EEP 596: LLMs: From Transformers to GPT || Lecture 3

Dr. Karthik Mohan

Univ. of Washington, Seattle

January 12, 2026

Logistics

- Assignment 1
- Paper Presentation Slots Assignment
- Assignment 2
- Anything else?

Last Lecture

- History of Large Language Models
- LLM use-cases in the industry

Today's Lecture

- Deep Learning Fundamentals

Outline for Lecture

- Motivation for DL

Outline for Lecture

- Motivation for DL
- DL Applications

Outline for Lecture

- Motivation for DL
- DL Applications
- DL History

Outline for Lecture

- Motivation for DL
- DL Applications
- DL History
- Logistic Regression

Outline for Lecture

- Motivation for DL
- DL Applications
- DL History
- Logistic Regression
- Deep Learning Models

Outline for Lecture

- Motivation for DL
- DL Applications
- DL History
- Logistic Regression
- Deep Learning Models
- Activation functions

Outline for Lecture

- Motivation for DL
- DL Applications
- DL History
- Logistic Regression
- Deep Learning Models
- Activation functions
- Tensorflow Demo

Outline for Lecture

- Motivation for DL
- DL Applications
- DL History
- Logistic Regression
- Deep Learning Models
- Activation functions
- Tensorflow Demo
- Training and Back-propagation

Outline for Lecture

- Motivation for DL
- DL Applications
- DL History
- Logistic Regression
- Deep Learning Models
- Activation functions
- Tensorflow Demo
- Training and Back-propagation
- Over-fitting and Hyper-parameters

Outline for Lecture

- Motivation for DL
- DL Applications
- DL History
- Logistic Regression
- Deep Learning Models
- Activation functions
- Tensorflow Demo
- Training and Back-propagation
- Over-fitting and Hyper-parameters
- Other DL architectures

Deep Learning Reference

Deep Learning

Great reference for the theory and fundamentals of deep learning: Book by Goodfellow and Bengio et al Bengio et al

Deep Learning History

Introduction to Deep Learning

Deep Learning

1. Lot of buzz around Deep Learning in the past decade and a half!

Introduction to Deep Learning

Deep Learning

1. Lot of buzz around Deep Learning in the past decade and a half!
2. Deep Learning refers to Neural Networks that is a loose approximation of how the brain works

Applications of Deep Learning

Applications

1. Self-driving cars

Applications of Deep Learning

Applications

1. Self-driving cars
2. Sentiment analysis

Applications of Deep Learning

Applications

1. Self-driving cars
2. Sentiment analysis
3. Text Summarization

Applications of Deep Learning

Applications

1. Self-driving cars
2. Sentiment analysis
3. Text Summarization
4. Arrythmia detection - Possible assignment for this course!

Applications of Deep Learning

Applications

1. Self-driving cars
2. Sentiment analysis
3. Text Summarization
4. Arrythmia detection - Possible assignment for this course!
5. Image to text generation. Caption images automatically.

Applications of Deep Learning

Applications

1. Self-driving cars
2. Sentiment analysis
3. Text Summarization
4. Arrythmia detection - Possible assignment for this course!
5. Image to text generation. Caption images automatically.
6. Machine Translation. Translate a French sentence to English sentence. Sequence to sequence architecture

Applications of Deep Learning

Applications

1. Self-driving cars
2. Sentiment analysis
3. Text Summarization
4. Arrythmia detection - Possible assignment for this course!
5. Image to text generation. Caption images automatically.
6. Machine Translation. Translate a French sentence to English sentence. Sequence to sequence architecture
7. Auto-complete sentence in Emails. How many of us use this?

Applications of Deep Learning

Applications

1. Self-driving cars
2. Sentiment analysis
3. Text Summarization
4. Arrythmia detection - Possible assignment for this course!
5. Image to text generation. Caption images automatically.
6. Machine Translation. Translate a French sentence to English sentence. Sequence to sequence architecture
7. Auto-complete sentence in Emails. How many of us use this?
8. Auto-complete search results.

Applications of Deep Learning

Applications

1. Self-driving cars
2. Sentiment analysis
3. Text Summarization
4. Arrythmia detection - Possible assignment for this course!
5. Image to text generation. Caption images automatically.
6. Machine Translation. Translate a French sentence to English sentence. Sequence to sequence architecture
7. Auto-complete sentence in Emails. How many of us use this?
8. Auto-complete search results.
9. Chat bots - Like ChatGPT/Sparrow/Anthropic, etc

Email auto-complete

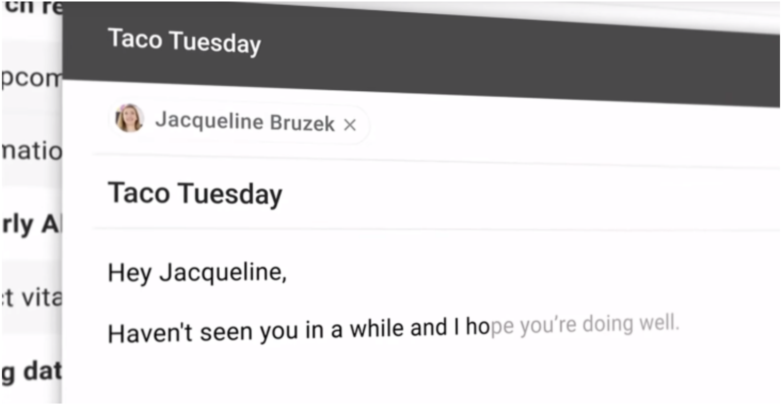
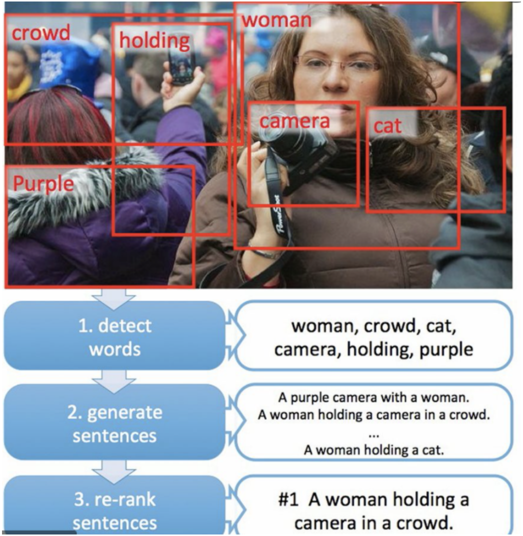


Image to Text!



Arrhythmia Detection

Industry Applications (NLP focused)

- Tagging prodcuts at a e-commerce companies with tags/category ids
- Coming up with category names, when none existed before
- Summarizing reviews for products
- Customer service help - Chatbots
- Live Speech Translation
- Intent Recognition: Understanding intent behind a sentence or paragraph
- Detecting malicious intent and content in text (part of Responsible AI)
- Converting natural language instructions into SQL queries to run and respond back in natural language
- And many more!

Brief History of Deep Learning

- **1965:** First deep-learning model came out in 1965 by Ivakhenko et al. Didn't use back-propagation for training but sequential least squares fit.
- **1979:** Earliest Convolutional Neural Network (CNN) by Fukushima et al.
- **1985:** Earliest back-propagation in 1985 by Hinton et al
- **1989:** Application of back-prop for recognizing MNIST hand-written digits at Bell labs by Yann LeCun
- **1993:** **LeNet** by Yann LeCun. The beginning of the **X-Nets** where X could be Alex, Inception, etc
- **1997:** Discovery of recurrent Neural Nets - RNN and LSTMs in 1997 by Horchreiter and Schmidhuber.

Brief History of Deep Learning (and LLMs)

- **1997 - 2006:** GPUs got faster - 1000x computational speed improvement
- **2011:** Ciresan et al showed that you can train a CNN without pre-trained weights just with good computational power.
- **2012:** Beginning of ILSVRC competition for improving image-net data set performance.
- **2017:** Transformers arrive on the scene with Vaswani et al and begin the **Language Model revolution**.
- **2020:** Transformer gets applied to Vision as well and matches CNN in performance through the Vi-Transformer.
- **2022:** ChatGPT (based on transformers) arrives on the scene and puts AI on the world map!

Brief History of Deep Learning (and LLMs)

- **2023:** Llama2 released with model weights (7b, 13b, 70b) by Meta and sees wider adoption as an open source alternative to ChatGPT. Context length of 4096 tokens
- **2024:** Llama3 released with 8b, 70b and 450b parameters. Context length of 8192 tokens but can extend to 128k tokens. Most popular and performant Meta model so far (unfortunately Llama4 didn't match expectations!)
- **2025:** DeepSeek releases their models with low-budget and architecture additions such as MLA and MOE, shaking up the industry and forcing big players to innovate!
- **2026:** Quest for super-intelligence continues as research explore multi-modal reasoning and mimicking how human brains work and reason with Thinking Machines and AMI as examples
- **2026:** DeepSeek again releases a paper that doesn't just train existing transformers architecture on better data, but re-does the architecutre for better results!

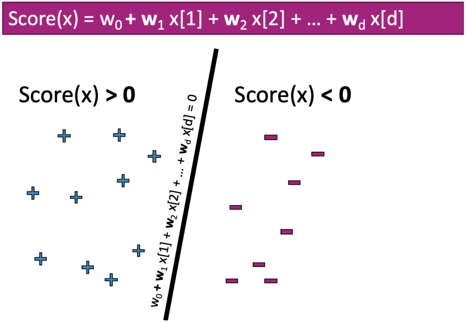
Perceptron to Deep Neural Networks/Deep Learning

Logistic Regression to Deep Learning

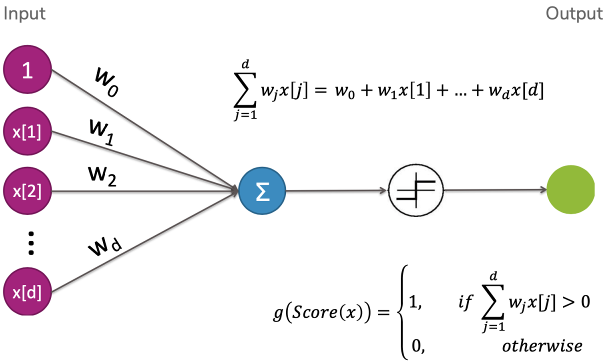
Linear to Non-linear Models

Let's work through the nitty-gritties of the logistic regression model and neural network model!

Perceptron



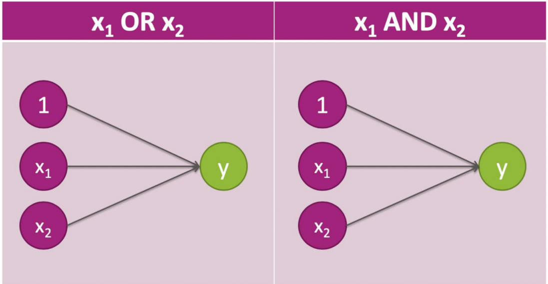
Perceptron



Input
Output
1
x[1]
x[2]
x[d]
w_0
w_1
w_2
w_d
Σ
$\sum_{j=1}^{d} w_j x[j] = w_0 + w_1 x[1] + \ldots + w_d x[d]$
$g(Score(x)) = \begin{cases} 1, & if \sum_{j=1}^{d} w_j x[j] > 0 \\ 0, & otherwise \end{cases}$

OR and AND Functions

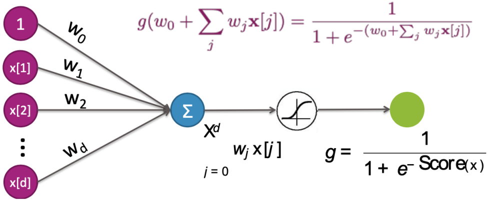
What can a perceptrons represent?



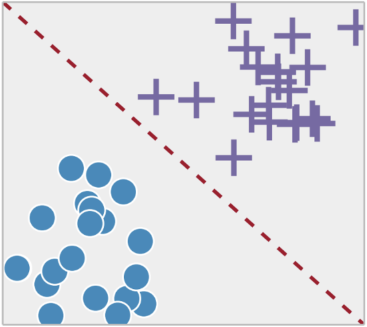
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Perceptron to Logistic Regression



Logistic Regression



LR fundamentals

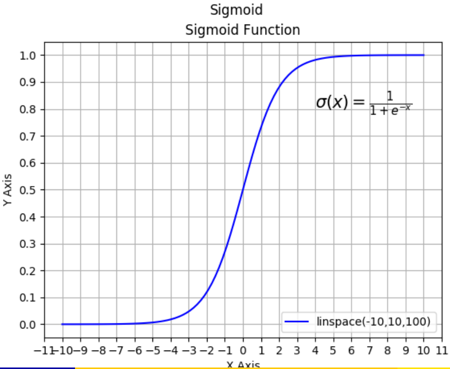
- Linear Model
- Want score $w^T x^i > 0$ for $y_i = +1$ and $w^T x_i < 0$ for $y_i = -1$!
- If linearly separable data, above is feasible. Else, minimize error in separability!!

Logistic Regression

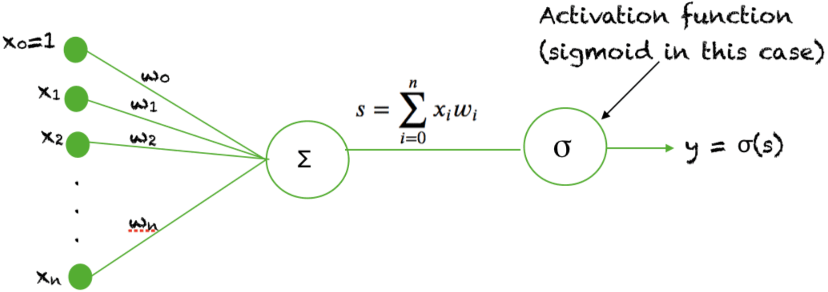
Probability for a class

In LR, the score, $w^T x$ is converted to a probability through the sigmoid function. So we can talk about $P(\hat{y}^i = +1)$ or $P(\hat{y}^i = -1)$

Sigmoid Function



LR represented Graphically



ICE #1

Compute $w^T x$ for $w = [1, 2, 3]$ and $x = [-1, 1, 2]$

1. 5
2. 6
3. 7
4. 8

Logistic Regression

LR Prediction

$$\hat{y}_i = \frac{1}{1 + e^{-\hat{w}^T x^i}}$$

LR Loss

Assume that $y_i = 0$ or $y_i = 1$ (i.e. the negative class has a label 0). Then the binary cross-entropy loss applies to LR:

$$\min_w -y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

ICE #2

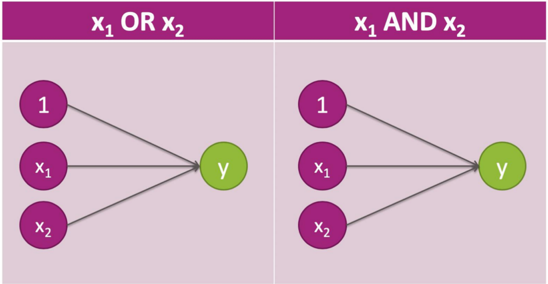
If log is to the base 2, what is the value of binary-cross entropy at a ground truth of 1 and prediction label of 0.5? Binary cross-entropy is given by,

$$-y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)$$

1. 0
2. 0.5
3. 1
4. 2

OR and AND Functions

What can a perceptrons represent?



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Learning XOR

XOR through Multi-layer perceptron

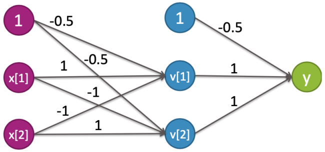
This is a 2-layer neural network

$y = x[1]\ XOR\ x[2] = (x[1]\ AND\ !\,x[2])\ OR\ (!\,x[1]\ AND\ x[2])$

$$v[1] = (x[1]\ AND\ !\,x[2]) \\ = g(-0.5 + x[1] - x[2])$$

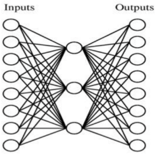
$$v[2] = (!\,x[1]\ AND\ x[2]) \\ = g(-0.5 - x[1] + x[2])$$

$$y = v[1]\ OR\ v[2] \\ = g(-0.5 + v[1] + v[2])$$



2 Layer Neural Network

Two layer neural network (alt. one hidden-layer neural network)



Single

$$out(x) = g\left(w_0 + \sum_j w_j x[j]\right)$$

1-hidden layer

$$out(x) = g\left(w_0 + \sum_k w_k g\left(w_0^{(k)} + \sum_j w_j^{(k)} x[j]\right)\right)$$

ICE #3

Which methods can learn the XOR function?

1. Logistics Regression
2. Naive Bayes Classifier
3. Decision Trees
4. Support Vector Machines

Deep Learning: Activations, FFN and more

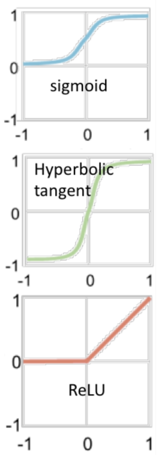
Choices for Non-Linear Activation Function

•Sigmoid
-Historically popular, but (mostly) fallen out of favor
•Neuron's activation saturates
(weights get very large -> gradients get small)
•Not zero-centered -> other issues in the gradient steps
-When put on the output layer, called "softmax" because interpreted as class probability (soft assignment)

•Hyperbolic tangent $g(x) = \tanh(x)$
-Saturates like sigmoid unit, but zero-centered

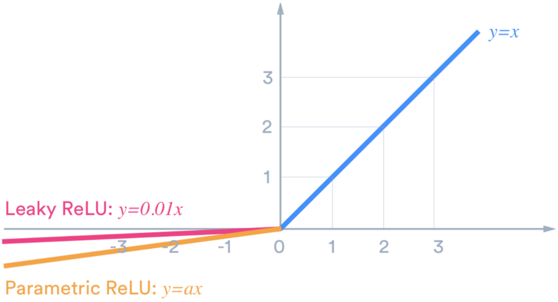
•Rectified linear unit (ReLU) $g(x) = x^+ = \max(0,x)$
-Most popular choice these days
-Fragile during training and neurons can "die off"… be careful about learning rates
-"Noisy" or "leaky" variants

•Softplus $g(x) = \log(1+\exp(x))$
-Smooth approximation to rectifier activation



Sigmoid vs RELU

RELU vs Leaky RELU

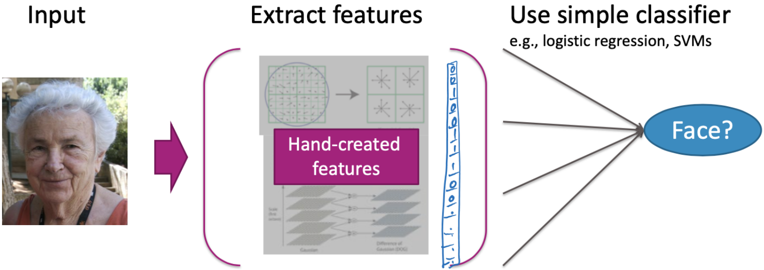


Phenomenon of Overfitting

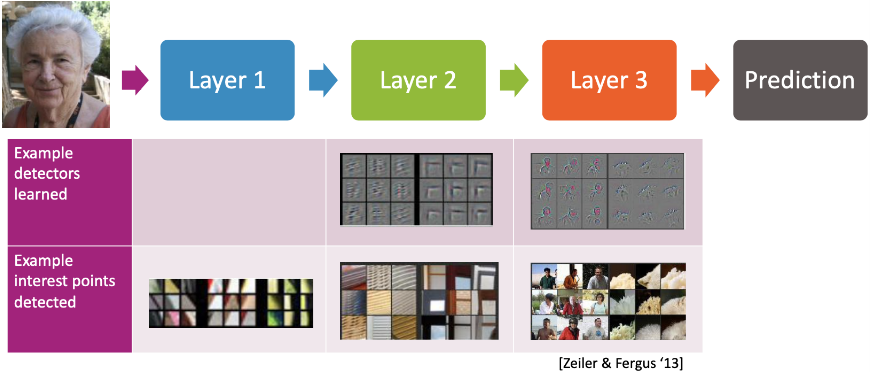
Tensorflow Playground Demo

Tensorflow Playground Demo

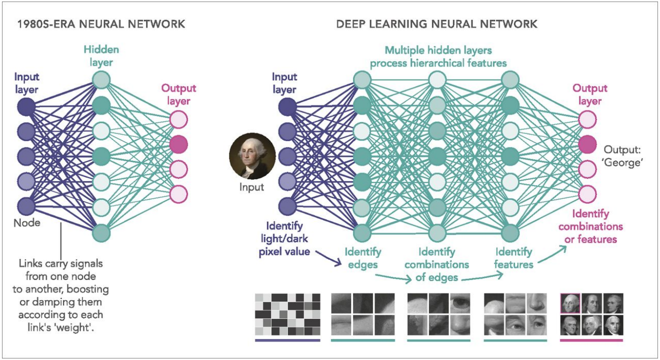
Computer vision before deep learning



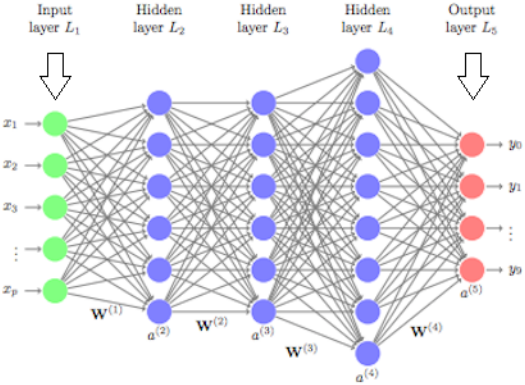
Computer vision after deep learning



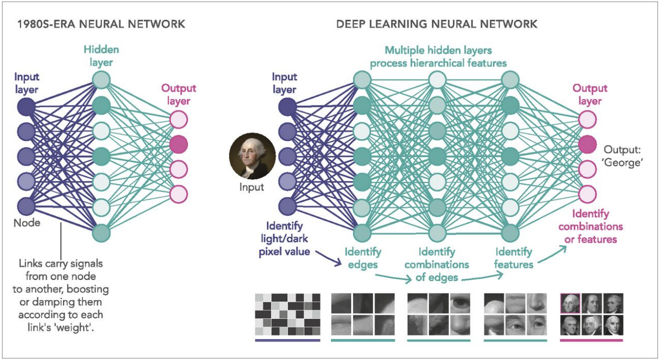
Feed-forward Deep Learning Architecture Example



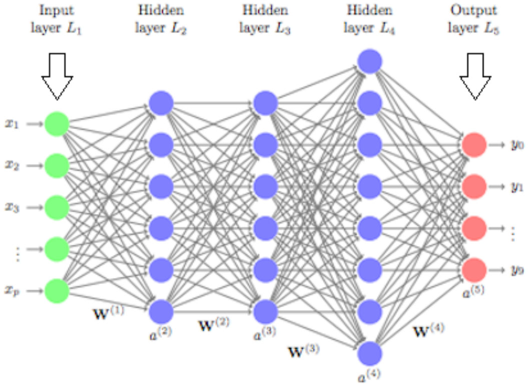
Feed-forward Deep Learning Architecture Example



Feed-forward Deep Learning Architecture Example



Feed-forward Deep Learning Architecture Example



ICE #4

Compute the number of parameters in DNN model

Consider a DNN model with 3 hidden layers where each hidden layer has 1000 neurons. Let the input layer be raw pixels from a 100x100 image and the output layer has 10 dimensions, let's say for a 10 class image classification example. How many total parameters exist in the DNN model?

1. 10 million parameters
2. 11 million parameters
3. 12 million parameters
4. 13 million parameters

Training a DNN

SGD with mini-batch

SGD mini-batch is the staple diet. However there are some **learning rate schedulers** that are known to work better for DNNs - Such as Adagrad and more recently, ADAM. ADAM adapts the learning rate to each individual parameter instead of having a global learning rate.

Training a DNN

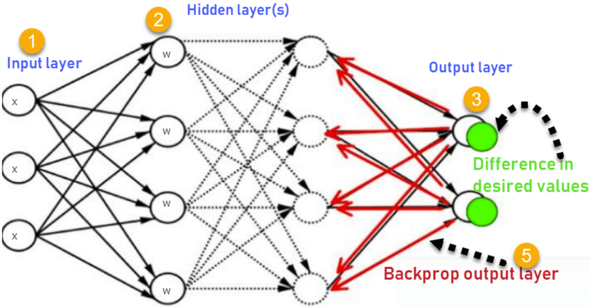
SGD with mini-batch

SGD mini-batch is the staple diet. However there are some **learning rate schedulers** that are known to work better for DNNs - Such as Adagrad and more recently, ADAM. ADAM adapts the learning rate to each individual parameter instead of having a global learning rate.

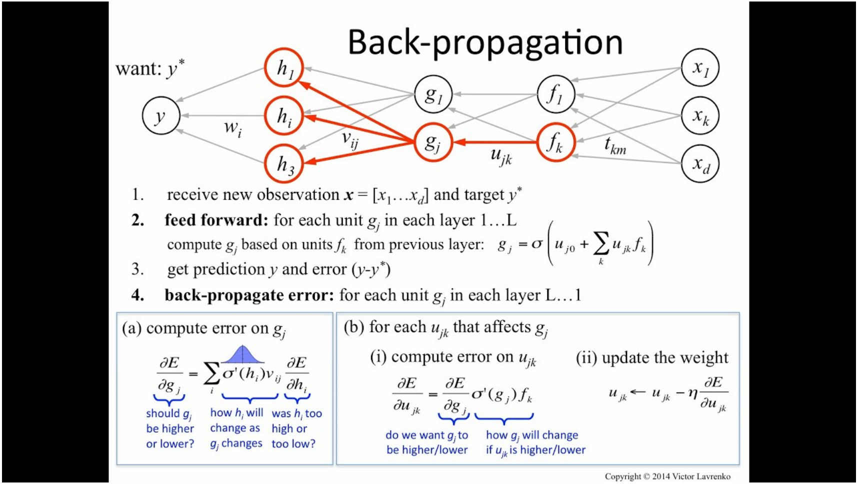
How do we compute gradient in a DNN?

Back-propagation!

Forward Propagation vs Back-propagation



Back Propagation explained



Back Propagation Summary

Back Prop

Back prop is one of the fundamental backbones of the training modules behind deep learning and beyond (including for example ChatGPT). What exactly is back prop? It is just a way to unravel gradient computation in the neural network. Back prop is how we would **compute the gradient** in a neural network.

Back Propagation Summary

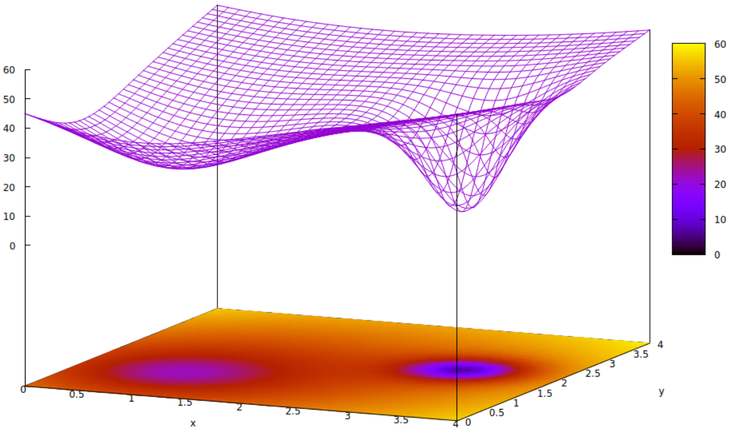
Back Prop

Back prop is one of the fundamental backbones of the training modules behind deep learning and beyond (including for example ChatGPT). What exactly is back prop? It is just a way to unravel gradient computation in the neural network. Back prop is how we would **compute the gradient** in a neural network.

Back Prop as information flow

It can also be thought of as flow information from the error in the output (the loss function) down to the weights. Update the weights so we don't make **this error** next time around. Back prop is a way to do **gradient descent in neural networks!**

Good vs Bad Local minima



Hyper-parameters in Deep Learning

ICE #5: Which of the following is not a hyper-parameter in deep learning?

1. Learning rate
2. Number of Hidden Layers
3. Number of neurons per hidden layer
4. All of the above

Hyper-parameters in Deep Learning

Hyper-parameters

1. Learning rate
2. Number of Hidden Layers
3. Number of neurons per hidden layer

Hyper-parameters in Deep Learning

Hyper-parameters

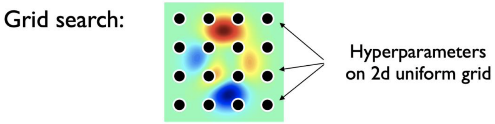
1. Learning rate
2. Number of Hidden Layers
3. Number of neurons per hidden layer
4. Type of non-linear activation function used

Hyper-parameters in Deep Learning

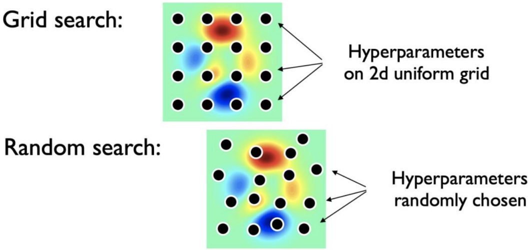
Hyper-parameters

1. Learning rate
2. Number of Hidden Layers
3. Number of neurons per hidden layer
4. Type of non-linear activation function used
5. Anything else?

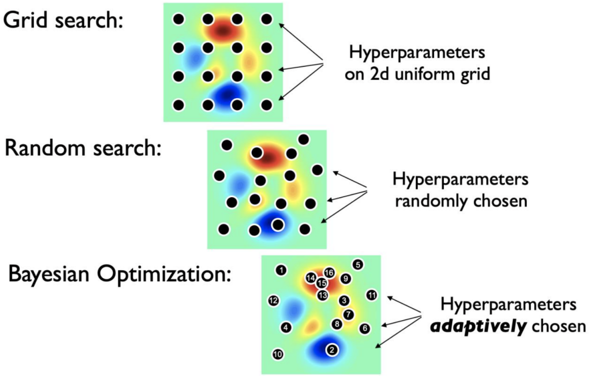
Hyper-parameter tuning methods



Hyper-parameter tuning methods



Hyper-parameter tuning methods



Over-fitting in DNNs

How to handle over-fitting in DNNs

1. A DNN model with 100 million parameters and only 100k data points or even a million data points will overfit unless we take care of over-fitting.

Over-fitting in DNNs

How to handle over-fitting in DNNs

1. A DNN model with 100 million parameters and only 100k data points or even a million data points will overfit unless we take care of over-fitting.
2. Weight regularization can help - ℓ_1, ℓ_2

Over-fitting in DNNs

How to handle over-fitting in DNNs

1. A DNN model with 100 million parameters and only 100k data points or even a million data points will overfit unless we take care of over-fitting.
2. Weight regularization can help - ℓ_1, ℓ_2
3. More common over-fitting strategy for DL?

Over-fitting in DNNs

How to handle over-fitting in DNNs

1. A DNN model with 100 million parameters and only 100k data points or even a million data points will overfit unless we take care of over-fitting.
2. Weight regularization can help - ℓ_1, ℓ_2
3. More common over-fitting strategy for DL?
4. Dropouts!

Over-fitting in DNNs

How to handle over-fitting in DNNs

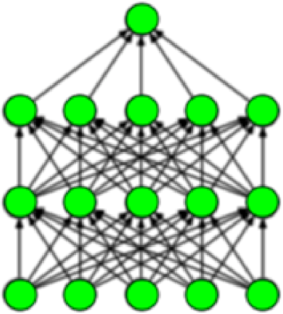
1. A DNN model with 100 million parameters and only 100k data points or even a million data points will overfit unless we take care of over-fitting.
2. Weight regularization can help - ℓ_1, ℓ_2
3. More common over-fitting strategy for DL?
4. Dropouts!
5. Early stopping is also a great strategy! Stop training the DL model when the validation error starts increasing. How's this different from regular validation we were doing earlier??

Over-fitting in DNNs

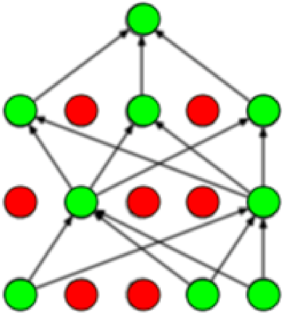
How to handle over-fitting in DNNs

1. A DNN model with 100 million parameters and only 100k data points or even a million data points will overfit unless we take care of over-fitting.
2. Weight regularization can help - ℓ_1, ℓ_2
3. More common over-fitting strategy for DL?
4. Dropouts!
5. Early stopping is also a great strategy! Stop training the DL model when the validation error starts increasing. How's this different from regular validation we were doing earlier??
6. Book by Yoshua Bengio has tons of details and great reference for Deep Learning!

Taking care of Over-fitting: Dropouts



(a) Standard Neural Net



(b) After applying dropout.

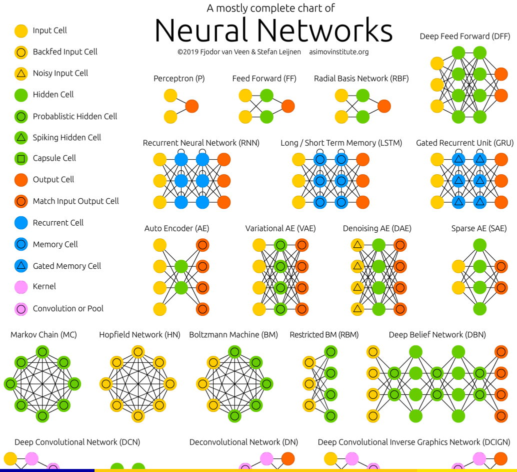
Tensorflow Playground Demo

Tensorflow Playground Demo

More DL Architectures

Neural Networks Zoo

Zoo Reference



More DL Architectures

Neural Networks Zoo