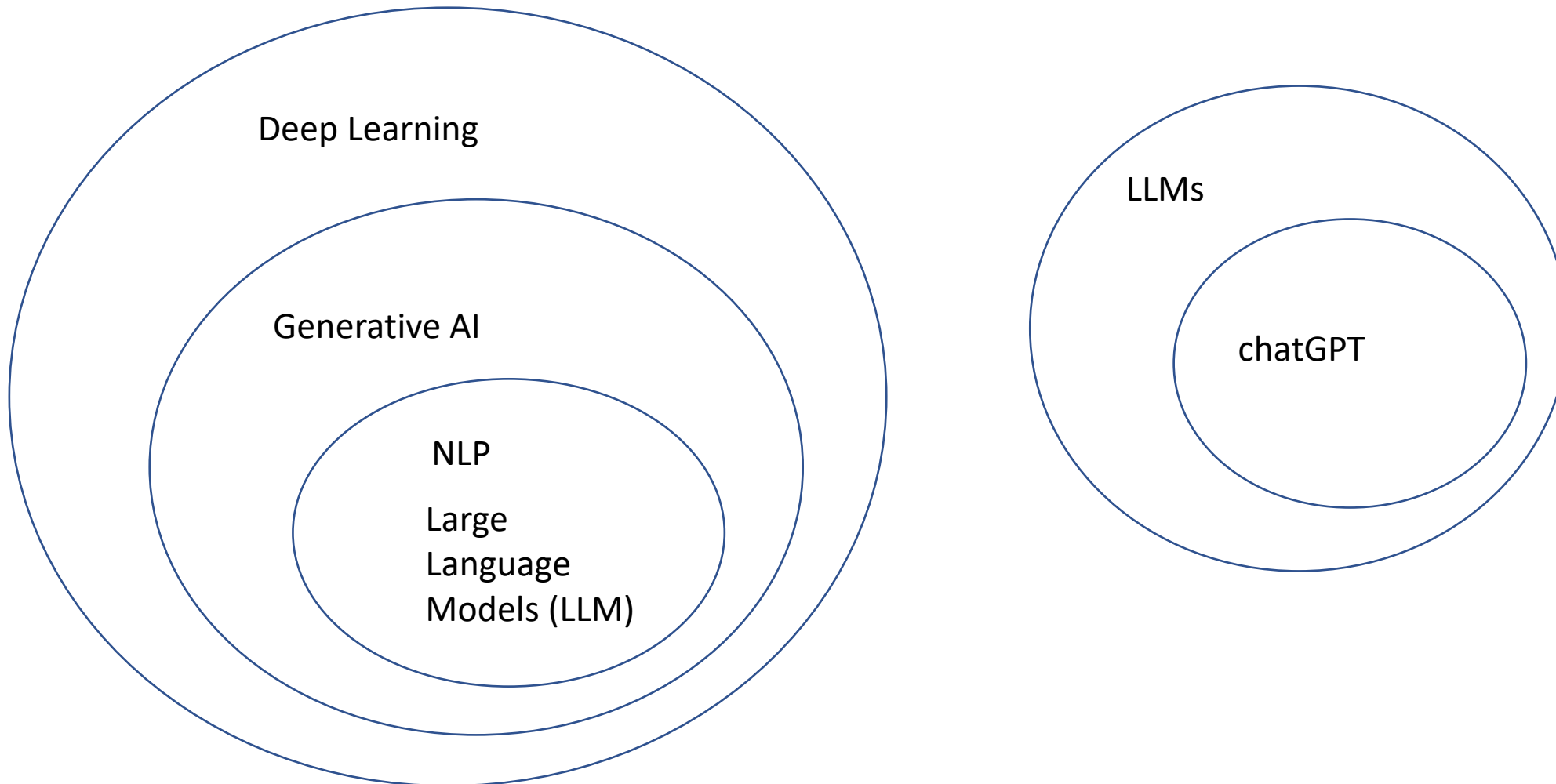
Relations among AI, Machine Learning, deep learning etc



Data Science



Relations among AI, Machine Learning, deep learning etc.



ChatGPT as a Large Language Models

Large Language Models



Categories of Machine Learning

1. Unsupervised

- Algorithm attempts to learn meaningful structures in the data
- Examples are k-means/hierarchical clustering

2. Supervised

- Involves output label associated with each instance in the dataset
 1. Real-Valued
 - Regression Models
 2. Discrete/Categorical
 - Classification models

Recall from Patrick's ML slides

- “Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then **make a determination or prediction** about something in the world.” – [Nvidia](#)
- “Machine learning is the science of getting computers to act **without being explicitly programmed.**” – [Stanford](#)
- “Machine learning is based on algorithms that can learn from data **without relying on rules-based programming.**” - [McKinsey & Co.](#)
- “Machine learning algorithms can figure out how to perform important tasks **by generalizing from examples.**” – [University of Washington](#)
- “The field of Machine Learning seeks to answer the question “How can we build computer systems that **automatically improve with experience**, and what are the fundamental laws that govern all learning processes?” – [Carnegie Mellon University](#)

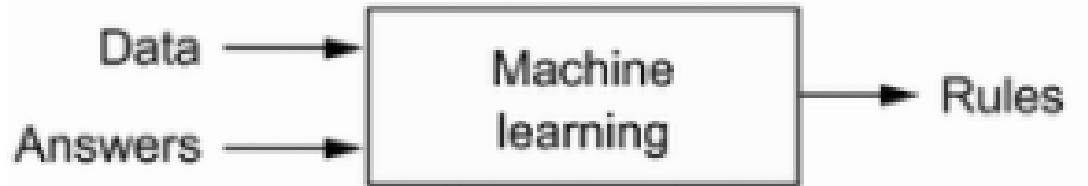
What is ML? How it works

Symbolic AI / Experts system



- no concepts of learning
- need domain experts
- Lots of if else statements
- Hard to cover all cases

Machine Learning



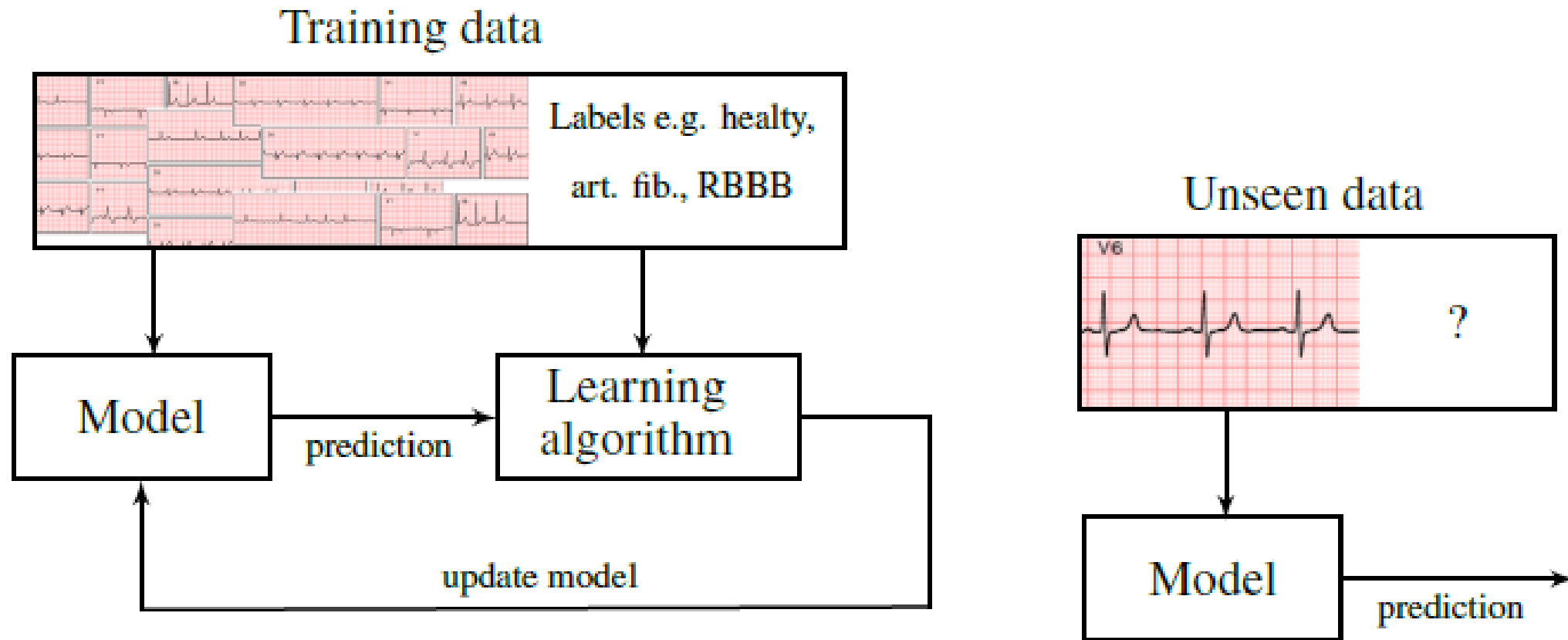
- Assume certain model structure (hypothesis space)
- Use optimization to empirically find model coefficients
- No details rules from domain experts, but need them for identify important factors/features

A Machine Learning Example

Electrocardiogram (ECG) examination



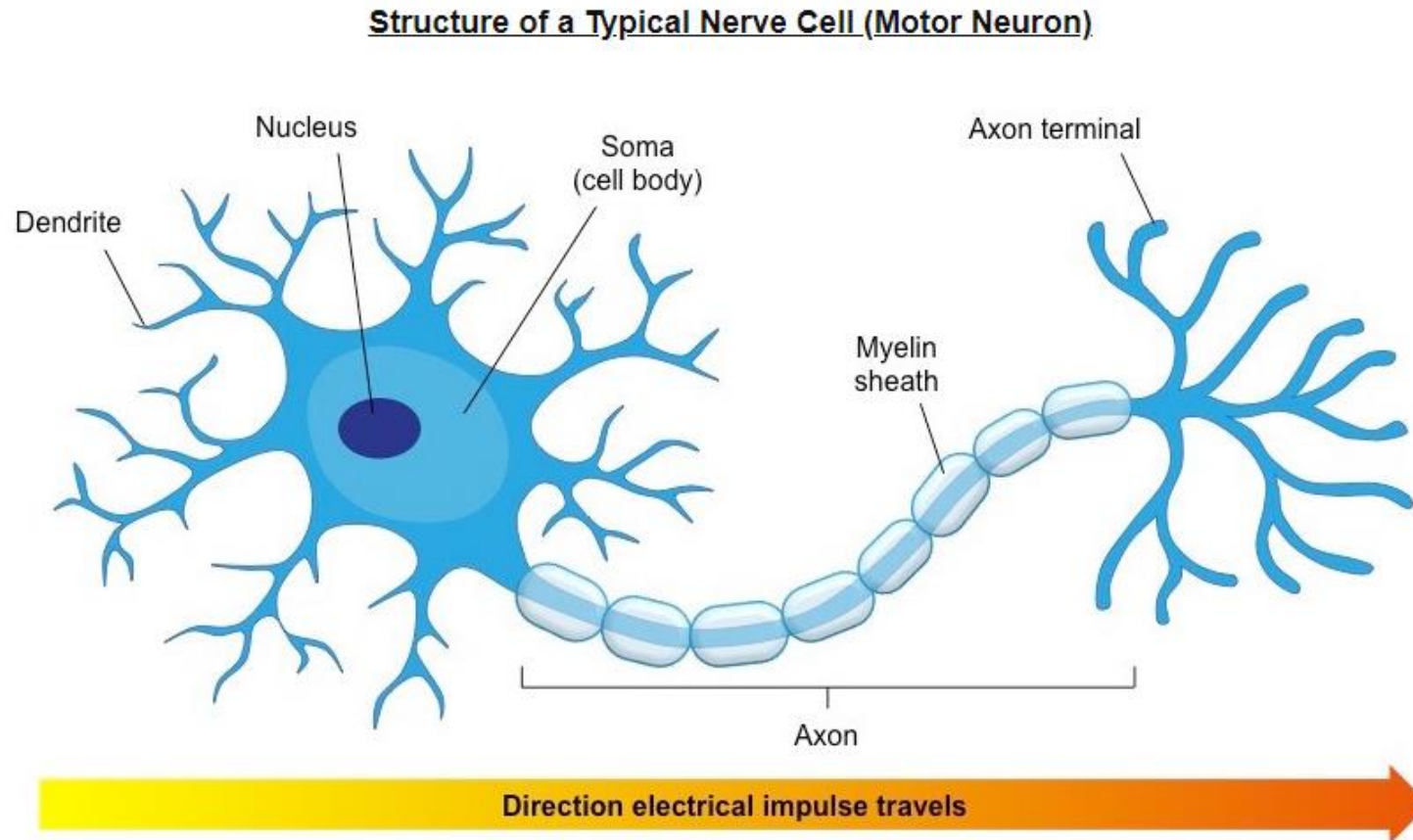
A Machine Learning Example (continued)



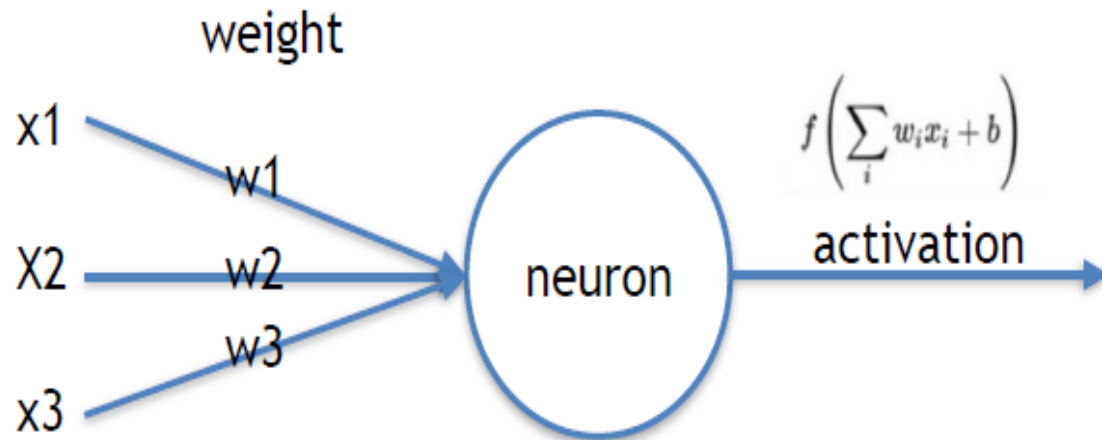
There are different kinds of Models

Neural Network (as a special kind of model)

- Originally inspired by how the Brain works
- Our brain has $\sim 10^{11}$ neurons, each of which communicates to $\sim 10^4$ other neurons



Mathematical representation of One Neuron

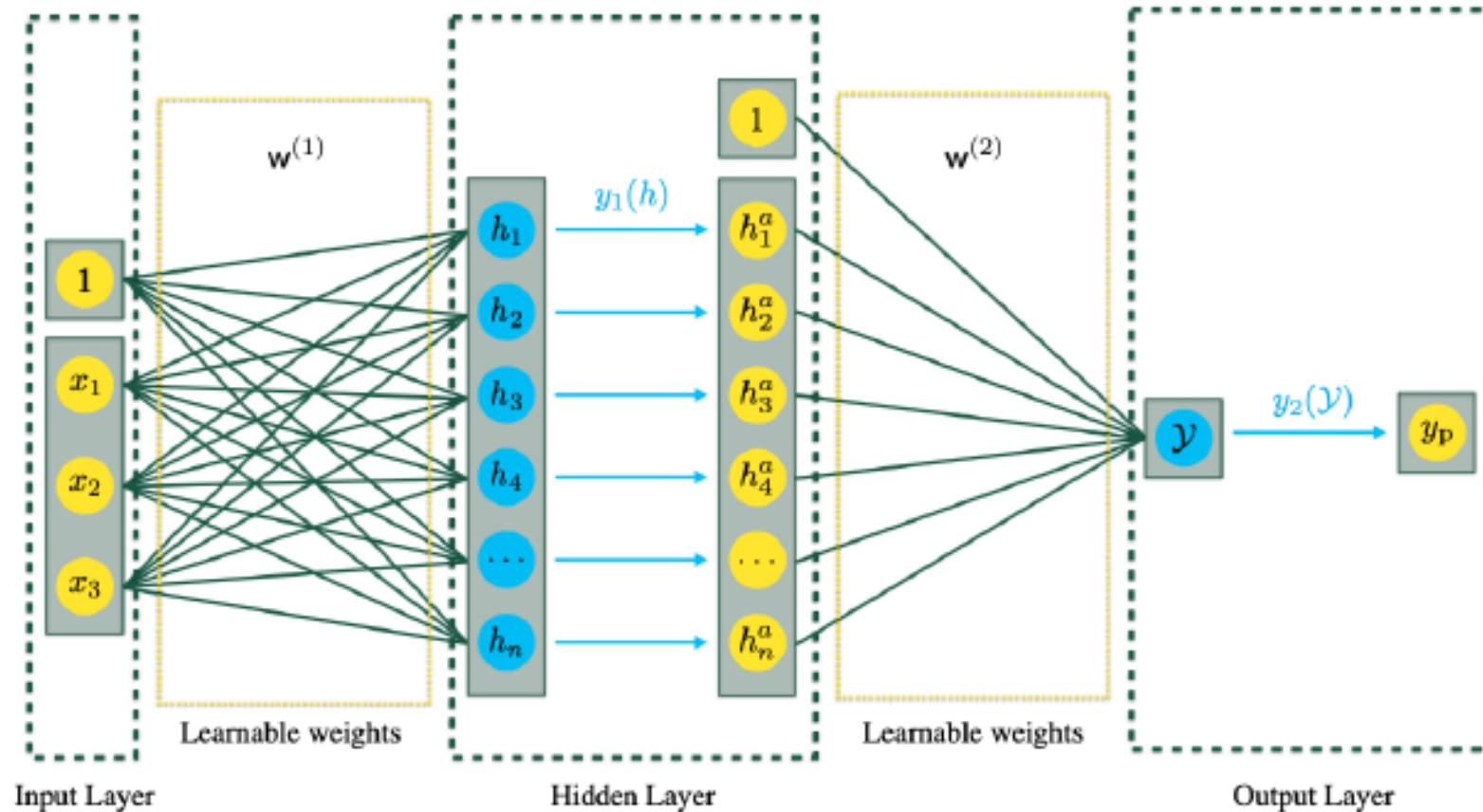


Think of x 's as electrical signals propagating through the axons from one neuron to the next

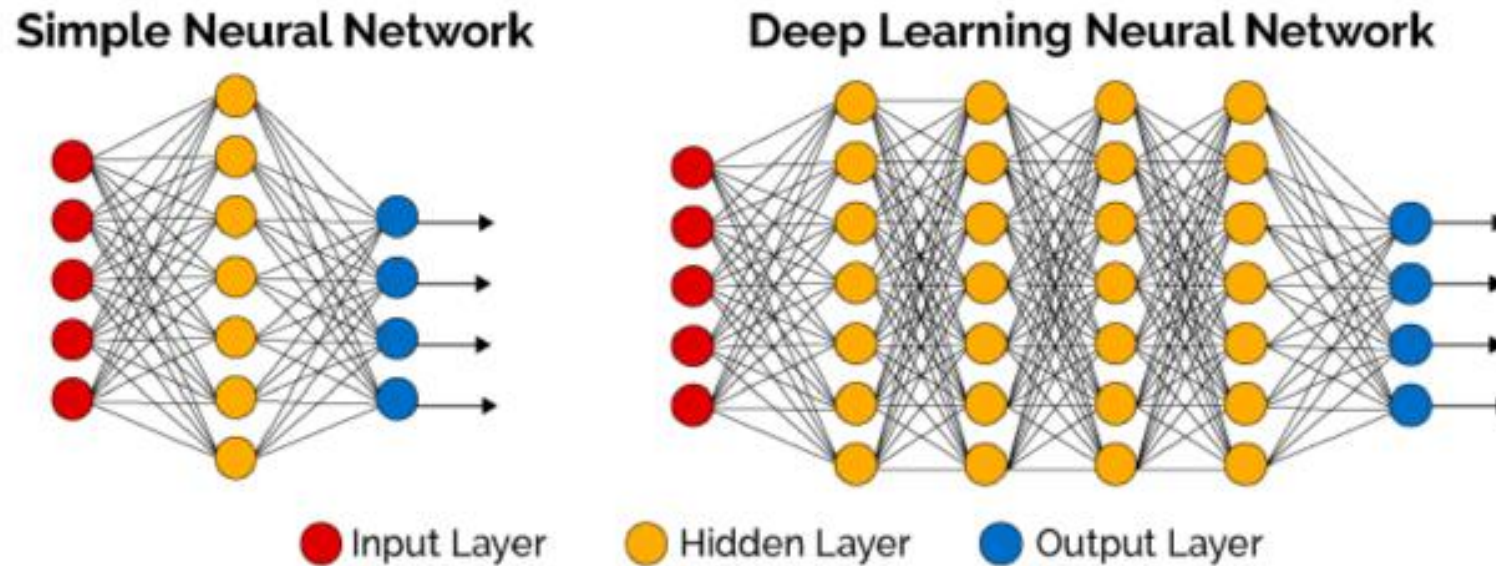
<https://cs231n.github.io/neural-networks-1/>

Neural Network

x 's are the input, y is the output



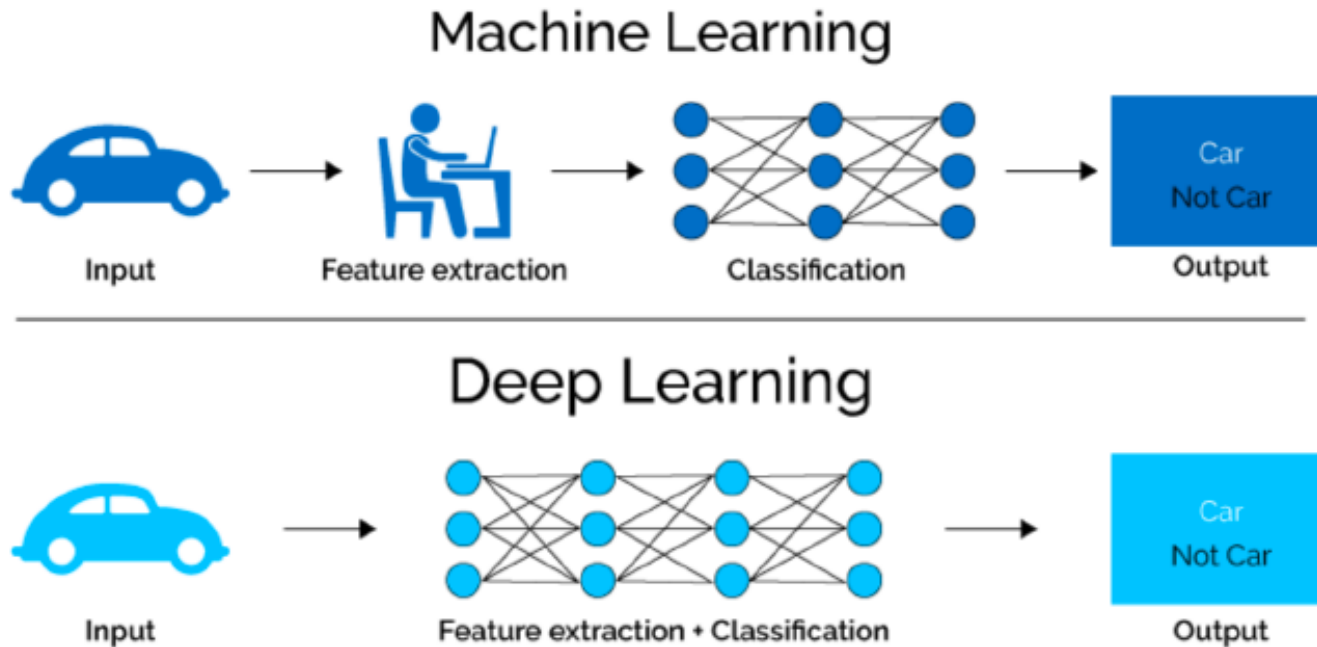
What is Deep Learning? How it works



Pic Credit: Xenonstack | Simple Neural Network and Deep Neural Network

- Lots of model parameters (170 Billions in the latest LLM)
- Requires lots of Data and Computational Power
- Un-heard of from traditional Statistics perspective

How and Why DL works



Pic Credit: Xenonstack | Machine Learning vs Deep Learning

- Universal Function Approximator
- Automatic Feature Engineering
- Deep Learning offers much bigger hypothesis search space

How and why DL works

- Layers from input to the output represent a series of adaptable non-linear transformation
- Internal nodes represent hidden features that are found automatically. Those features may not be obvious to human domains
- Uncrumpling paper balls is what deep learning is about: finding neat representations for complex, highly folded data manifolds; incrementally decomposing a complicated geometric transformation into a long chain of elementary ones, which is pretty much the strategy a human would follow to uncrumple a paper ball.

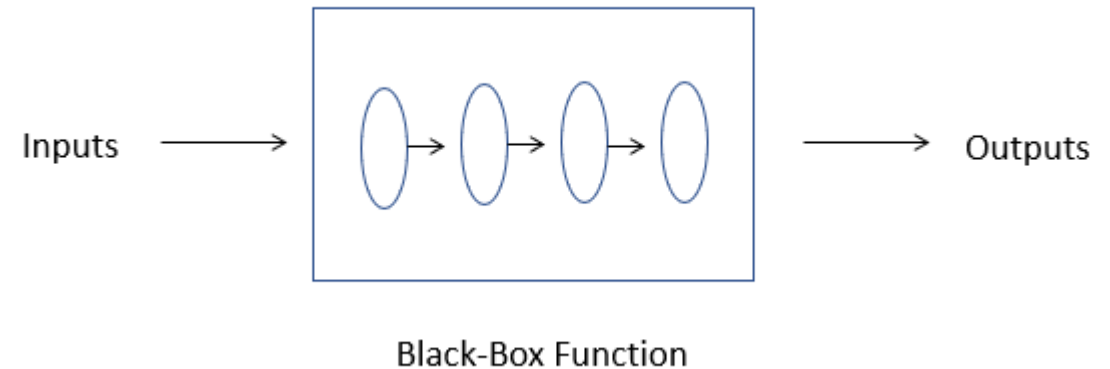


Figure 2.14 Uncrumpling a complicated manifold of data

Outline

- Overview
- **Classical Machine Learning**
- Deep Learning
- Generative AI
- List of Resources

Classical Machine Learning

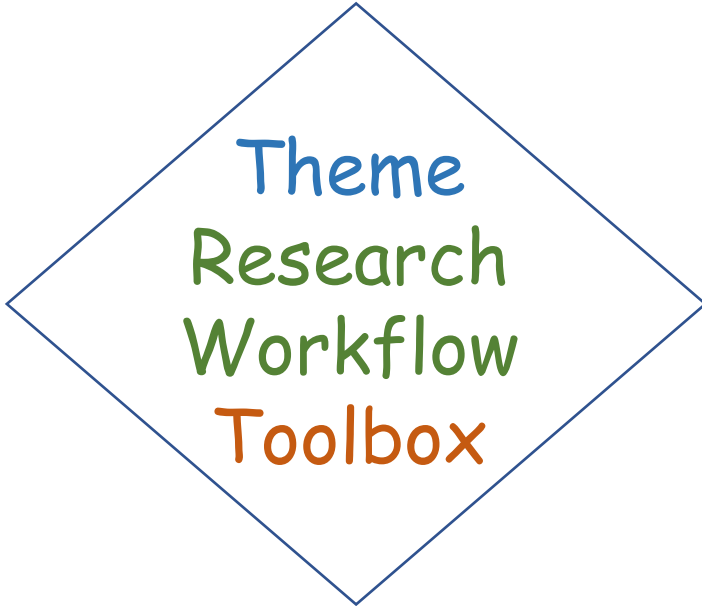
Supervised Learning (One of the variable, referred as target variable, specify the so-called “model answer”)	Classification – Machine Learning	Decision Trees K-Nearest Neighbors Support Vector Machine
		Neural Networks
	Classification - Statistics	Regression;
Unsupervised Learning (Exploratory analysis to discover patterns, no target variable)	Clustering Analysis	K-Means
	Association Rules Dimension Reduction	Apriori Principal Components Analysis

Learn the best practice in Data Science

Apply different algorithms to solve different problems
based on the same theme and research workflow

Algorithms

- SVM
- KNN
- Naïve Bayes
- Neural Network
- Logistics Regression
- NLP



Theme
Research
Workflow
Toolbox

Problems

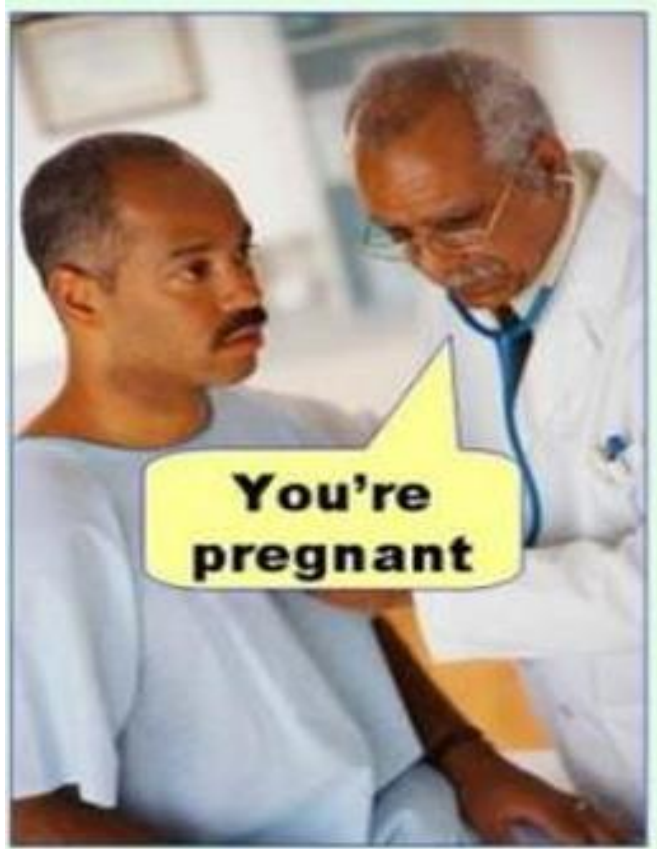
- Regression
- Classification
- Recommendation System
- Clustering
- Association

Common Terminology in Machine Learning



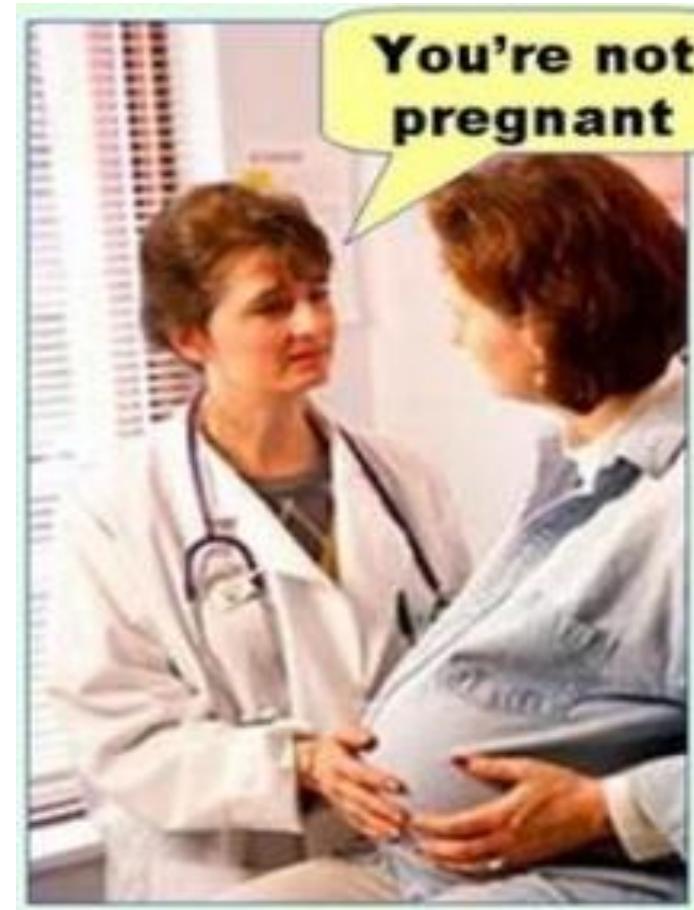
Everyone makes error

Type I error



False Positive

Type II error



False Negative

Linear Regression (Classical Example)

Predicting house price based on features such as "sum of bathrooms", "square foot", "crime rate in the neighborhood", "school performance" etc

X_1	X_2	X_3	...	X_{D-1}	X_D	Y
0.23	0.79	0.33	...	-2.4	0.91	4.5
		
-0.34	0.44	-0.32	...	0.11	0.007	3.7
0.40	-0.68	0.82	...	-0.45	0.004	-4.3

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1x_1 + \dots + w_Dx_D$$

How to find the best fitting line?

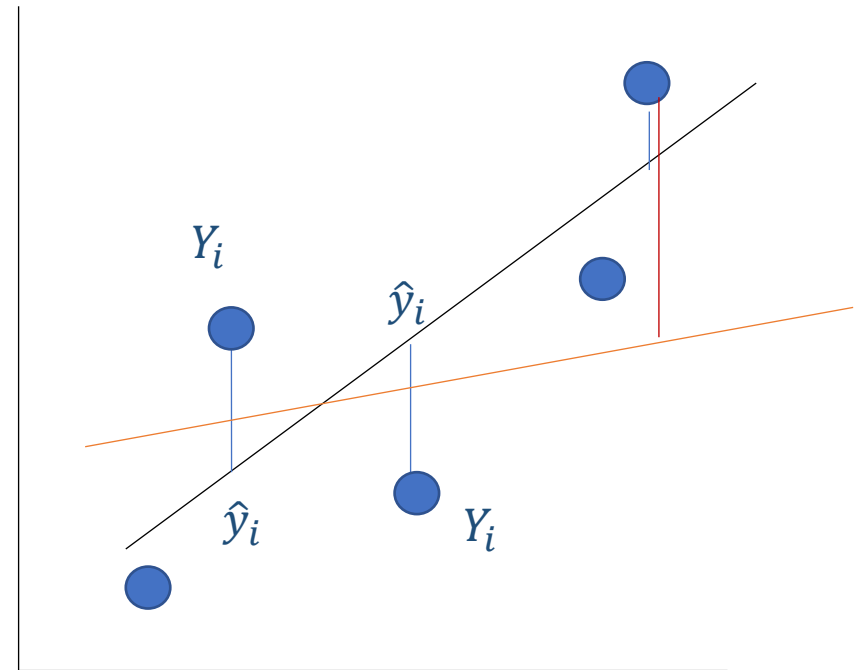
Define Mean Squared Error (MSE)
To be the square of the distance
between actual and predict Y values

$$\text{MSE} = \frac{1}{N} \sum_i^n (y_i - \hat{y}_i)^2$$

Best fitted line is the line that
minimize the MSE =>

Least Square Methods

\hat{y}_i = prediction, Y_i = actual value



Learning Algorithm

- Define a cost function or Loss function (e.g. MSE, Likelihood function for classification), usually denoted as $J(\theta)$ or $L(w)$, where θ or w are the model parameters
- Use an optimization engine such as Gradient descent to find the optimal model parameters
- Use Regularization to penalize models that have too much parameters or any one model parameter too big to overfit in-sample data

$$\mathcal{L}(w) + \lambda \cdot \mathcal{R}(w)$$

$$\mathcal{R}(w) = \sum_{i=1}^k |w_i|$$

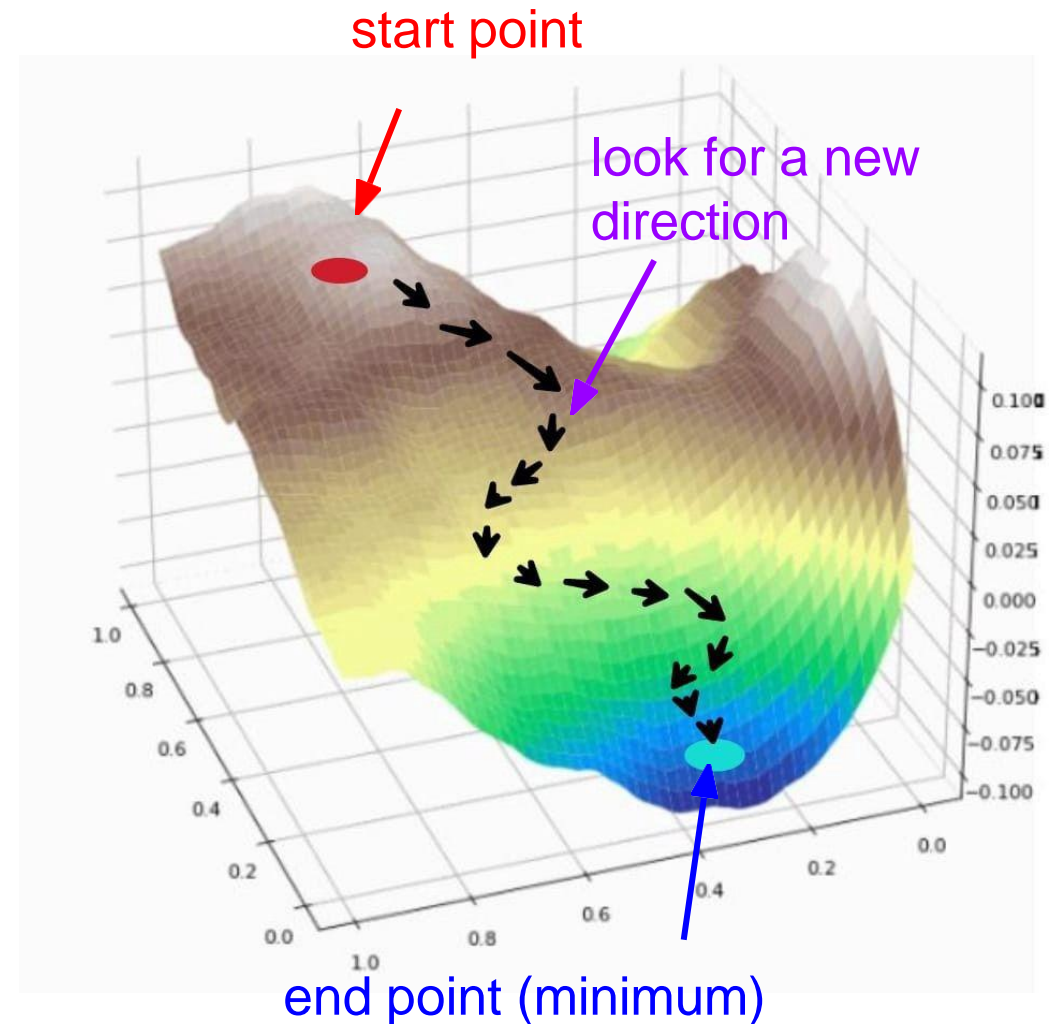
Gradient Descent (2-dimensional)

Idea:

- ❑ start somewhere
- ❑ take steps based on the gradient vector of the current position till convergence

Convergence:

- ❑ happens when changes between two steps $< \epsilon$



Gradient Descent

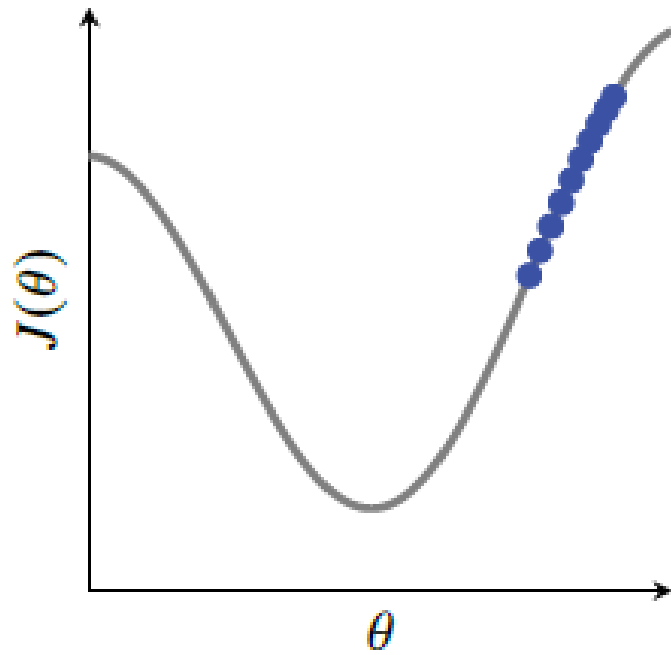
Algorithm 5.1: Gradient descent

Input: Objective function $J(\theta)$, initial $\theta^{(0)}$, learning rate γ

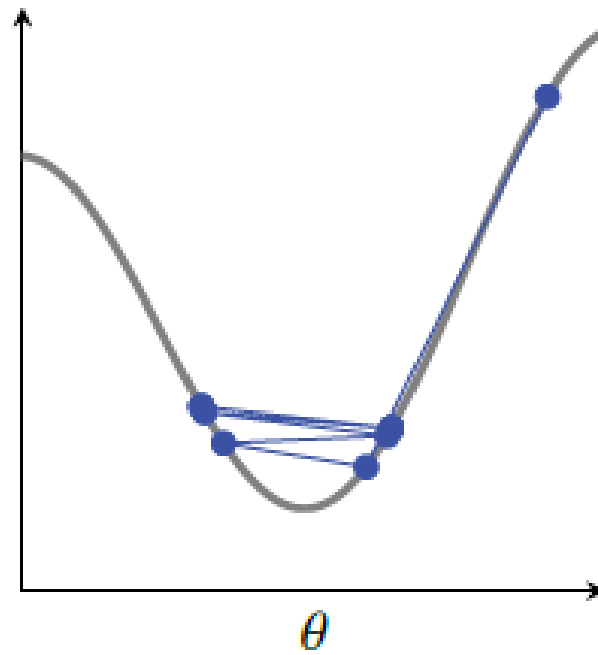
Result: $\hat{\theta}$

- 1 Set $t \leftarrow 0$
 - 2 **while** $\|\theta^{(t)} - \theta^{(t-1)}\|$ *not small enough* **do**
 - 3 | Update $\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma \nabla_{\theta} J(\theta^{(t)})$
 - 4 | Update $t \leftarrow t + 1$
 - 5 **end**
 - 6 **return** $\hat{\theta} \leftarrow \theta^{(t-1)}$
-

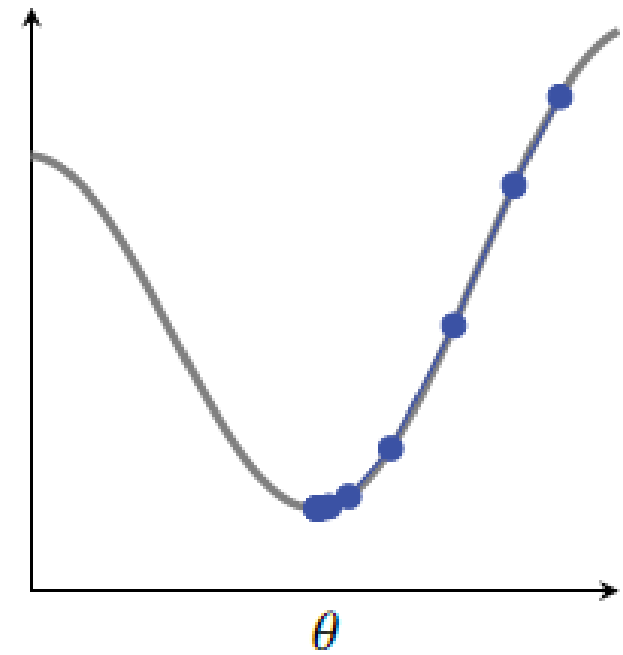
Effect of different learning rate



(a) Low learning rate $\gamma = 0.05$



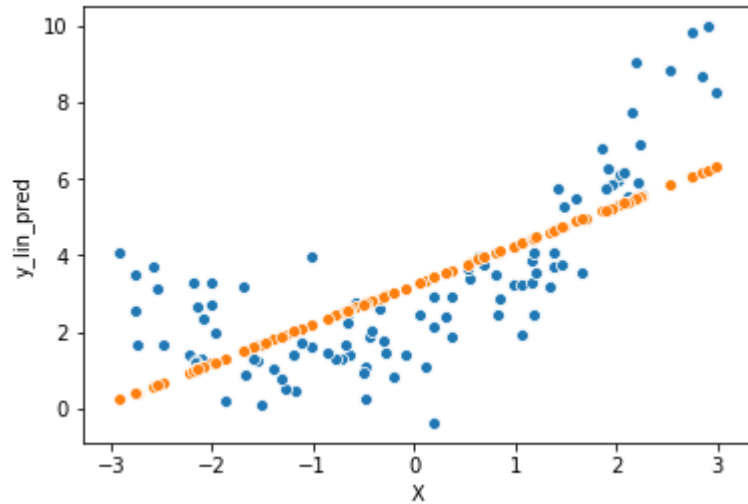
(b) High learning rate $\gamma = 1.2$



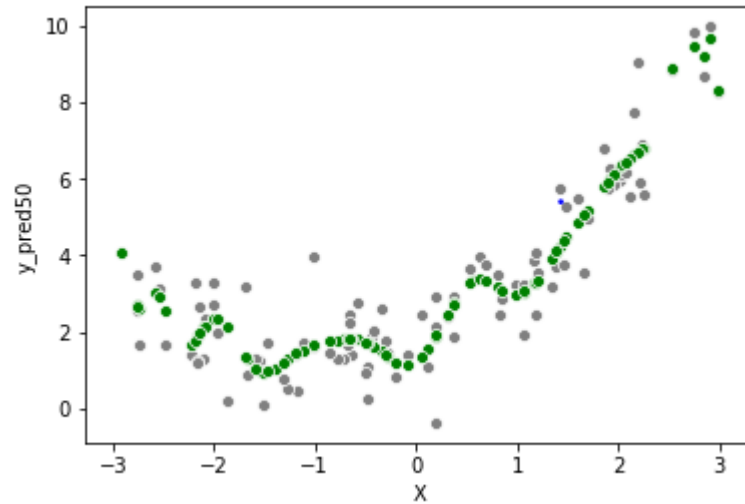
(c) Good learning rate $\gamma = 0.3$

Lessons Learned from Polynomial Regression

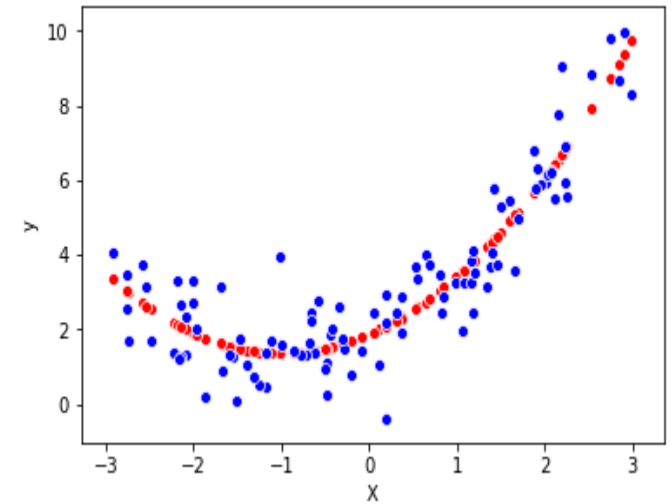
Underfit



Overfit



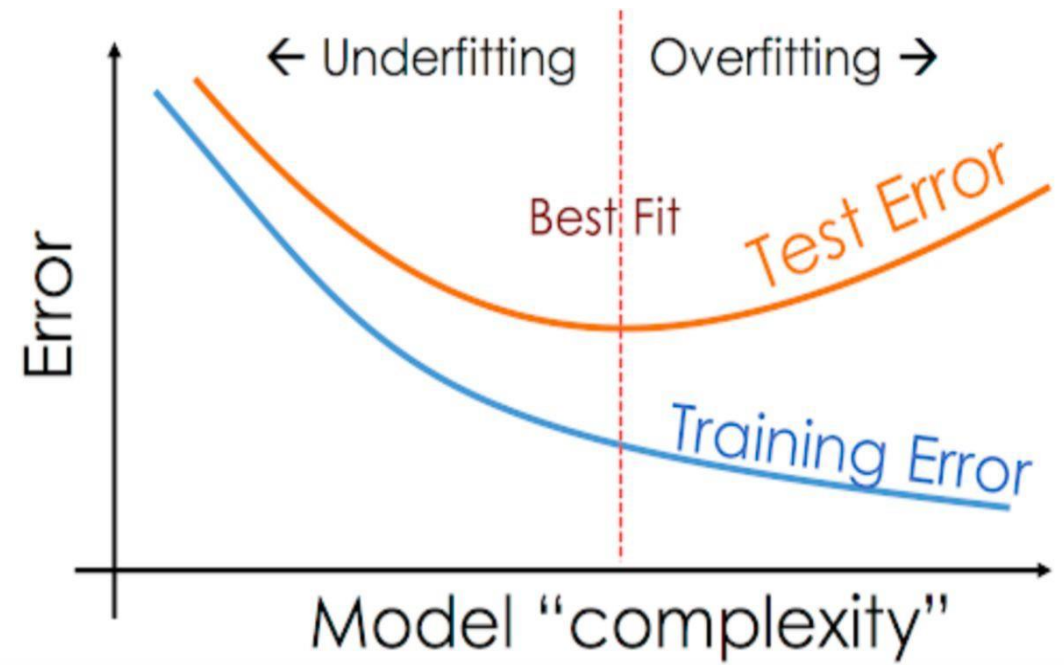
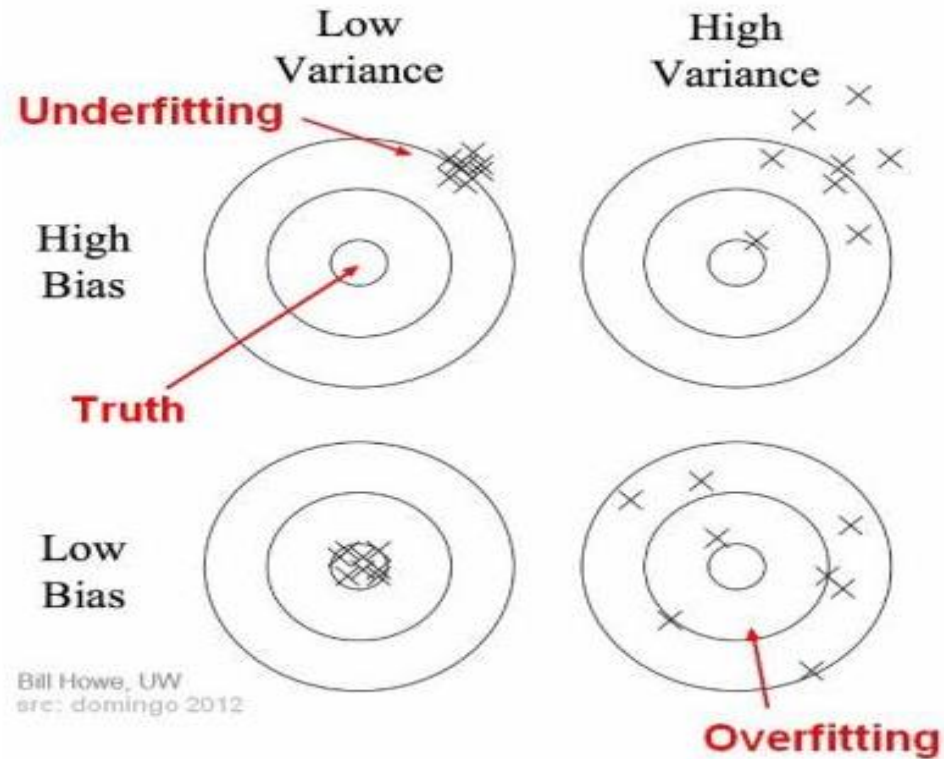
Good fit



A more sophisticated model tends to have smaller errors in the training set, but can perform worse in testing dataset because it overfit




A too simplistic model will never be able to fit well on both the training set as well as the testing dataset

Bias vs Variance Tradeoff


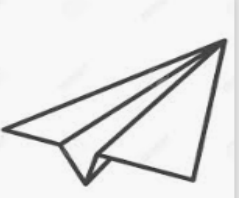






Designing an airplane flying from DC to NY

Any Big Bang Theory Fans here?

Modeler	Design	Bias (low/high)	Variance (low/high)	Risk (over or underfit)
	?	?	?	?
	?	?	?	?
	?	?	?	?

Designing an airplane flying from DC to NY

Modeler	Design	Bias (low/high)	Variance (low/high)	Risk (over or underfit)
		High	Low	underfit
		Low	High	overfit
		Low	Low	goodfit

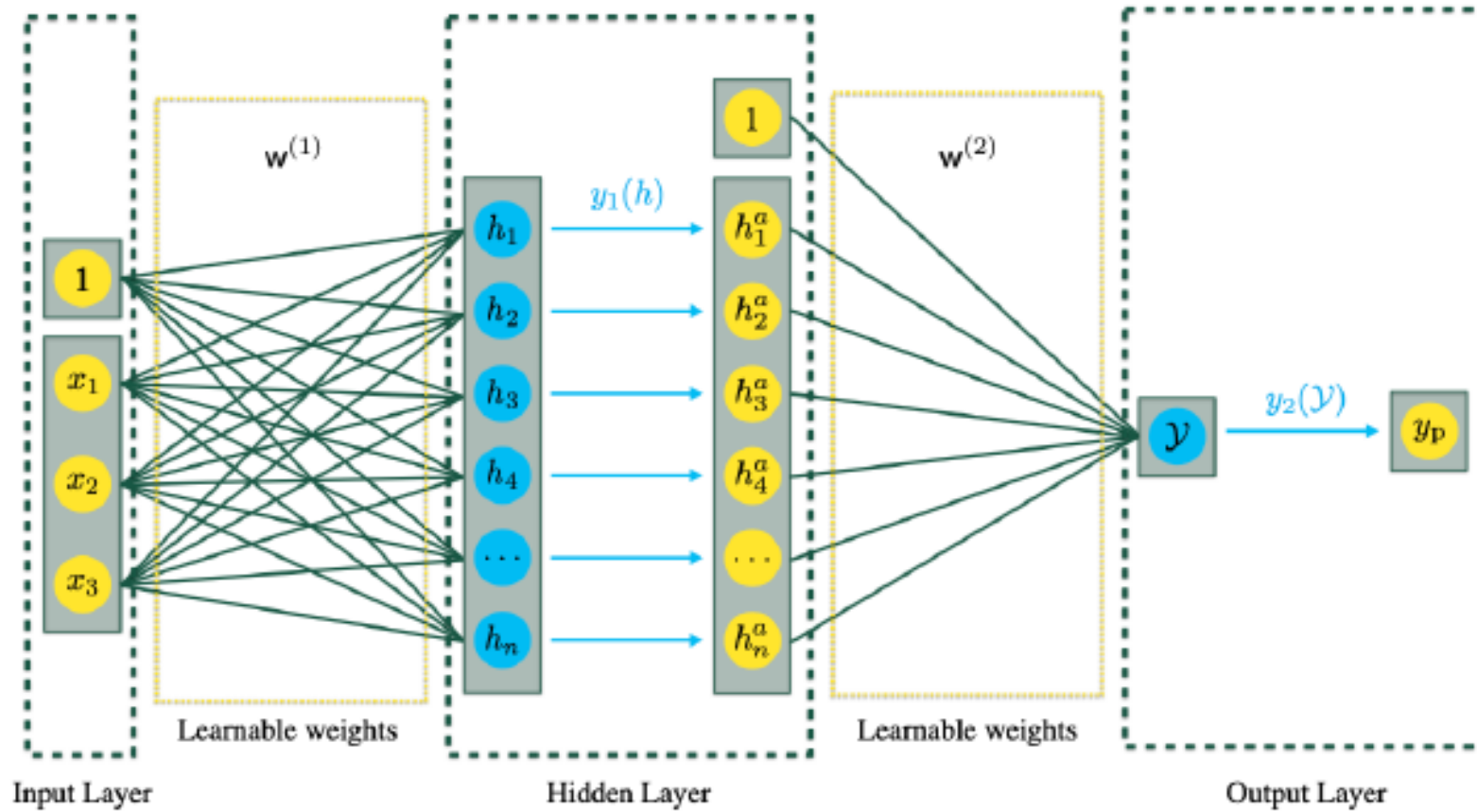
Outline

- Overview
- Classical Machine Learning
- Deep Learning
- Generative AI
- List of Resources

Common
Terminology
in Deep
Learning

output layer
input layer stochastic gradient desce
learning rate backward pass
loss function drop out
backpropagation
tensor hidden layer
activation function adagrad
forward pass
gradient descent rmsprop optimizer
dot product

Recall Patrick's slide



Concepts in Deep Learning

- Many concepts in classical ML applies to DL as well
- **Forward Pass** - for given sets of weights calculate the target variables and hence the loss function
- **Back Propagation** - efficient algorithm to prescribe how the weights should be adjusted for each iteration of the learning algorithms so that in the next iteration, the error is smaller
- **Dropout** - temporarily disable some internal nodes in some iterations of the weights as a way to avoid overfitting
- **Stochastic Gradient Descent** - instead of using all data at the same time, we random select a batch of samples for each iteration of the learning algorithms

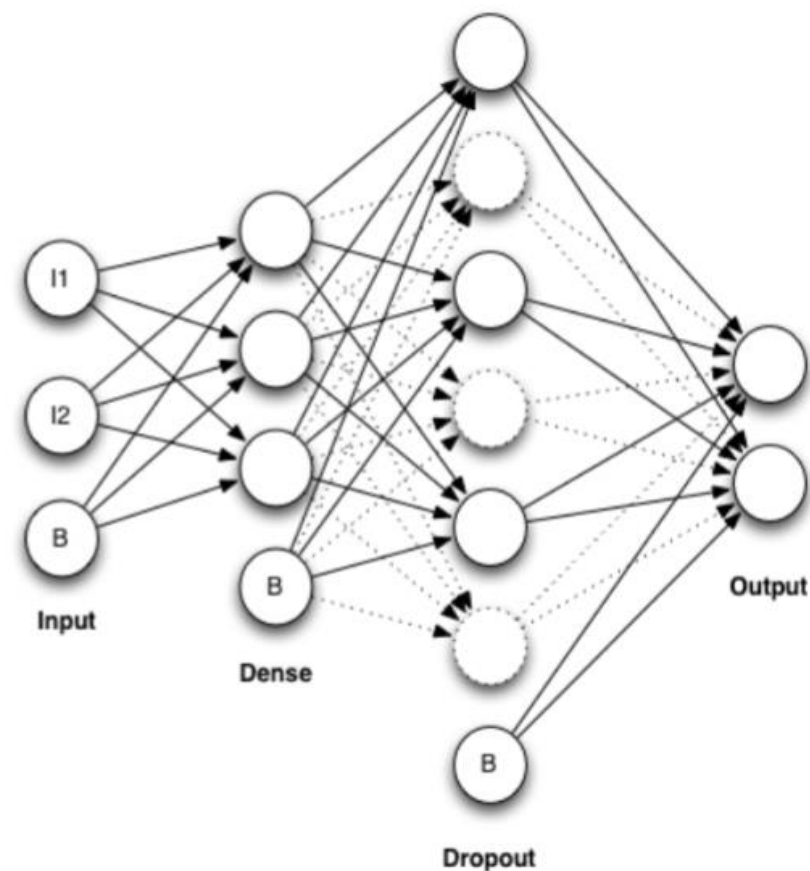
Dropout

A Dropout layer is a dense layer that temporarily eliminate some of its neurons.

The discarded neurons are not permanently removed but being temporarily disabled. They are represented by the dash line.

Each dropped neuron connection will not be trained so its weights will not be based on trying to best fitted the training dataset. The final neural network will use *ALL* of these neurons for the output

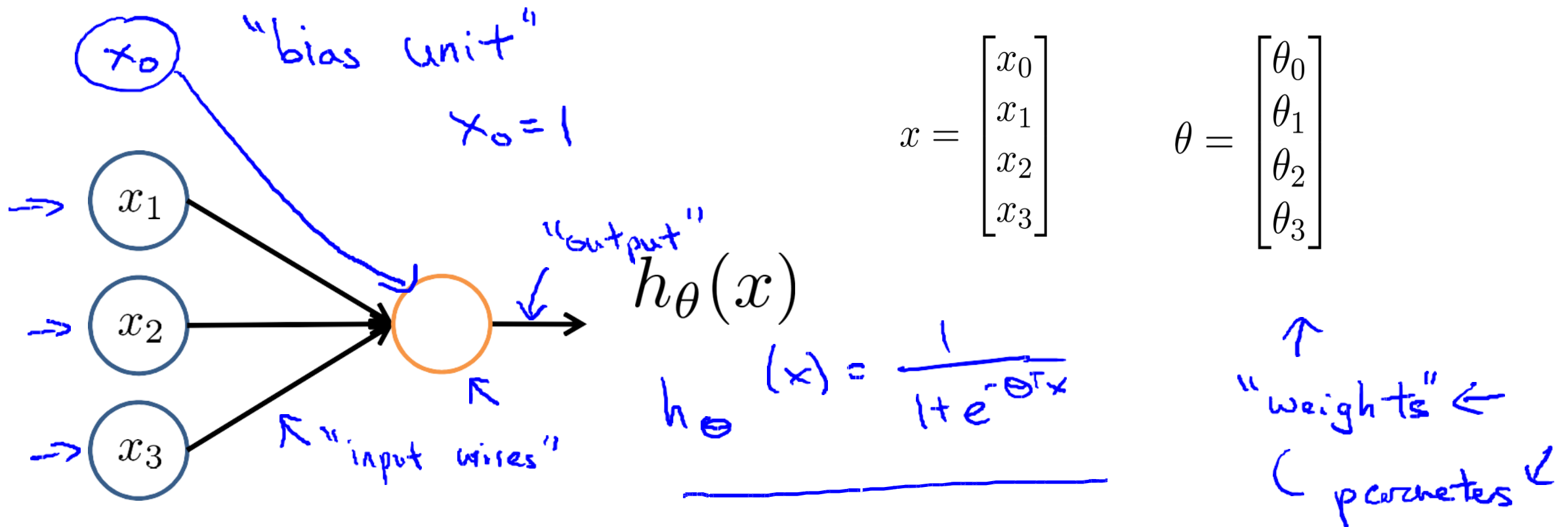
The goal of having a dropout layer is to decrease the chance of overfitting



Technical details

- Forward Propagation
 - Given any weights, calculate the prediction
- Back Propagation
 - Iterative algorithm on how to adjust the weights to minimize the error

Neuron model: Logistic unit

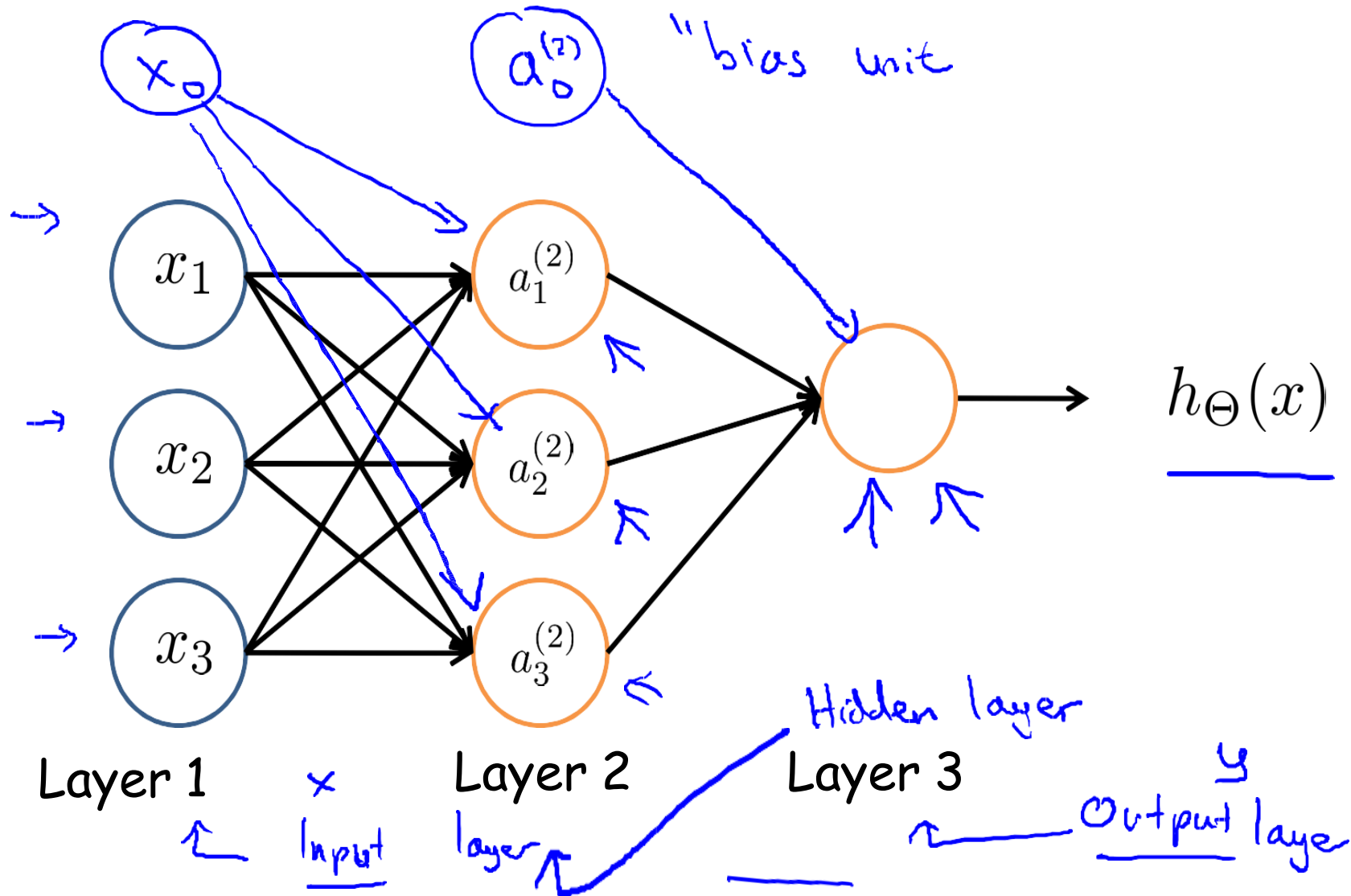


Sigmoid (logistic) activation function.

$$g(z) = \frac{1}{1 + e^{-z}}$$

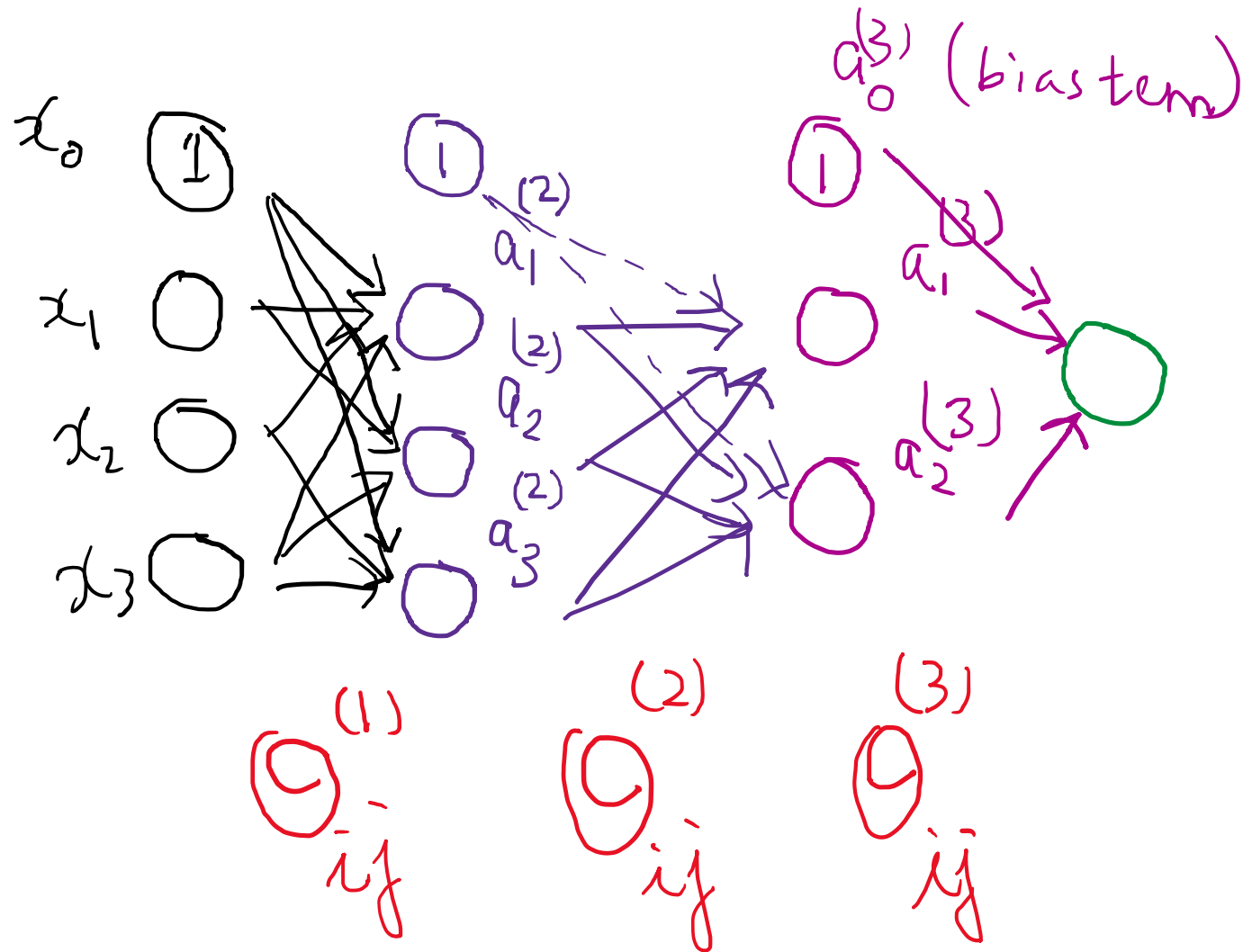
Can choose a different activation function $g(z)$ such as ReLU(z)

Neural Network



Forward Propagation

input layer \Rightarrow 1st hidden layer \Rightarrow 2nd hidden layer ...



$$x_0 = 1 = a_0^{(1)}$$

$$x_1 = a_1^{(1)}$$

$$x_2 = a_2^{(1)}$$

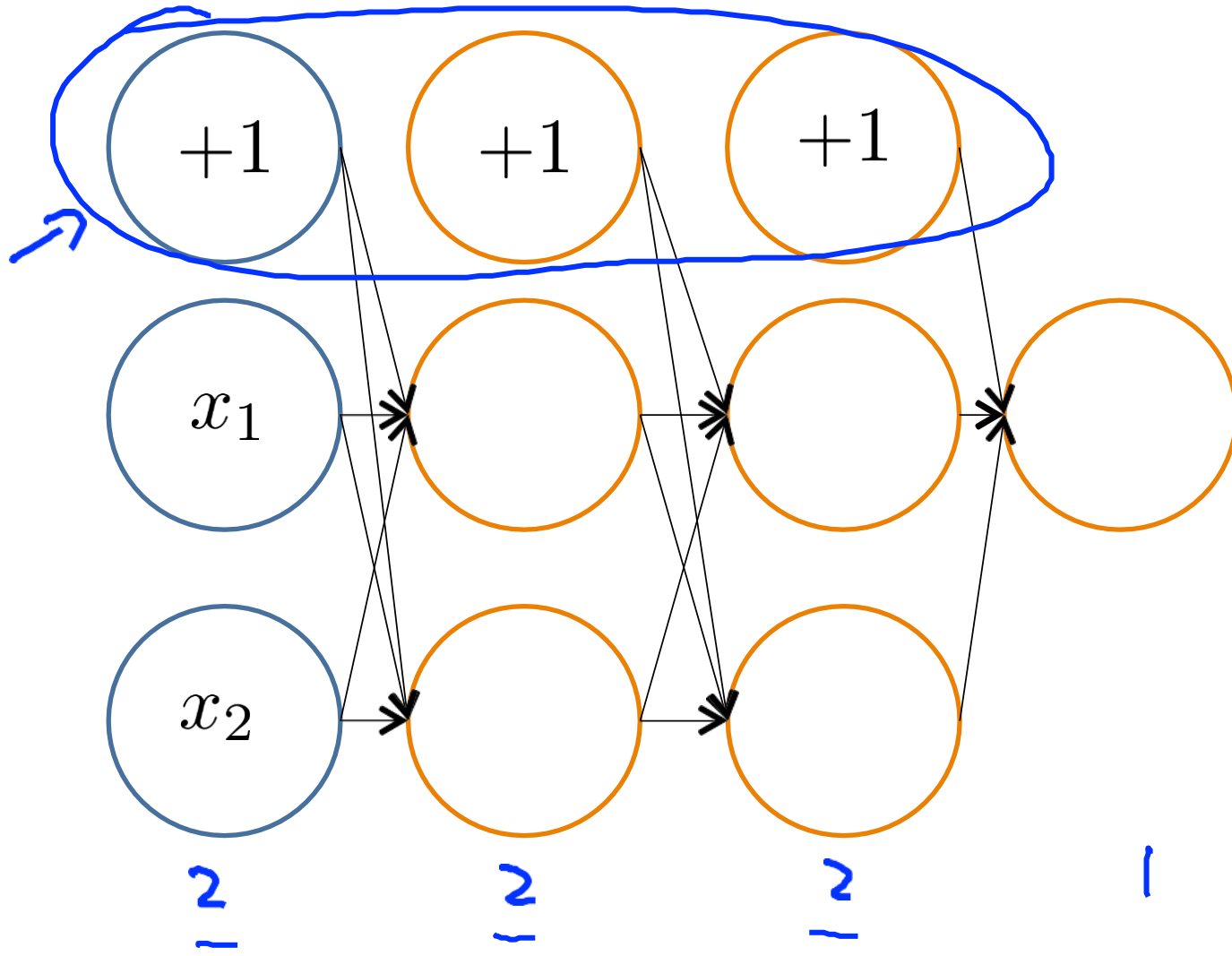
$$a_1^{(2)} = g\left(\sum_{j=0} \theta_{1j}^{(1)} a_j^{(1)}\right)$$

$$a_2^{(2)} = g\left(\sum_{j=0} \theta_{2j}^{(1)} a_j^{(1)}\right)$$

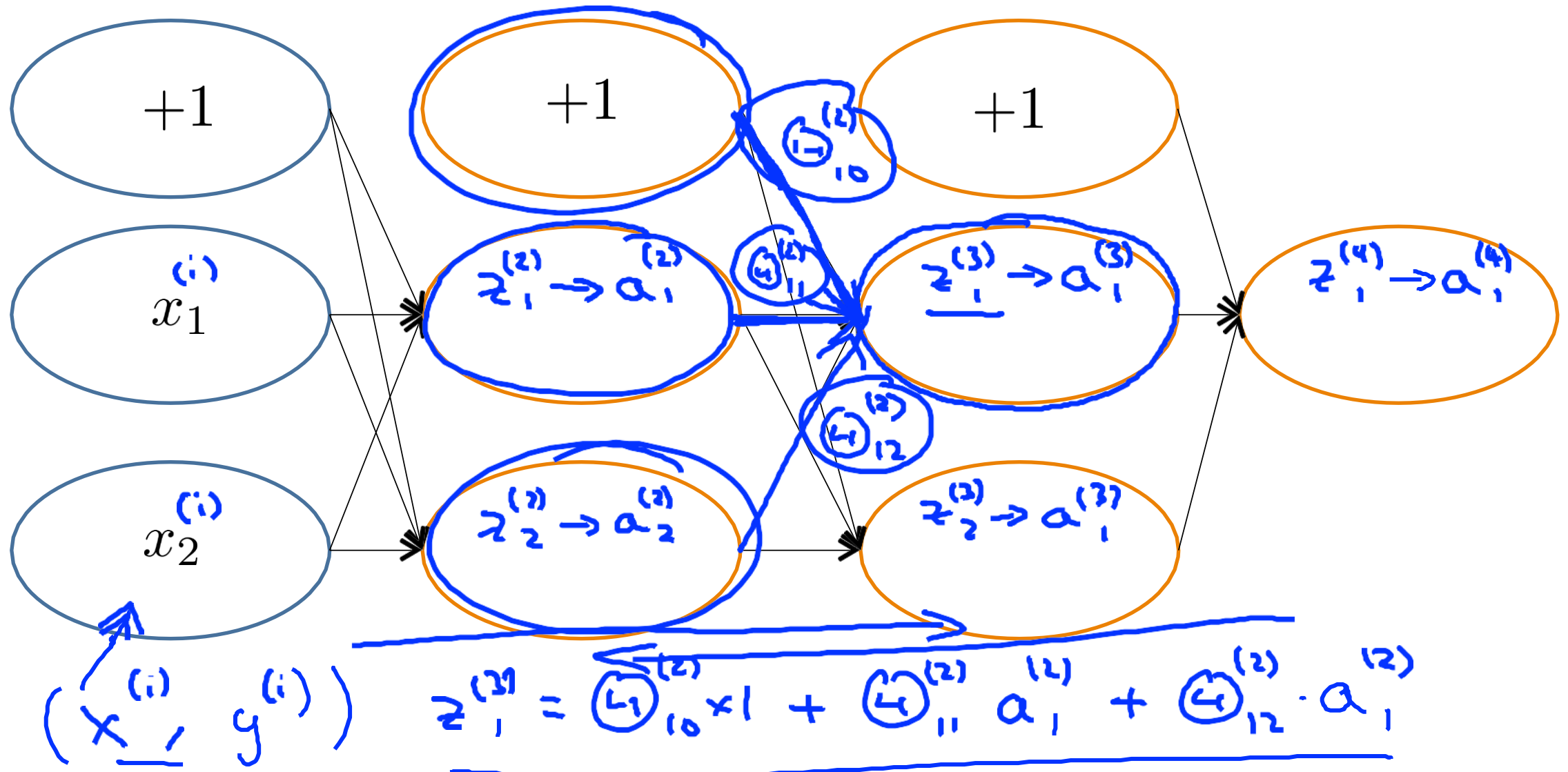
$$a_1^{(3)} = g\left(\sum_j \theta_{1j}^{(2)} a_j^{(2)}\right)$$

$$a_2^{(3)} = g\left(\sum_j \theta_{2j}^{(2)} a_j^{(2)}\right)$$

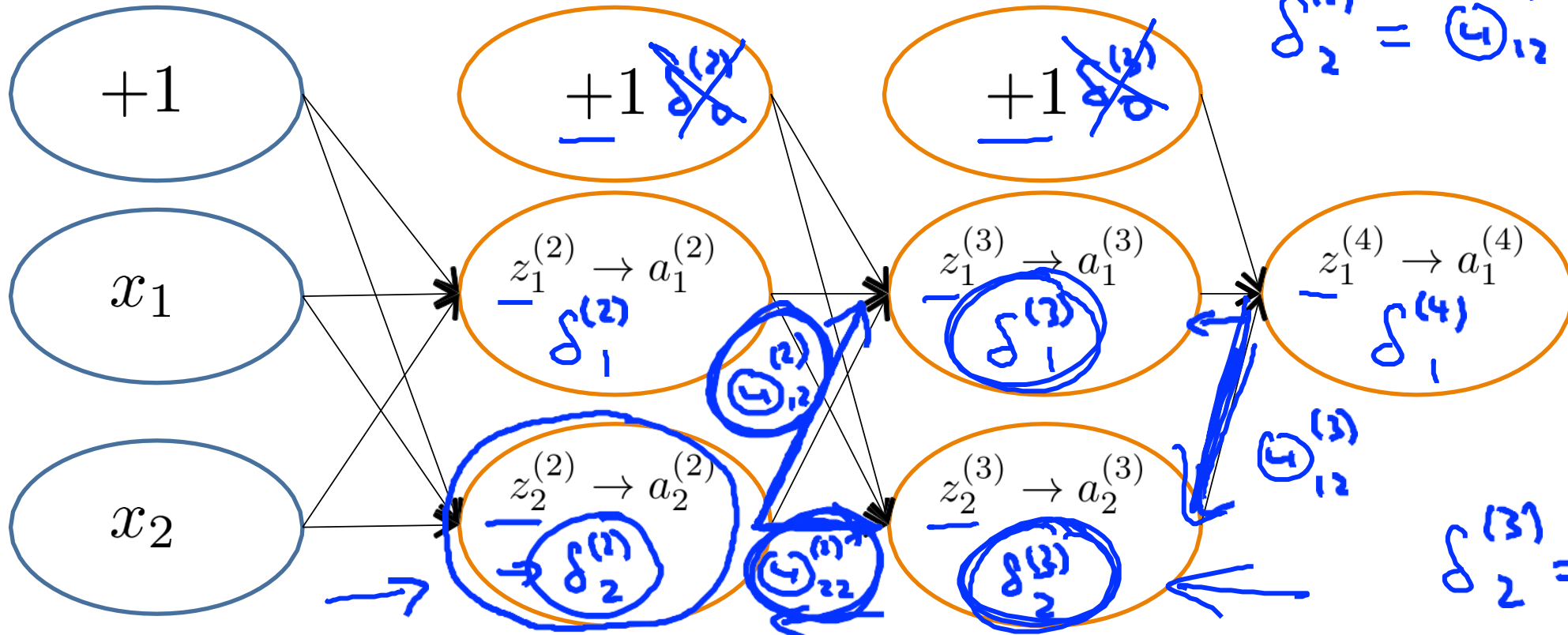
Forward Propagation Example



Forward Propagation



Forward Propagation



$$\delta_1^{(4)} = y^{(i)} - a_1^{(4)}$$

$$\delta_2^{(2)} = W_{12}^{(2)} \delta_1^{(3)} + W_{22}^{(2)} \delta_2^{(3)}$$

$$\delta_2^{(3)} = W_{12}^{(3)} \delta_1^{(4)}$$

→ $\delta_j^{(l)}$ = "error" of cost for $a_j^{(l)}$ (unit j in layer l).

Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(i)$ (for $j \geq 0$), where

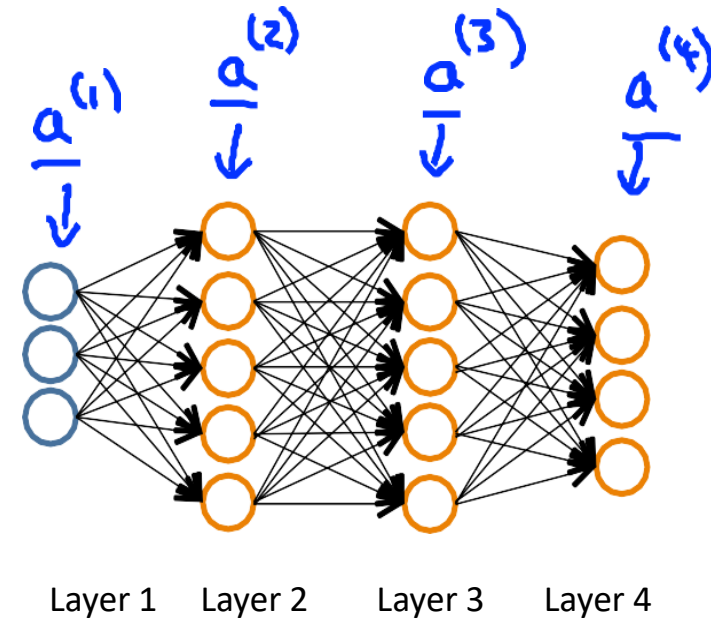
$$\text{cost}(i) = y^{(i)} \log h_{\Theta}(x^{(i)}) + (1 - y^{(i)}) \log h_{\Theta}(x^{(i)})$$

Gradient Computation

Given one training example (x, y) :

Forward Propagation

$$\begin{aligned} & \underline{a^{(1)}} = \underline{x} \\ \rightarrow & z^{(2)} = \Theta^{(1)} a^{(1)} \\ \rightarrow & a^{(2)} = g(z^{(2)}) \quad (\text{add } \underline{a_0^{(2)}}) \\ \rightarrow & z^{(3)} = \Theta^{(2)} a^{(2)} \\ \rightarrow & a^{(3)} = g(z^{(3)}) \quad (\text{add } \underline{a_0^{(3)}}) \\ \rightarrow & z^{(4)} = \Theta^{(3)} a^{(3)} \\ \rightarrow & \underline{a^{(4)}} = \underline{h_{\Theta}(x)} = g(z^{(4)}) \end{aligned}$$



Gradient Computation: Backpropagation Algorithm

Denote $\delta_j^{(l)}$ = "error" of node j in layer l

For each output unit (layer $L=4$)

$$\delta_j^{(4)} = a_j^{(4)} - y_j^{(4)}$$

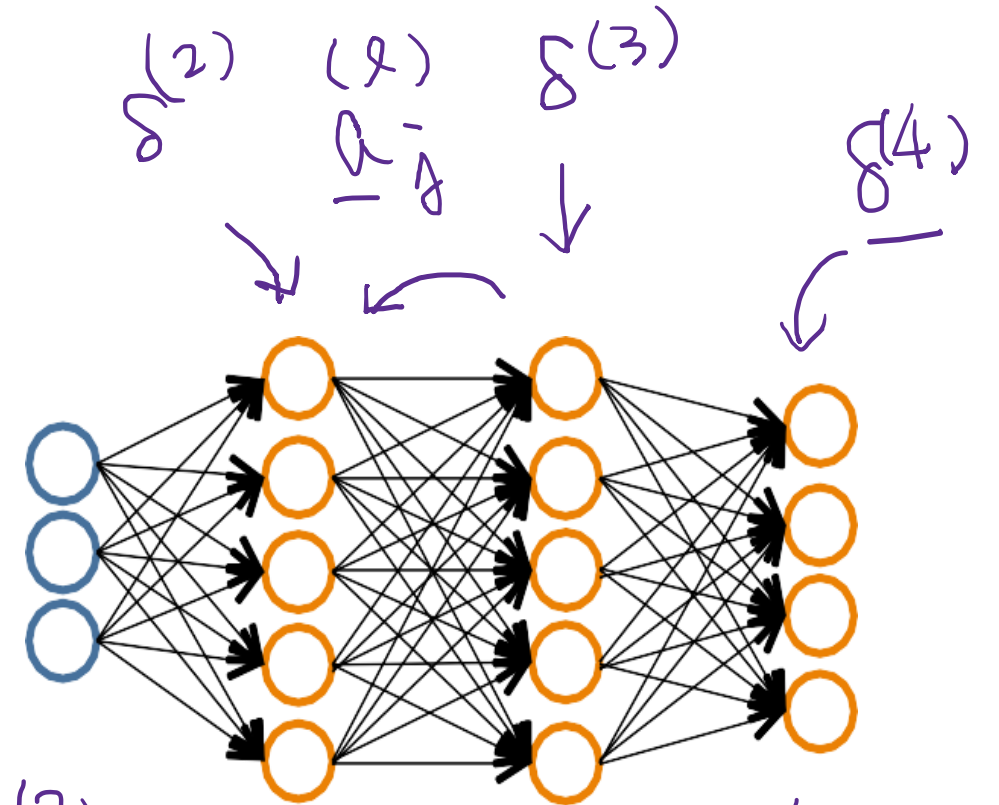
$$\delta^{(3)} = (\theta^{(3)})^T \delta^{(4)} * g'(z^{(3)})$$

$$\delta^{(2)} = (\theta^{(2)})^T \delta^{(3)} * g'(z^{(2)})$$

(No $\delta^{(1)}$)

$$\frac{\partial J(\theta)}{\partial \theta_{ij}^{(l)}} J(\theta) = a_j^{(l)} \delta_i^{(l+1)} \quad (\text{ignore } \lambda)$$

dot product



Backpropagation Algorithm (proof is complicated)

Training set $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j).

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $\underline{a^{(l)}}$ for $l = \underline{2}, \underline{3}, \dots, \underline{L}$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \underline{\delta^{(2)}}$

$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

$$= \Delta_{ij}^{(l)} + \delta_i^{(l+1)} (a_j^{(l)})^T.$$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \underline{\Theta_{ij}^{(l)}} \text{ if } \underline{j \neq 0}$$
$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \text{ if } \underline{j = 0}$$

$$\underline{\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}}$$

Backpropagation Algorithm Proof (use chain rule)

Handwritten derivation of the backpropagation algorithm for a sigmoid neuron. The derivation shows the gradient of the loss with respect to the weights, using the chain rule.

$$\frac{\partial J}{\partial w^{(2)}} = - \left[y^{(i)} \frac{\partial}{\partial w^{(2)}} \log \sigma(w^{(2)} a + b^{(2)}) + (1 - y^{(i)}) \frac{\partial}{\partial w^{(2)}} \left(\log (1 - \sigma(w^{(2)} a + b^{(2)})) \right) \right]$$

$$= - \left[y^{(i)} \frac{1}{\sigma^{(2)}} \cdot a^{(2)} \cdot (1 - a^{(2)}) \cdot a^{(2)T} + (1 - y^{(i)}) \frac{1}{1 - \sigma^{(2)}} \cdot (-1) \cdot a^{(2)} \cdot (1 - a^{(2)}) \cdot a^{(2)T} \right]$$

$$= - \left[y^{(i)} (1 - a^{(2)}) a^{(2)T} - (1 - y^{(i)}) a^{(2)} \cdot a^{(2)T} \right]$$

$$= - \left[y^{(i)} a^{(2)T} - a^{(2)} a^{(2)T} \right] = (y^{(i)} - a^{(2)}) a^{(2)T}$$

Additional notes on the right side of the whiteboard:

$$\frac{\partial J}{\partial w^{(2)}} = - \frac{1}{m} \sum_{i=1}^m (y^{(i)} - a^{(2)}) a^{(2)T}$$

$$\frac{\partial J}{\partial w^{(2)}} = \frac{\partial J}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w^{(2)}}$$

The chain rule is applied to show that the gradient of the loss with respect to the weights is the product of the error term and the input vector.

<https://youtu.be/zUazLXZZA2U> (Stanford Course 2018 in details)

<https://youtu.be/yXcQ4B-YSjQ> (Andrew Ng short video)

Technical Details

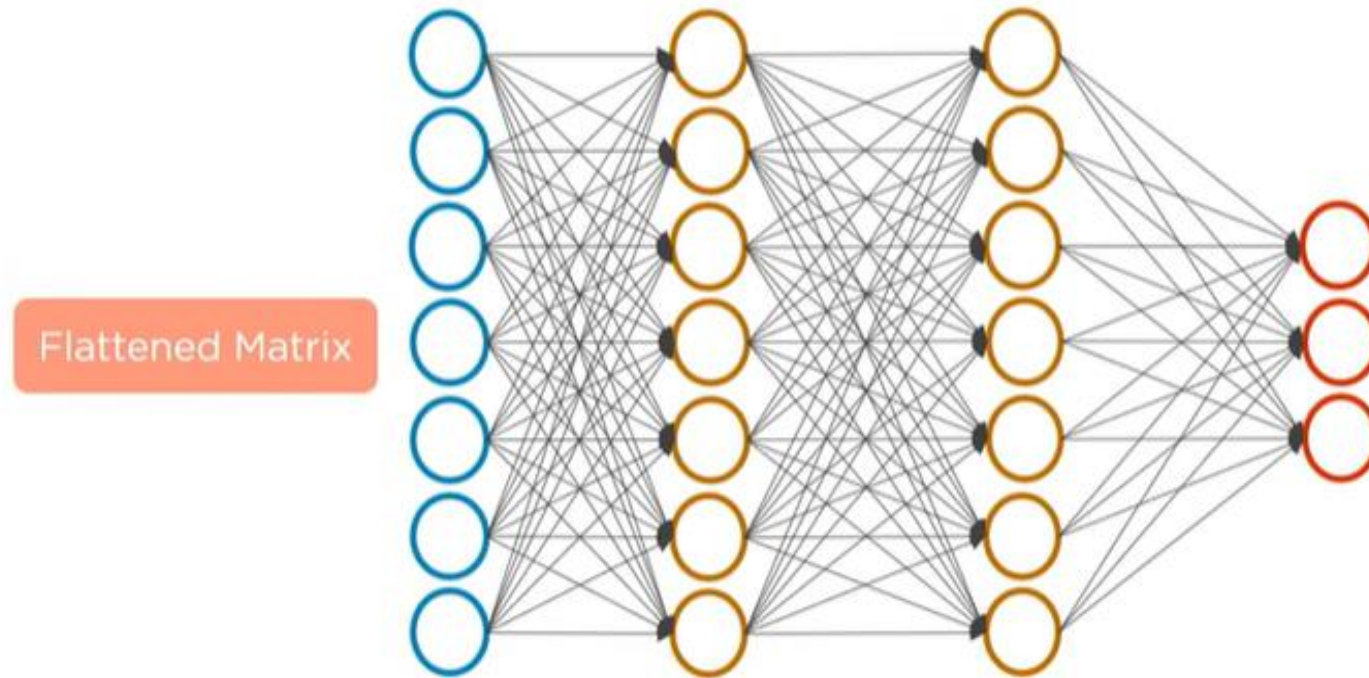
Good News:

All these calculations have been implemented in various deep learning library

Types of Neural Network Architecture

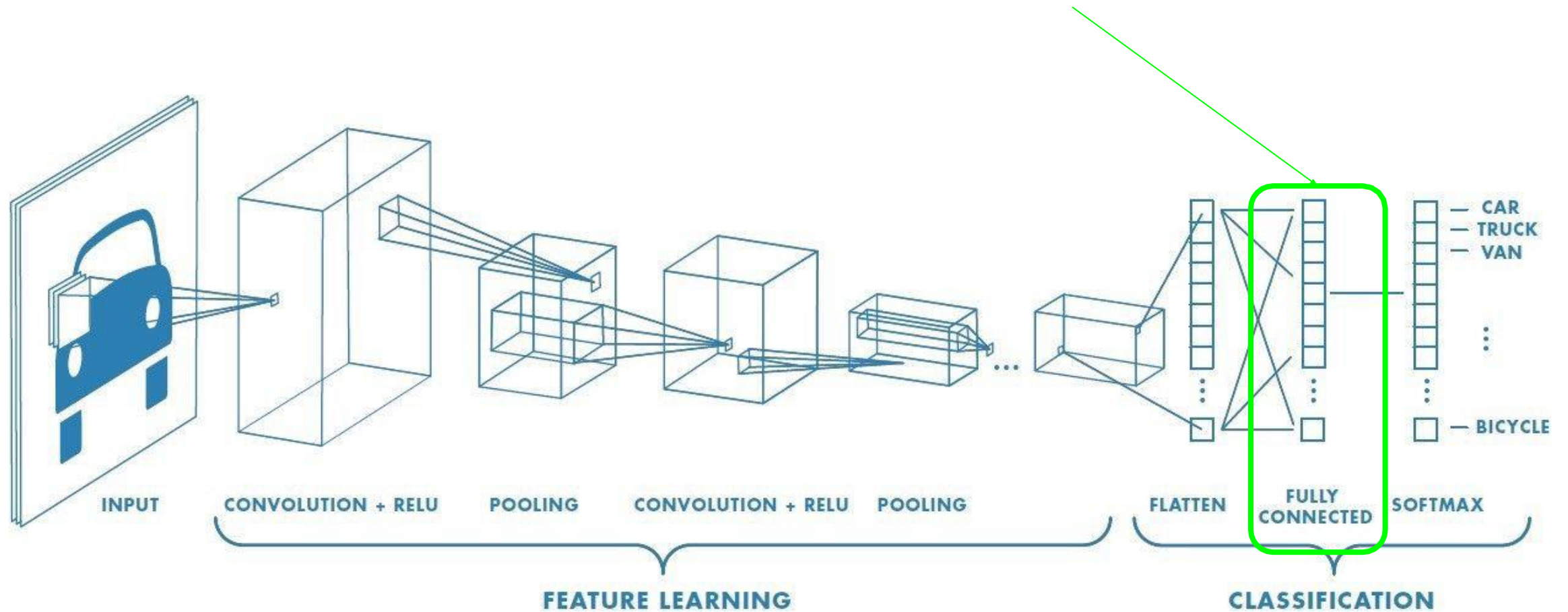
- Multi-layer Perceptions (MLP)
- Convolution Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Transformers

Multilayer Perceptions



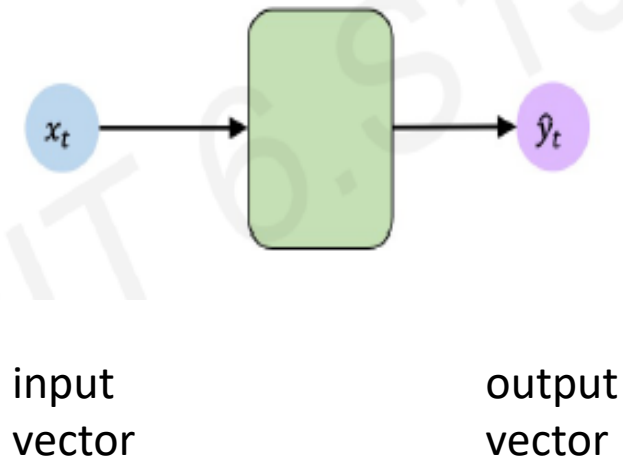
Go to hand-written digits as the “Hello World” DL example

Convolution Neural Network

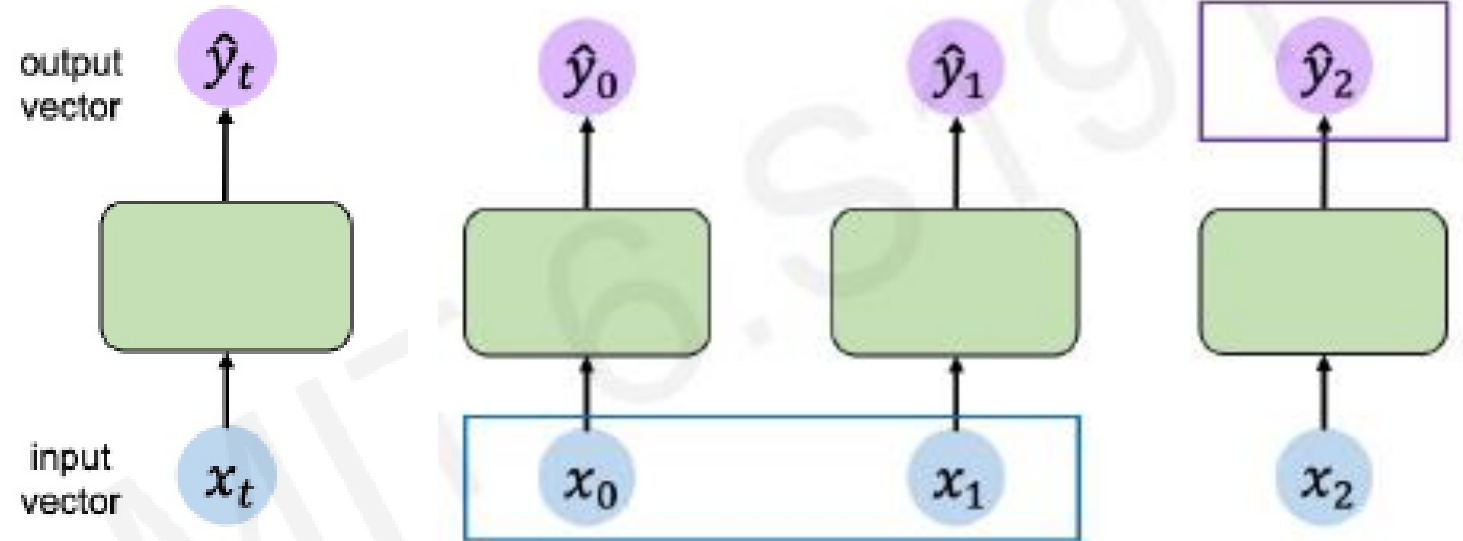


Go to weather or cats versus dogs classification example

Recurrent Neural Network (for time-series data/sequential data)



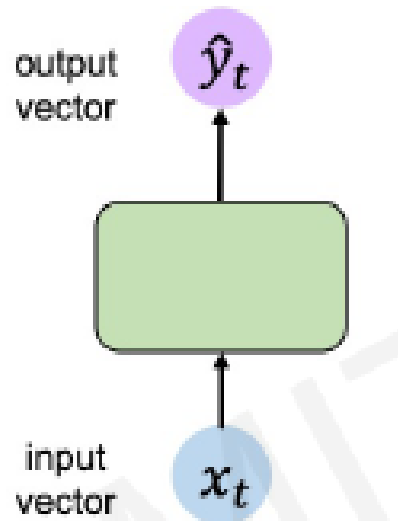
$$\hat{y}_t = f(x_t)$$



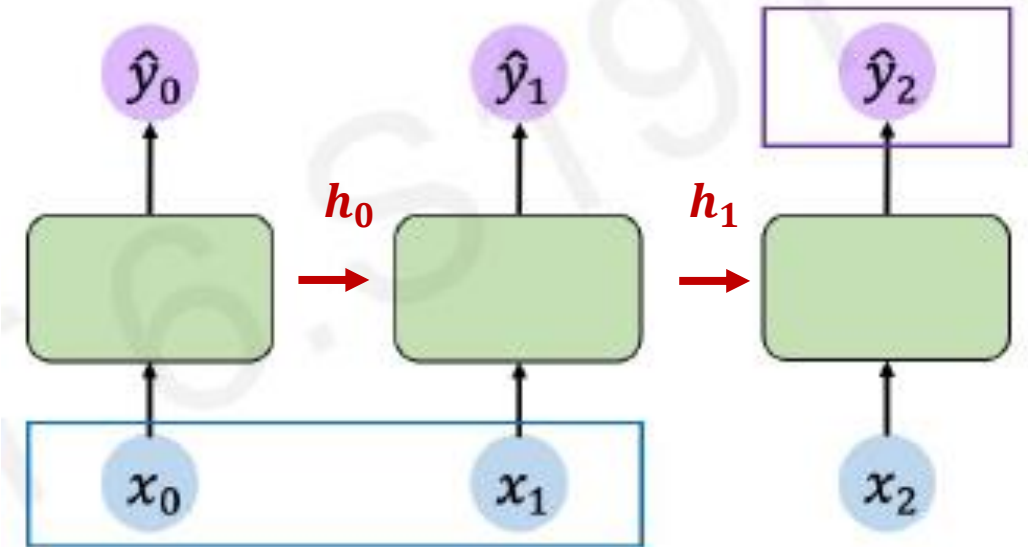
$$\hat{y}_t = f(x_t)$$

Same diagram, but rotated 90 degree
So that we can play out the time component horizontally

Feed-Forward Networks Abstracted out



$$y_t = f(x_t)$$

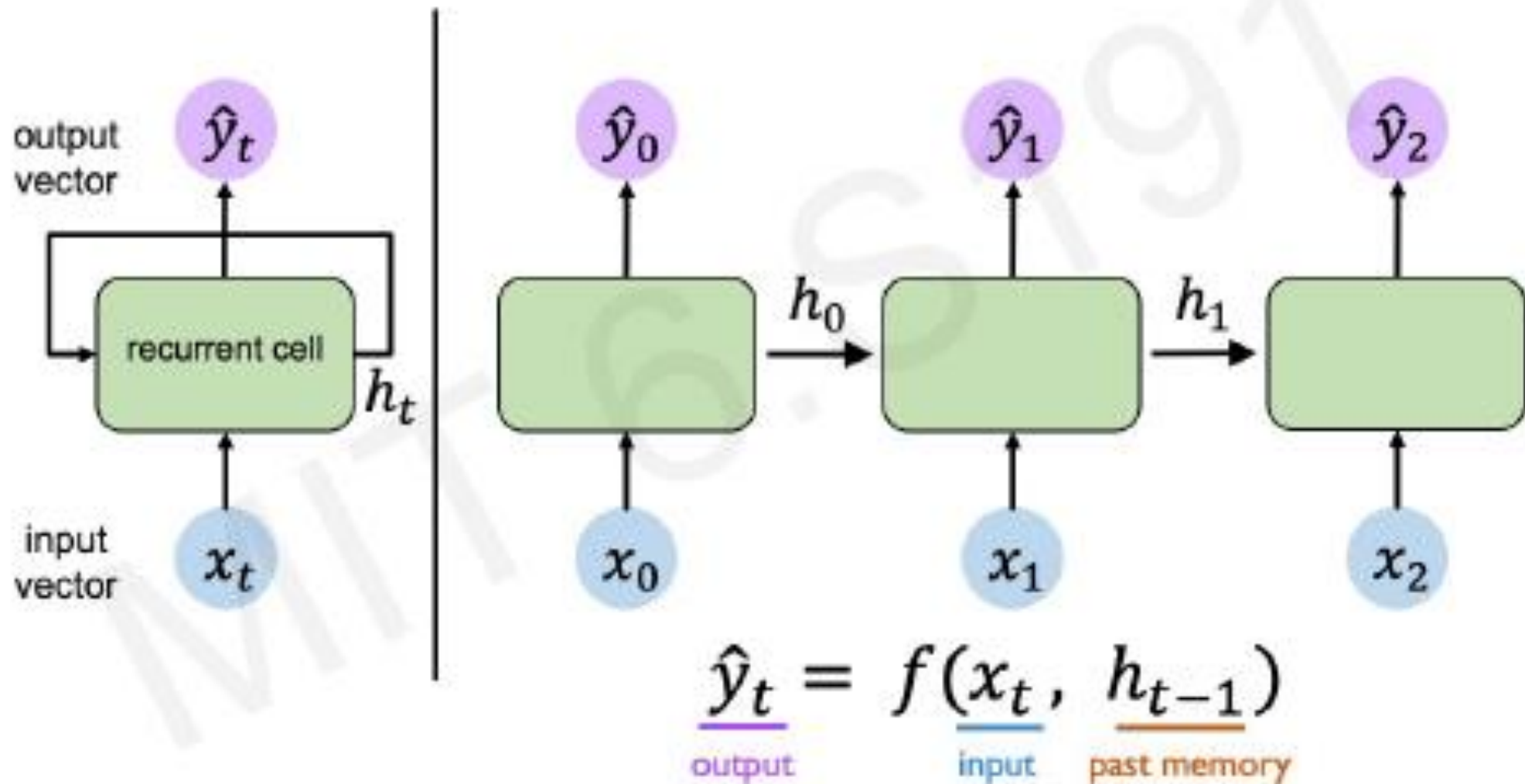


Introduce concepts of a state h_i for capturing memory from the past

$$y_t = f(x_t, h_{t-1})$$

output input past memory

Neurons with Recurrence

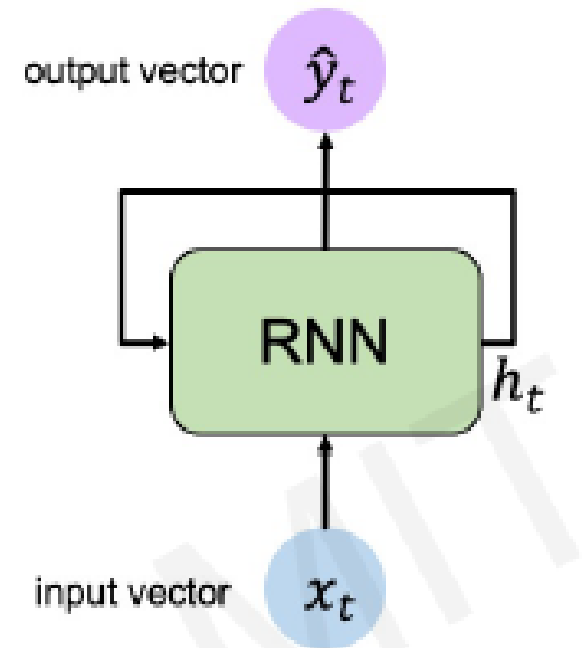


Recurrent Neural Networks (RNNs)

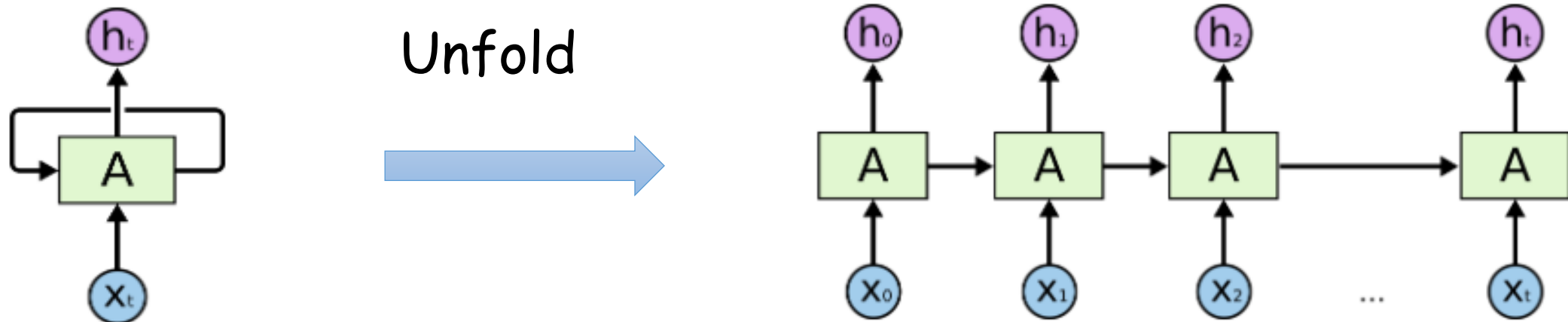
We can process a sequence of vectors \mathbf{x} by applying a **recurrence formula** at every time step:

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state some function with parameters W old state input vector at some time step



Recurrent Neural Networks (RNNs)



$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$

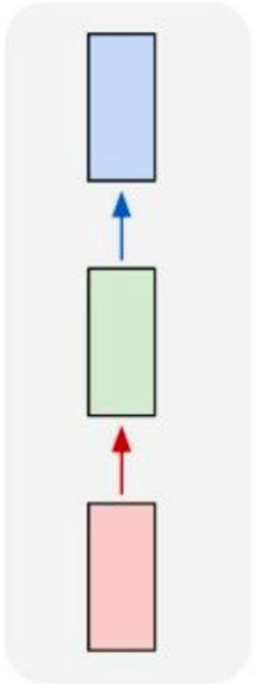
$$y_t = \sigma_y(W_y h_t + b_y)$$

Note: it requires 3 weights matrix for each RNN unit

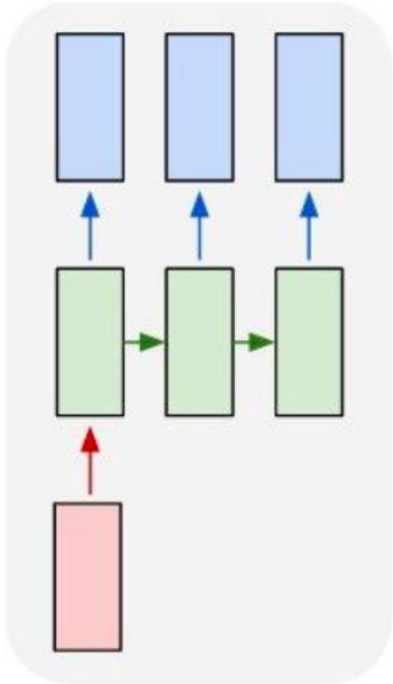
[Understanding LSTM Networks -- colah's blog](#)

Recurrent Neural Networks: Application examples

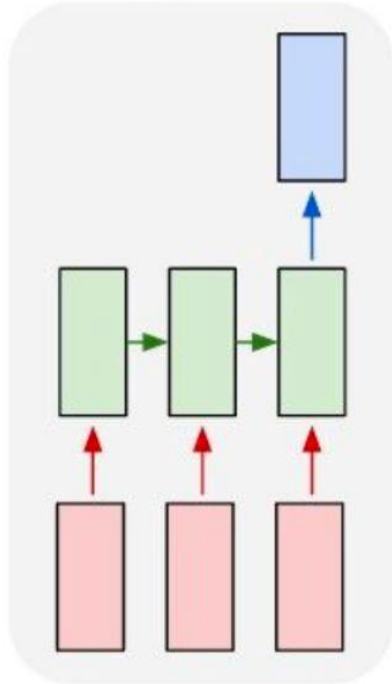
one to one



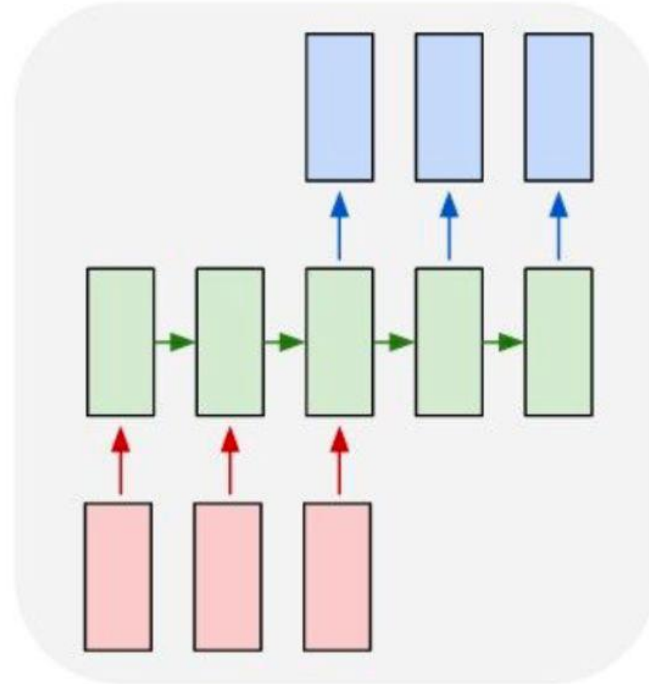
one to many



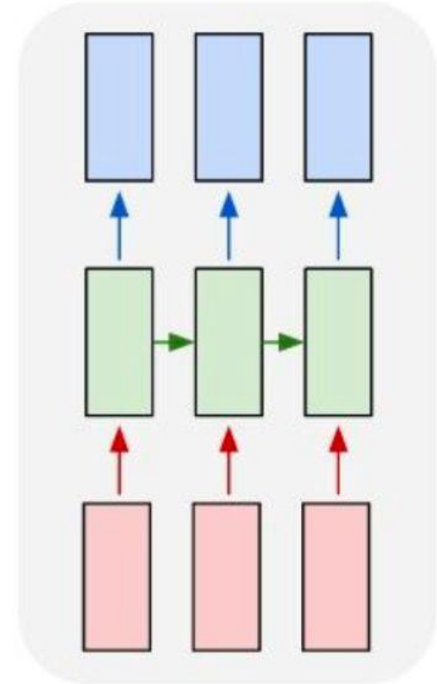
many to one



many to many



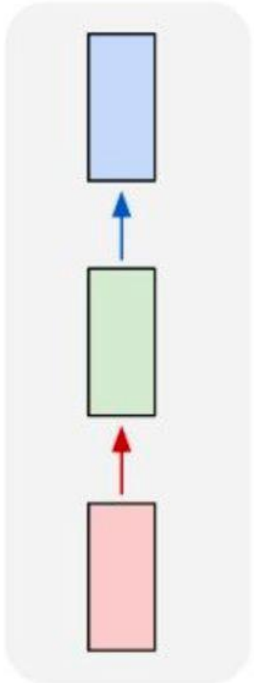
many to many



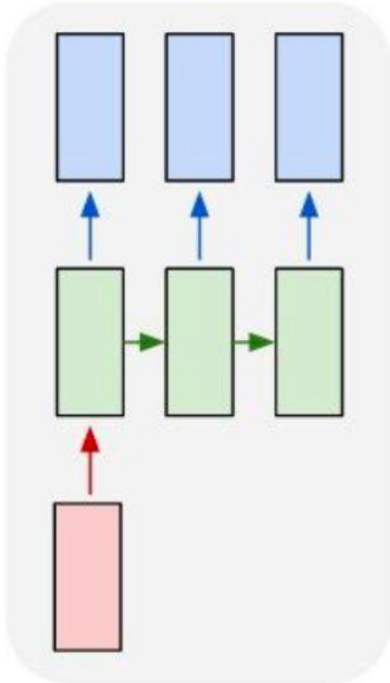
e.g. Image Captioning
image -> sequence of words

Recurrent Neural Networks: Process Sequences

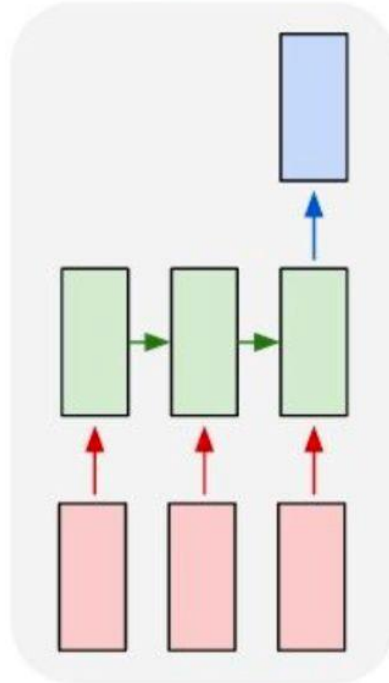
one to one



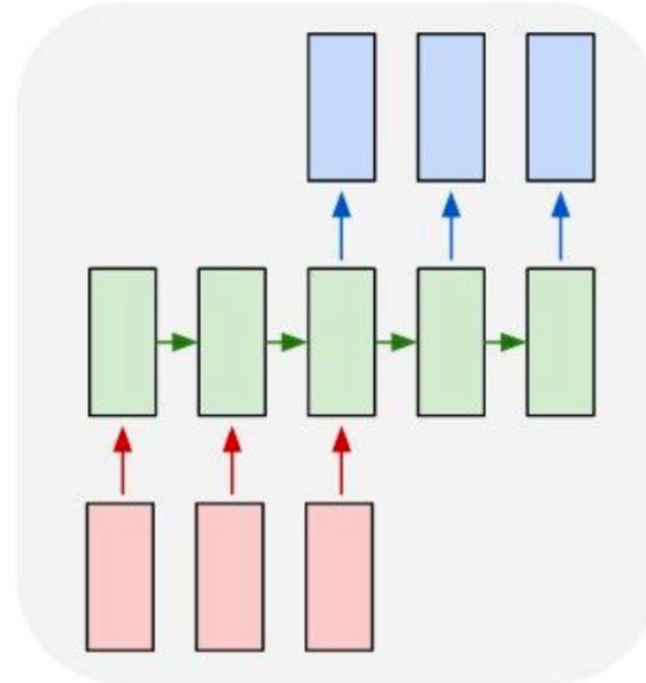
one to many



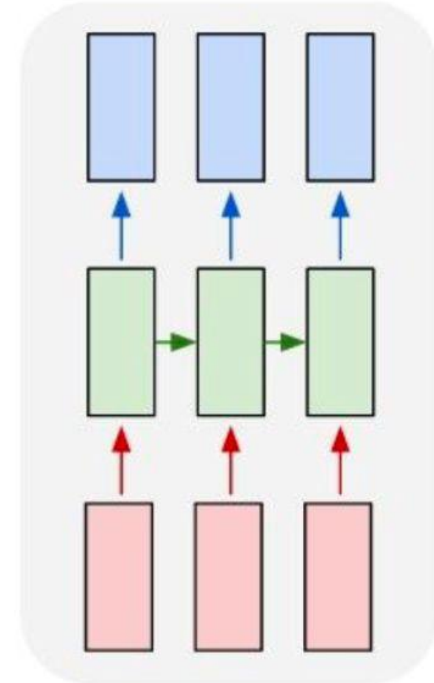
many to one



many to many



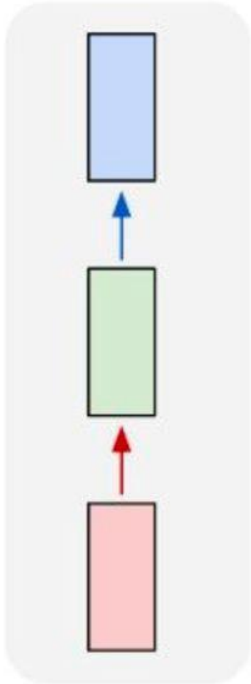
many to many



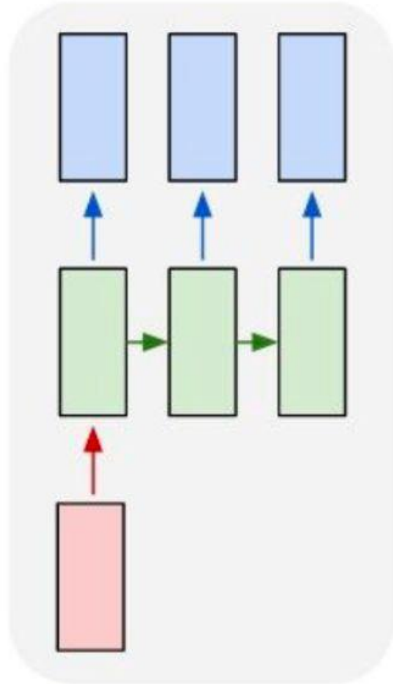
e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Neural Networks: Process Sequences

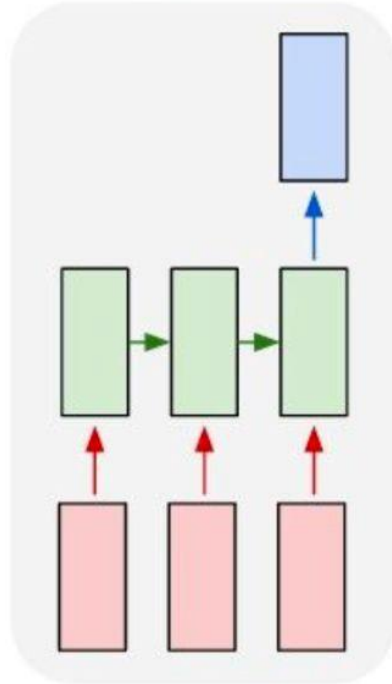
one to one



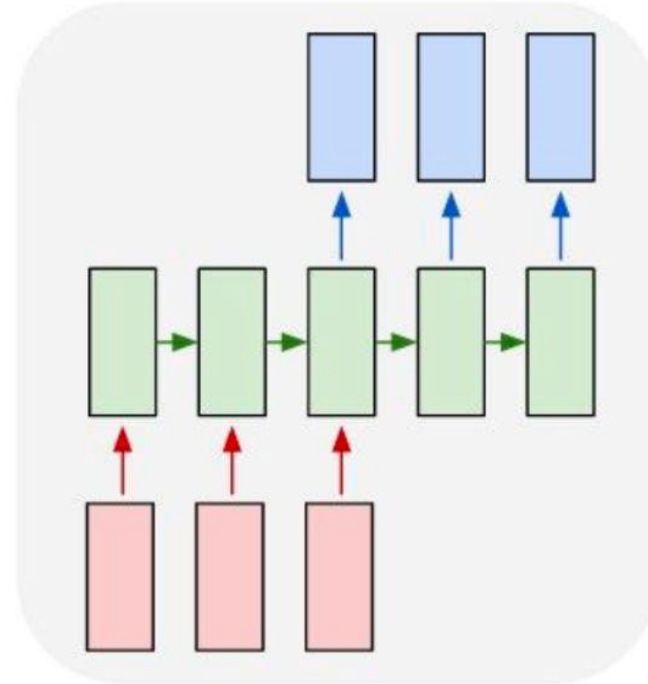
one to many



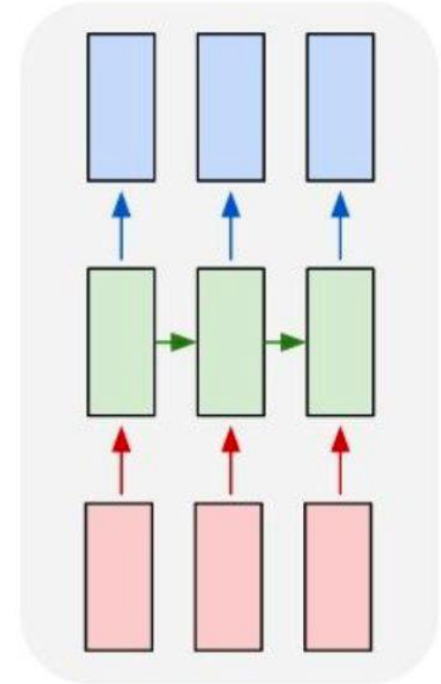
many to one



many to many



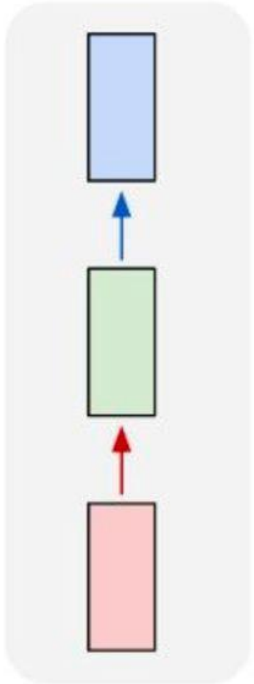
many to many



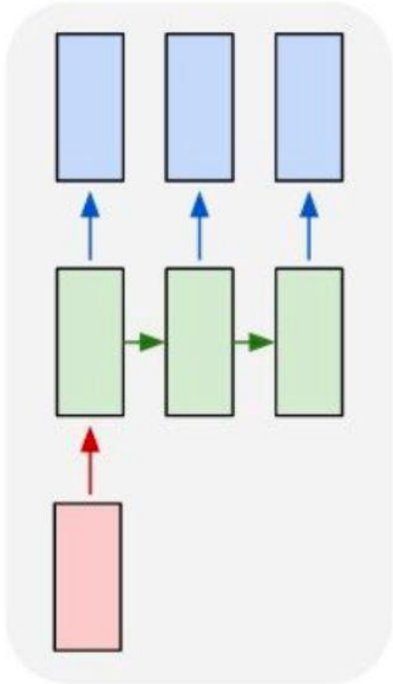
e.g. **Machine Translation**
seq of words -> seq of words

Recurrent Neural Networks: Process Sequences

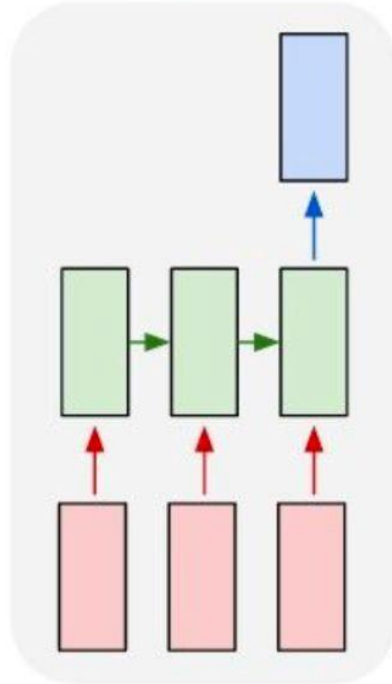
one to one



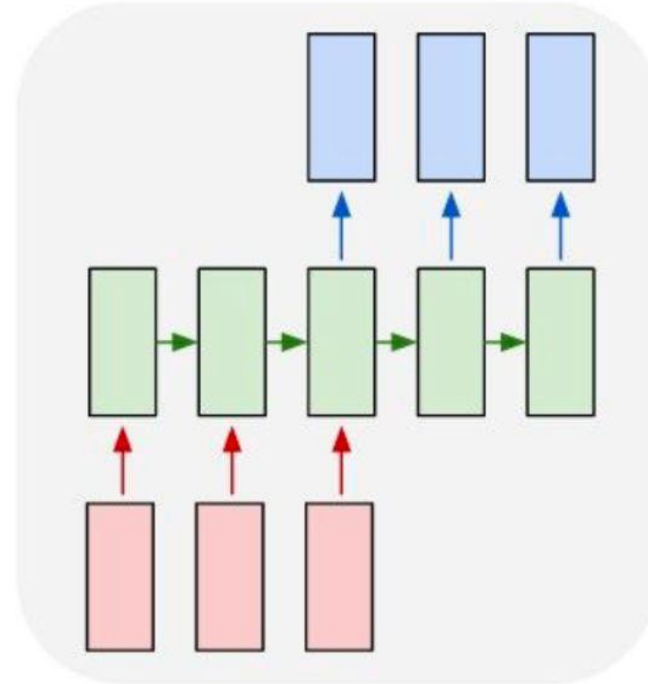
one to many



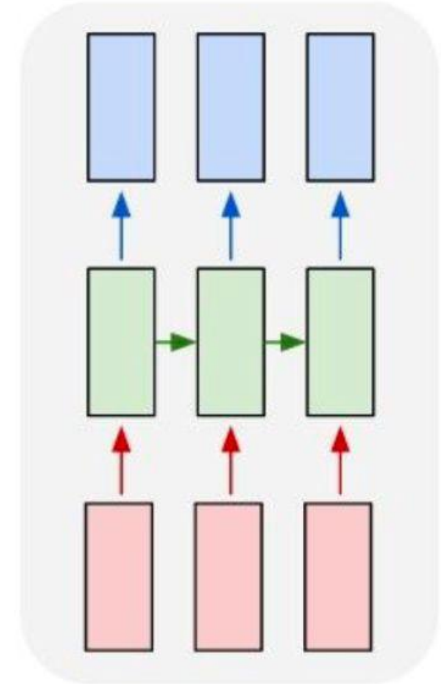
many to one



many to many



many to many



e.g. **Video classification on frame level**

Problem with RNN

- Not easy to parallelize as RNN process sequence sequentially
- Cannot handle long memory
 - Vanishing gradient problem (sensitivity to distant past is very hard to estimate)
- Recent Breakthrough is based on a paper on 2017 called
 - "Attention is all you needed" by Google Researchers
 - which leads to a new Neural Network Architecture called Transformers
- First application is focus on Natural Language Processing, but later on apply to other "Generative Ai" areas.

The rest is history

Outline

- Overview
- Classical Machine Learning
- Deep Learning
- **Generative AI**
- List of Resources

What is Generative AI

- It got started in Natural Language Process
 - Machine Translation
 - chatbot
- Classification of Images
- Text to images
- Now expand to everywhere, virtual assistant, Github Copilot

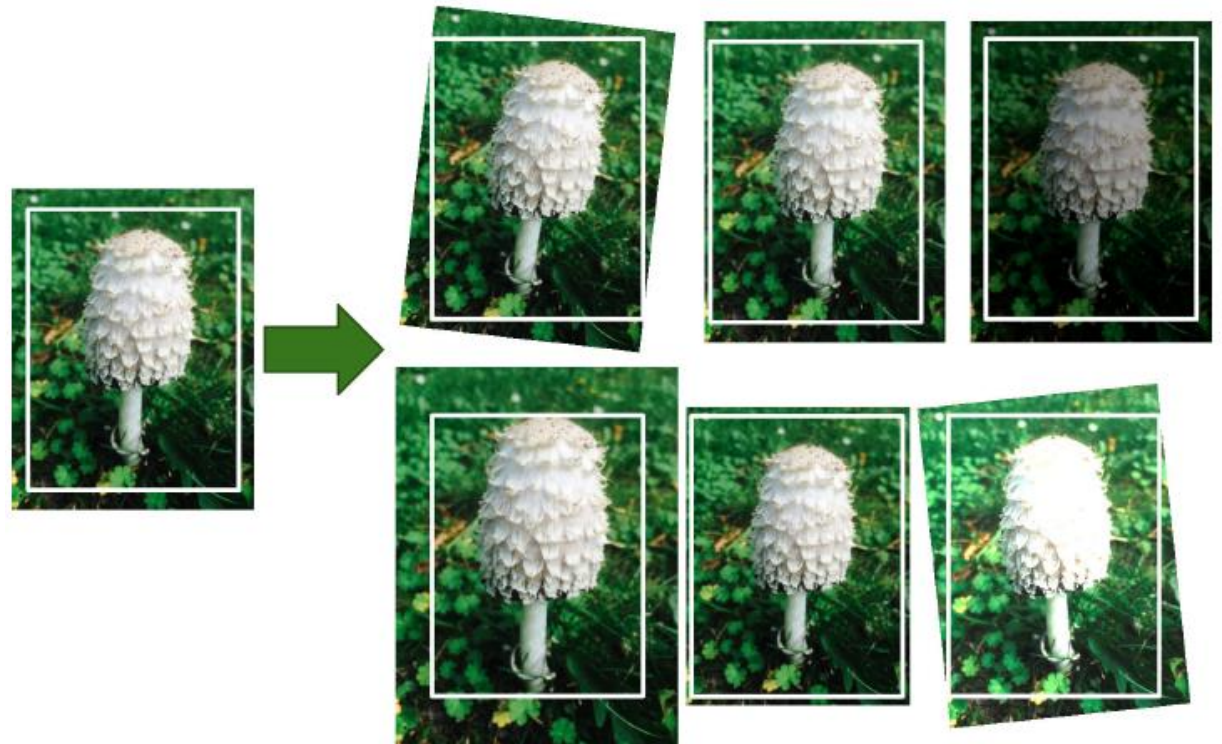
How it got started? (other than NLP)

- If you want to model un-common events, you may not have enough data
- Data Augmentation to avoid overfitting and make your model more robust

Example: you can generate new images by slightly shift, rotate, and resize every picture in the training set by various amounts.

This forces the model to be more tolerant to variations in the position, orientation, and size of the objects in the pictures.

You can also generate images with various contrasts, or flipping the pictures horizontally



Generative AI

Traditional Deep Learning

Use lots of dogs and cats' images to train a neural network. Once trained, use the trained model to

- label a new given image whether it is a dog or a cat

Generative AI

Use limited number of dogs, cats and horse images to train a neural network. Once trained, use the trained model to

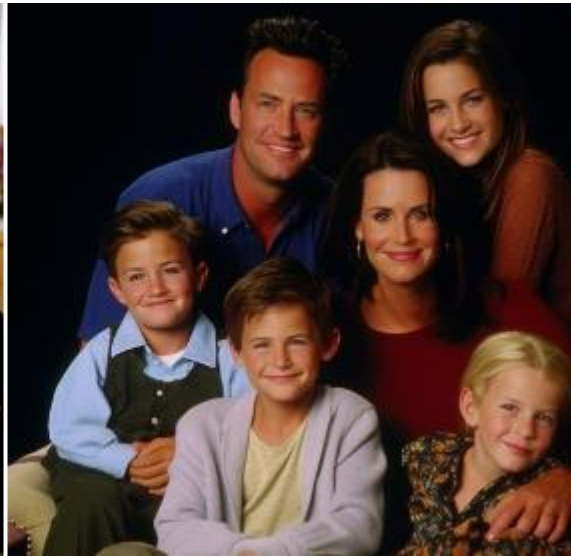
- generate new artificial dog and cat's pictures
- generate a picture of Zebra

Generative AI Show Cases

- We can have Obama teaching Quantum Field Theory

[\(1217\) Barack Obama: Intro to Deep Learning | MIT 6.S191 - YouTube](#)

- Any Friends Fans here?



Or Kim Kardashian's fan



How does chatGPT work

On given a list of words, predict the next word by picking the highest probable word from all vocabulary, then repeat

"France is where I grew up, but I now live in Boston. I speak fluent French"

Given these words

Predict the next word

"The students opened their book"

Given these words

Predict the next word

"The students opened their laptops"

Given these words

Predict the next word

Self-Attention (technical details)

RNN fail because it poorly capture

- Context (same word can mean different things)
- Words from distant past could be significant
- Very slow to feed in one word at a time, re-calculate the internal state

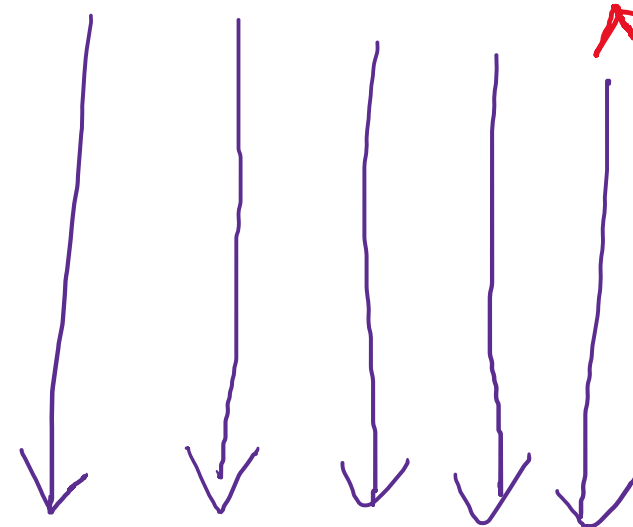
On the other hand, self-attention

- Capture the idea of one has to pay particular attention to certain words even it is in the distant past
- Consider the whole sentence together
- Can be parallelized

Intuition of Attention

more relevant

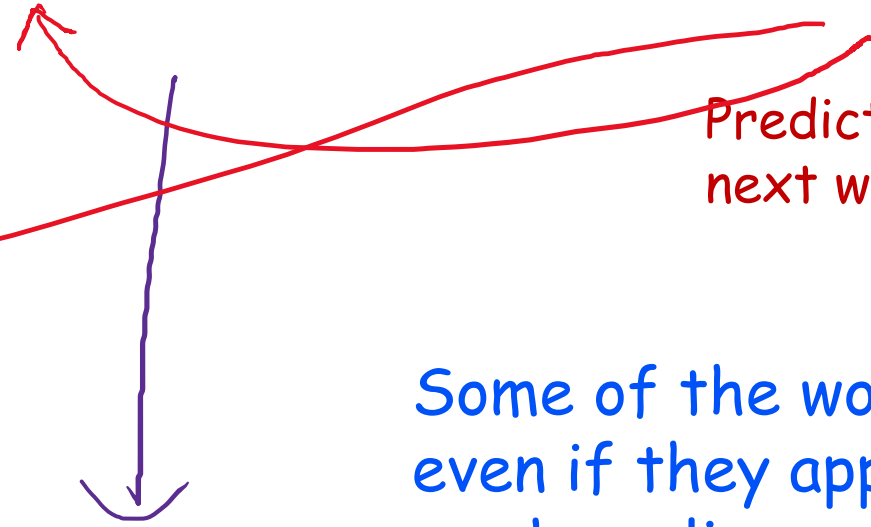
"France is where I grew up, but I now live in Boston. I speak fluent French"



t_1 t_2 t_3

Given these words

less relevant



Predict the next word

t_{12}

Some of the words even if they appear much earlier may require more attention

Intuition of Attention



When the model process the word "it", the system should give more weights to the word "cat"



When the model process the word "it", the system should give more weights to the word "milk"

Transformers

The paper chatGPT is based on

Use Attention heavily

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaizer@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. **We propose a new simple network architecture, the Transformer,** based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

Transformers

[All you need to know about 'Attention' and 'Transformers' — In-depth Understanding — Part 2 | by Arjun Sarkar | Towards Data Science](#)

More details in the appendix
Won't be in the exam

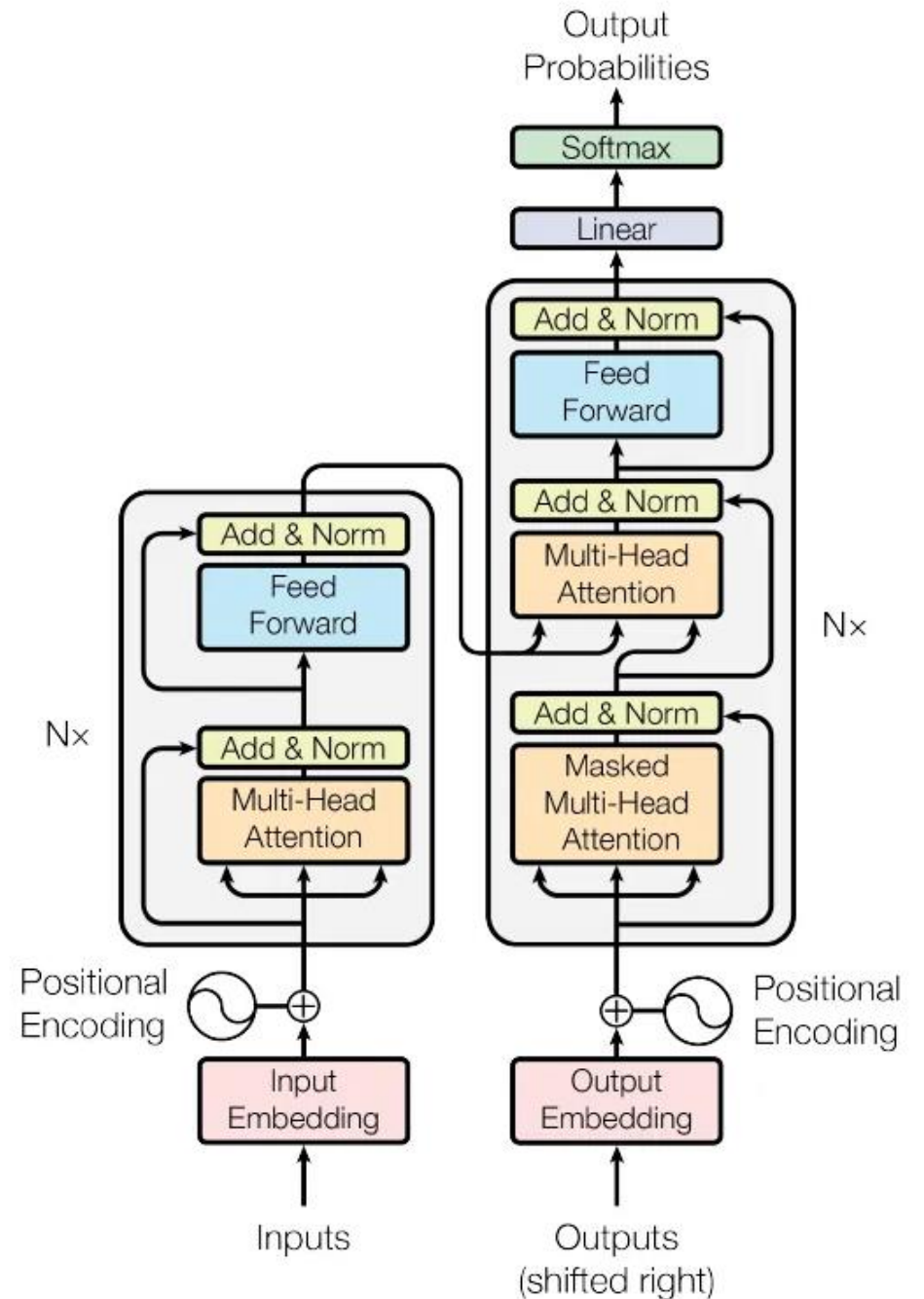


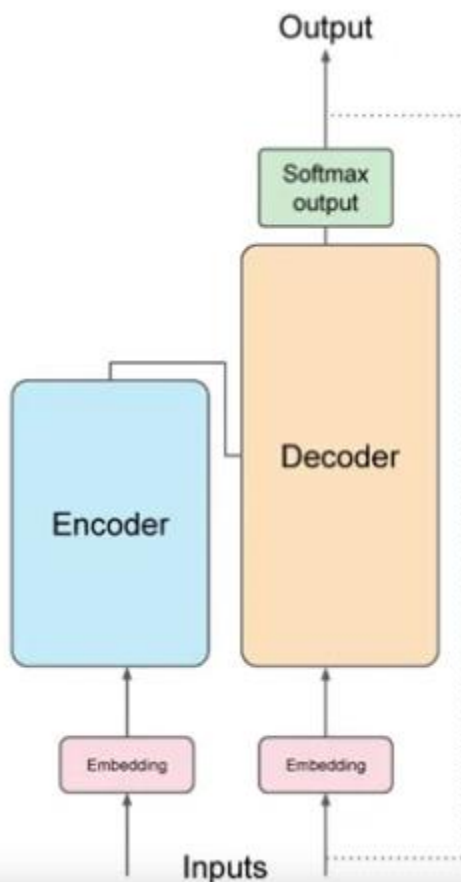
Figure 1. The Transformer Network (Source: Image from the original paper)

Transformers

Transformers

Encoder

Encodes inputs (“prompts”) with contextual understanding and produces one vector per input token.



Decoder

Accepts input tokens and generates new tokens.

Prompt Engineering

- We used to interact with a software library using API (application programming interface)
- Now, we use human language as input to get output from an LLM.
- LLM answer can be different in the second time when you ask the same question
- Prompt Engineering is the ideas/skills/strategy on how to construct your prompt in order to get the answers you want

LangChain - Chaining LLM and other tools together

- LLM by itself only serves as advanced google search
- LLM is a pre-trained general model
- Need action and models that are specialized at a certain field (such as medical)
- Concepts of Chaining models and agents/tools together

Outline

- Overview
- Classical Machine Learning
- Deep Learning
- Generative AI
- List of Resources

Resources for ML

- <https://www.datacamp.com/> (best introductory learning site)
- [Towards Data Science](#) (best intermediate learning site)
- [Statisticslectures.com](https://www.statisticslectures.com) (best short and sweets review on statistics)
- [StatQuest with Josh Starmer](#) (just enjoyable)
- [3Blue1Brown YouTube](#) (if you are not aware of his youtube channel, you don't know what you are missing in life)

- [Kaggle](#) (Data Science Competitions)
- [Kdnuggets](#) (Data Science Community)
- [Analytics Vidhya | Learn everything about Data Science](#) (Data Science Community)
- <https://machinelearningmastery.com/> (Great tutorials)

- [How I would learn Machine Learning if I could start again \(assembly ai\)](#) (Good starting point)
- [Andrew Ng Machine Learning Course at Stanford](#) (Andrew Ng has classic ML course)
- [MIT Deep Learning Course \(2023\)](#) (Recent best Deep Learning Lectures)
- [ChatGPT Prompt Engineering for Developers - DeepLearning.AI](#) (Prompt engineering)
- [Generative AI with Large Language Models | Coursera](#) (Latest Generative AI course)
- [fchollet/deep-learning-with-python-notebooks:](#) (Deep Learning with Python by Francois Chollet)

The End

Thank you

Relations to Physics

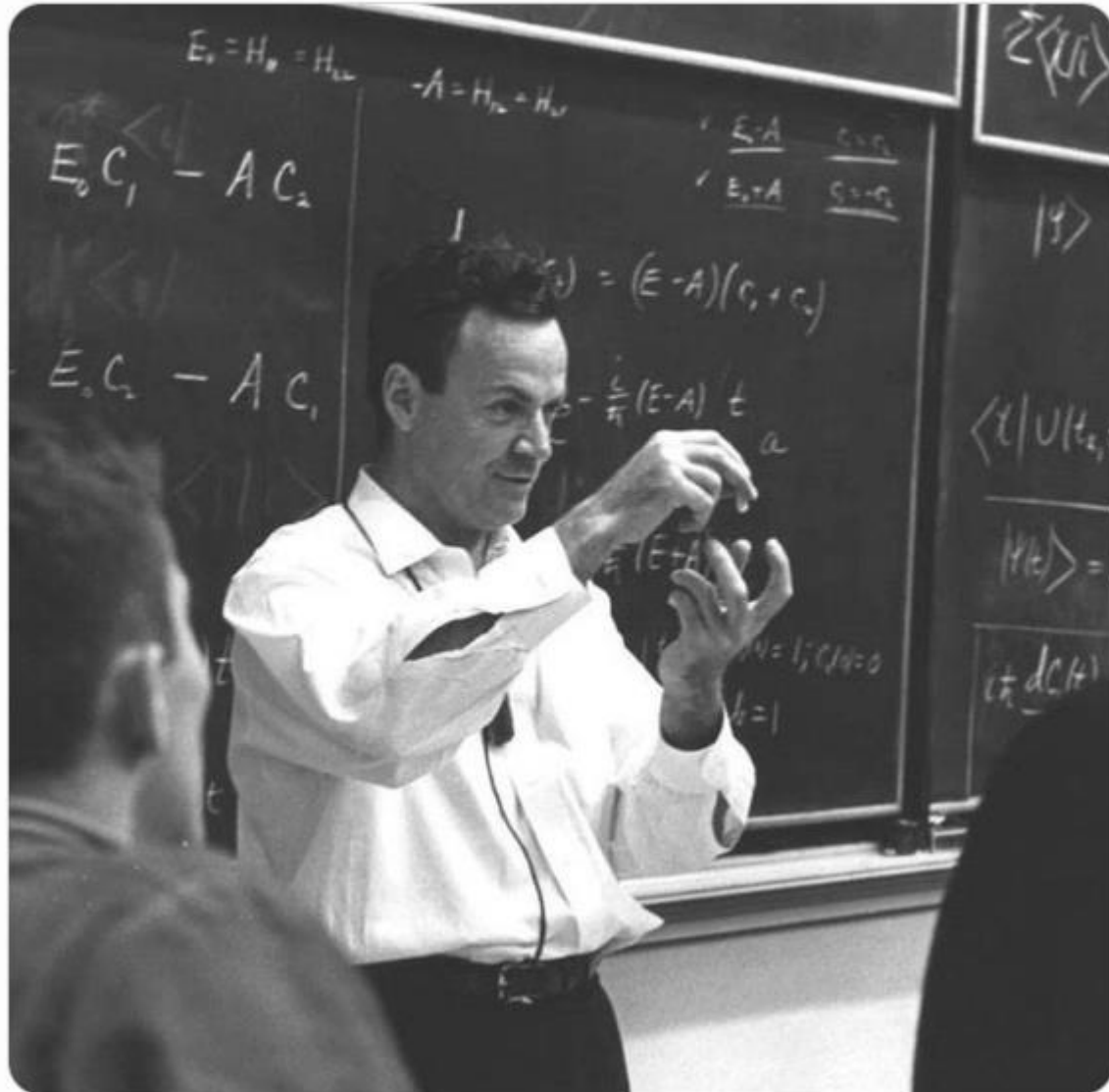
- Big Question: Does ChatGPT has intelligence (as in AGI)?
 - If you ask older generation of statisticians, most will say having 70B model parameters is crazy. It just overfits. No way it can generalize well to unseen cases. But somehow it works which is the real surprise
- Deep Learning is a black box
 - From data to output, no intermediate "theory" or constraints
- Intelligence shown in Physics is the greatest achievement in humanity
 - Special Relativity - Lorentz Transformation is known before Einstein who revolutionized by the concepts of Spacetime
 - Design a theory based on symmetry, prediction follows from geometry
 - A machine as of now does not have that kind of reasoning mechanism



Prof. Feynman  @ProfFeynman · Jun 2

If you cannot explain something in simple terms, you don't understand it.

...



818

10.6K

61.8K

5.1M



No one can really explain how and why deep neural network works

It just works

We don't even know when we can trust the ML/AI output.

Trustworthy AI is a research topics

There are also research now trying to understand the math behind NN etc

AI can help Physicists and vice versa

Machine versus Physicists

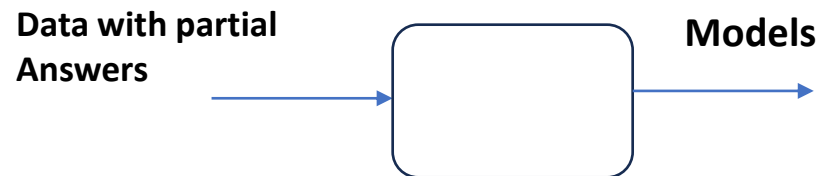


Classical Programming



Machine Learning

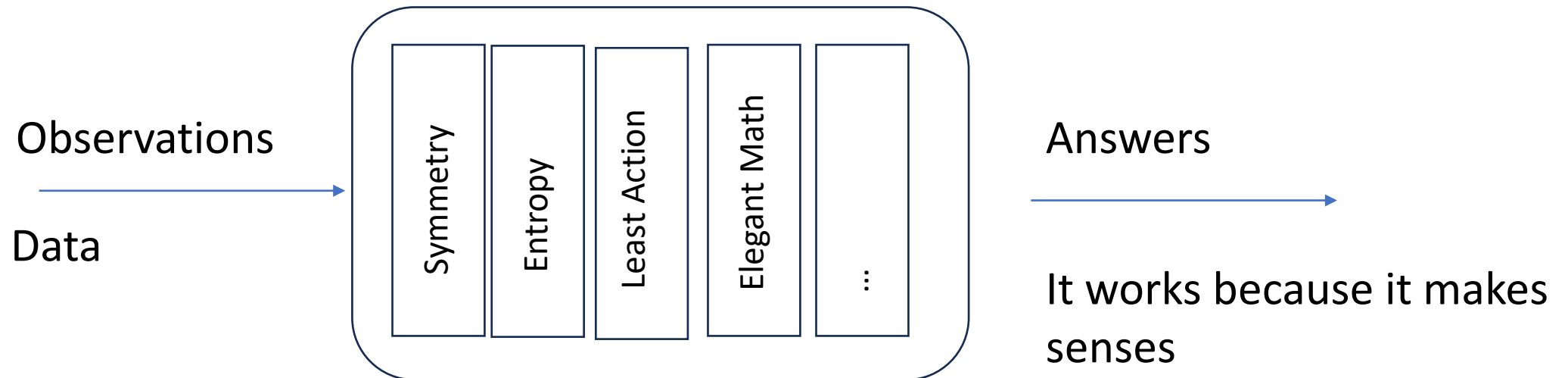
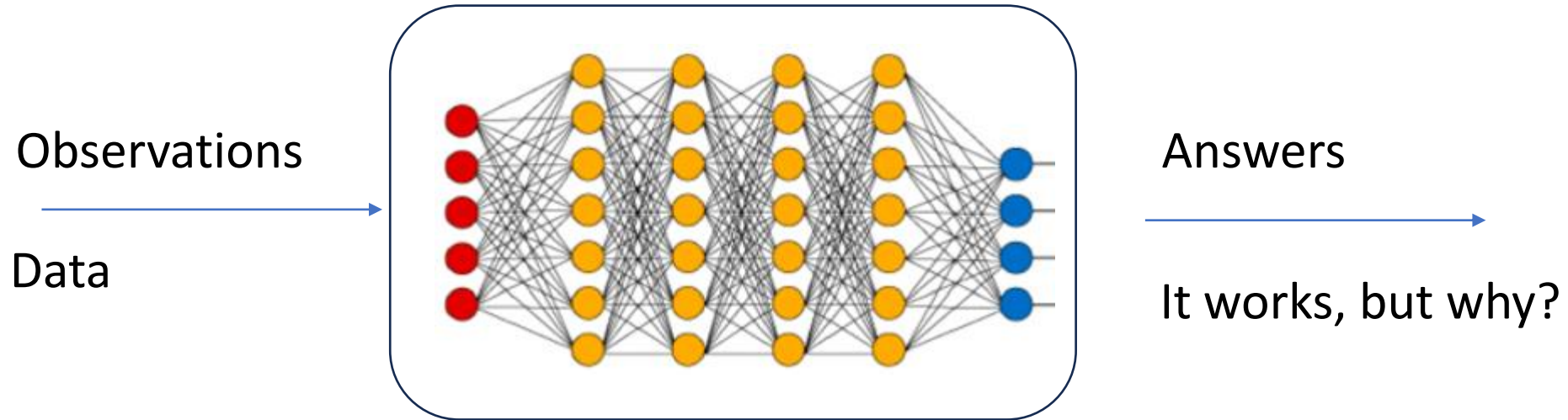
Answers, Data = Observations
Rules = Knowledge



Physicists Modelling

Observations => Models
Models => Answers
Models = Knowledge

Physicists Modelling is Better



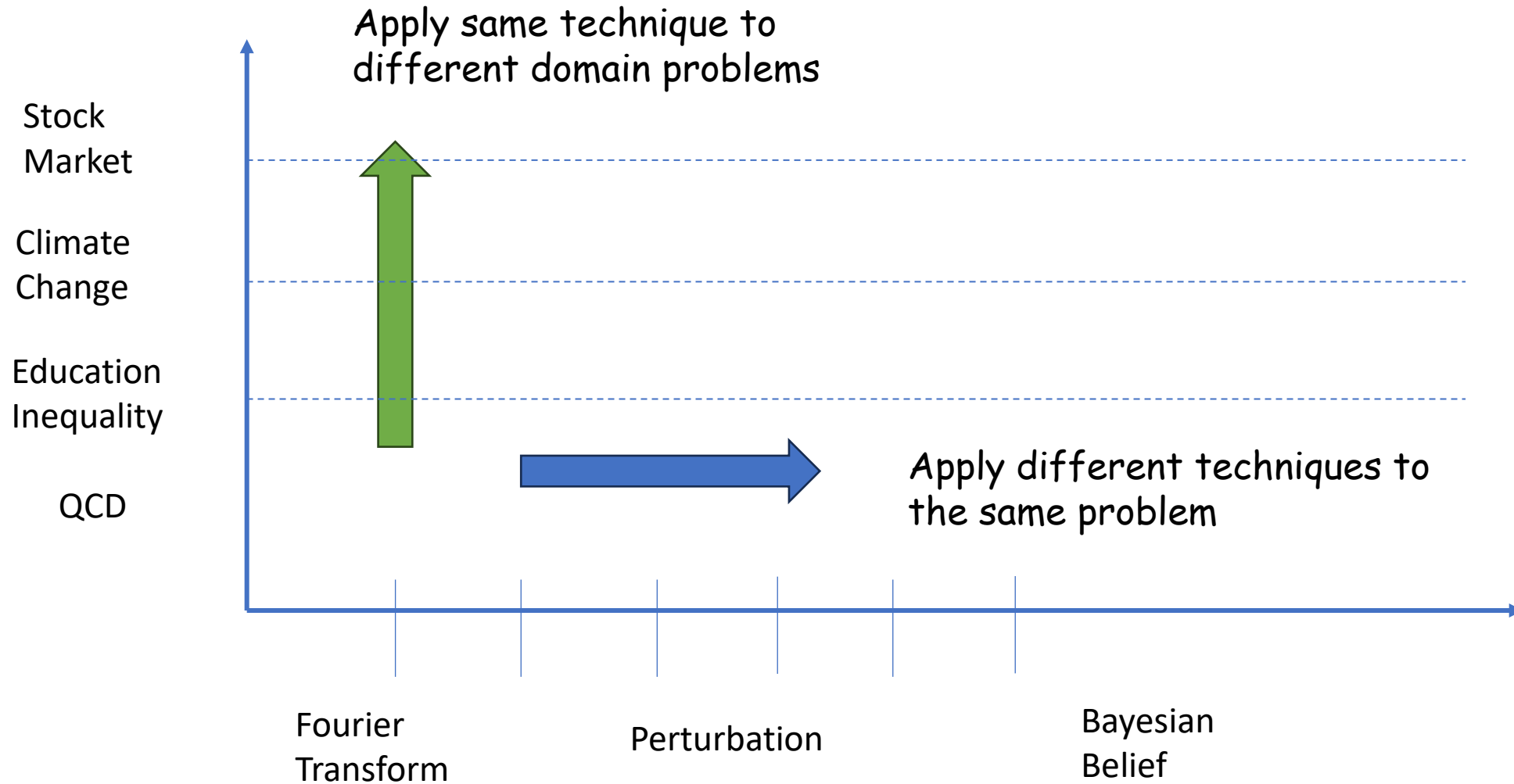
Next steps of AGI (Artificial General Intelligence)

An intentionally Blank Page

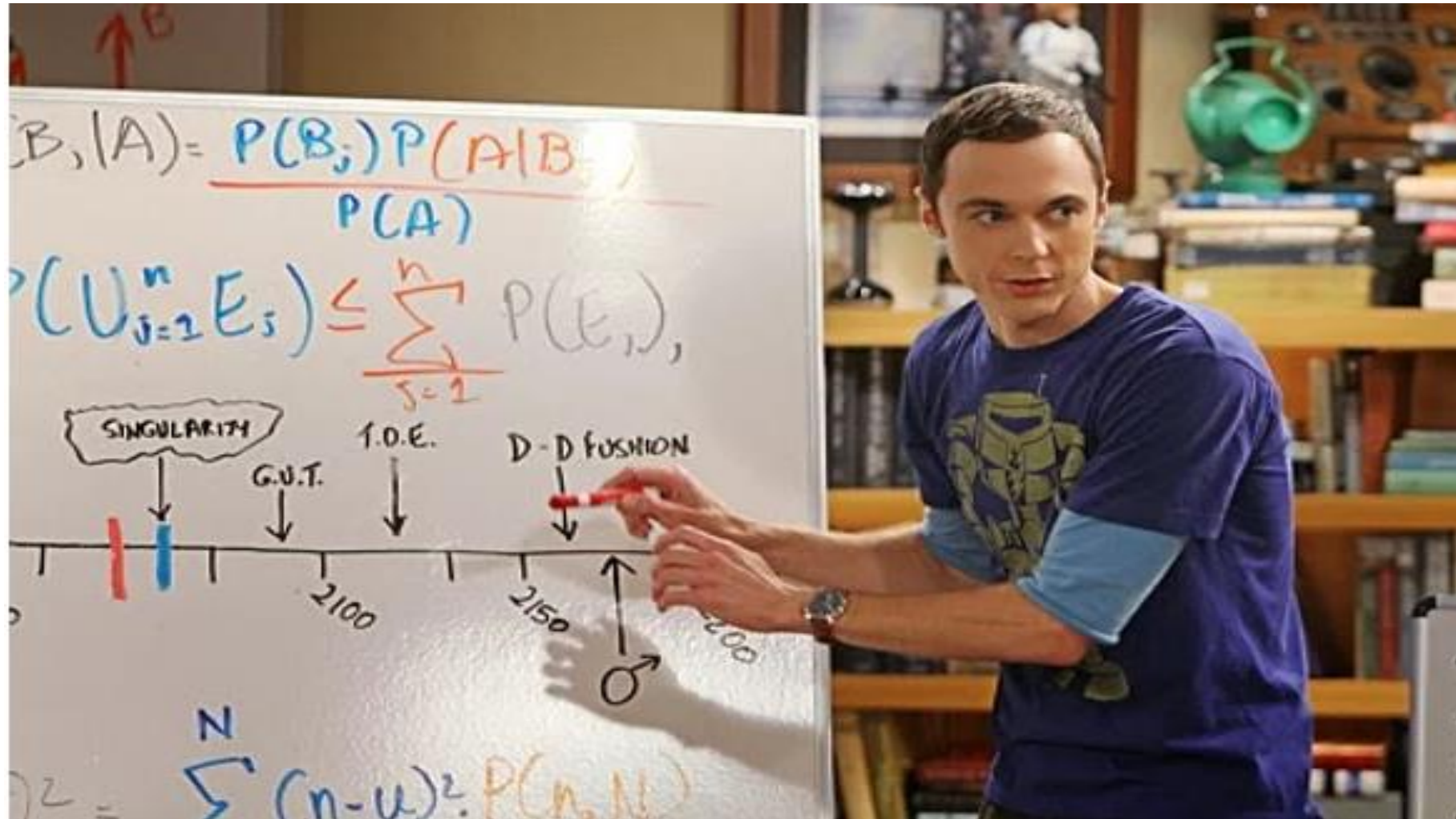
Next steps of AGI

Instead of just talk about various neural network architecture, I personally consider AGI has to be based on a search space with reasoning, logical deduction and inferential principle found in common human intelligence

Where do you get new research ideas



Sheldon Cooper is our guest lecturer today



Bayes Theorem

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

But A and B can be flipped: $P(A \cap B) = P(B \cap A) = P(A|B)P(B)$, therefore

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

This is called the "Bayes Theorem".

Note: $P(A) = P(A|B)P(B) + P(A|\sim B)P(\sim B)$ where the $\sim B$ means the NOT B event (or complement of B)

Apply Bayes Theorem to ML classification

In classification, we want to predict the label on given certain features

$$P(\text{label} \mid \text{features}) = \frac{P(\text{features} \mid \text{label}) P(\text{label})}{P(\text{features})}$$

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i \mid y)}{P(x_1, \dots, x_n)}$$

$$P(y \mid x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i \mid y)$$

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^n P(x_i \mid y)$$

That is, we predict y by choosing the class that maximize the A Posteriori (MAP) distribution ($P(y \mid x)$)

Examples of the blue arrow methodology

- [\[2209.03546\] Extending the Predictive Power of Perturbative QCD Using the Principle of Maximum Conformality and Bayesian Analysis \(arxiv.org\)](#)

....

In this paper we propose a novel Bayesian-based approach to estimate the size of the unknown higher order contributions based on an optimized analysis of probability distributions. We show that by using the PMC conformal series, in combination with the Bayesian analysis, one can consistently achieve high degree of reliability estimates for the unknown high order terms. Thus the predictive power of pQCD can be greatly improved.

...

Apply same techniques to a new domain problem

Example of the green arrow methodology

- The whole AI frenzy is based on the ideas of using chatbot as a personal assistant such as
 - Coming up with itinerary planning
 - Scheduling meetings with others
- In Physics, can you develop a symbolic program to help Physicist to perform standard calculations which is well-defined, but just taking too much efforts.

A few more interesting ideas and results

Eric Schmidt: This is how AI will transform the way science gets done

Science is about to become much more exciting—and that will affect us all, argues Google's former CEO.

By Eric Schmidt

July 5, 2023

- [Eric Schmidt: This is how AI will transform how science gets done | MIT Technology Review](#)
- [AI Reveals New Possibilities in Matrix Multiplication | Quanta Magazine](#)
- [AlphaDev discovers faster sorting algorithms \(deepmind.com\)](#)

20 Years in Making

- [Stephen Wolfram: A New Kind of Science | Online—Table of Contents \(wolframscience.com\)](#)
- [The Wolfram Physics Project hopes to find fundamental theory of physics](#)

He called it points and rules. A neural network has points and rules (sigmoid function, non-linear activation function).

A neural network can be thought of as a multi-agents with the rules for interacting agents (nodes) as simple as linear + non-linear activation function

In finance, the whole market is an aggregated voting machine from individual investors.

Reinforcement Learning is focus on how an agent interact with its environment. Choose actions and receive award from the environment

Extending current Multi-agents modelling problem is something that I would love to explore as my main focus for the next 5 to 10 years

The End

Or Shall I say
The beginning