Computer Vision: Fall 2022 — Lecture 8

Dr. Karthik Mohan

Univ. of Washington, Seattle

October 27, 2022

Check-In

1. How was Assignment 3?

Check-In

1. How was Assignment 3?
2. Next Assignment: Mini-Project

Check-In

1. How was Assignment 3?
2. Next Assignment: Mini-Project
3. Other thoughts/questions?

References

1. Good Book for Machine Learning Concepts
2. Deep Learning Reference

Yoshua Bengio

Mini-Project

10 classes

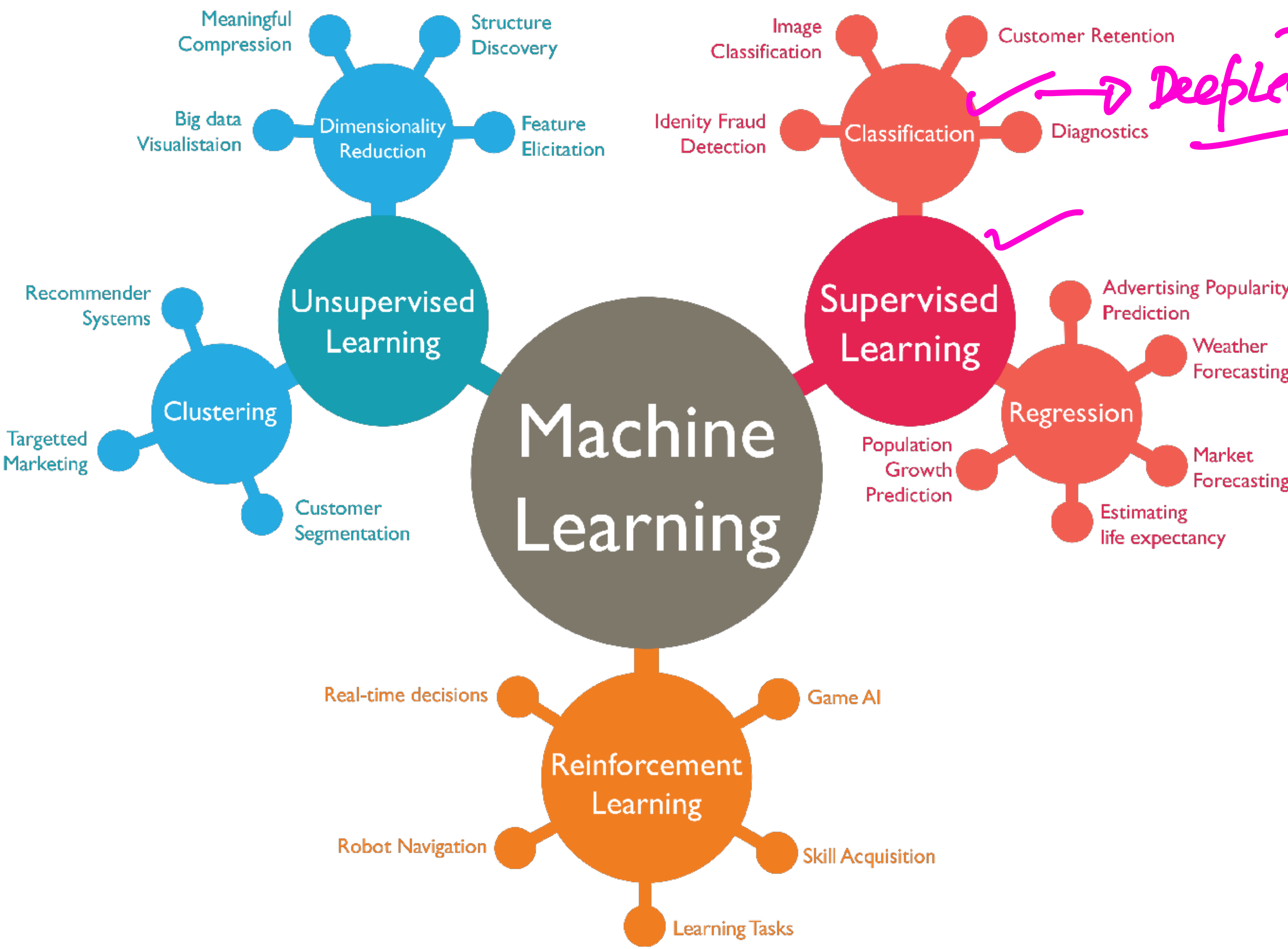
- **Multi-class classification:** On Fashion MNIST data set. Given an image - Pick the appropriate class for it.
- **Deliverables:** You have to submit a Jupyter/IPython notebook file and report as part of your submission. You can use the template notebook given and add your solutions to it.
- **Team Work:** You can work in a team of 2. Pick your team mate for this project - When you make your report submission, you are expected to breakdown the contribution of each team member. Ensure that both team members get to work and test the Neural Network models.
- **Report:** The report should be in pdf format and have all images, plots and metrics added in it. Feel free to use either latex or word for creating it. You are required to answer all of the conceptual questions in the write up below, and show your learnings and insights.

Mini-Project

- You may discuss/brainstorm ideas to solve the assignment with peers - However, your submission should be your own and show your code implementation.
- **Kaggle Contest:** There is a Kaggle competition as well for this assignment, submit your predictions on the "held out" test data set for a fun peer learning experience!

Today

1. Neural Networks/Deep Learning
2. Back propogation
3. Overfitting in Deep Learning



Deep Learning

Computer Vision Topics

1. Image Processing using convolutions
2. Image De-noising
3. Image Smoothing
4. Image Clustering
5. Image Classification
6. Object Detection
7. Semantic Segmentation
8. Instance Segmentation (maybe)
9. Image Embeddings
10. Image to Text
11. Image Captioning
12. Text to Image (high-level)

DL (CNN)

Introduction to Deep Learning

Deep Learning

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

Introduction to Deep Learning

1. More Data to work with
2. More compute Power (GPUs)

Deep Learning

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

$i \xrightarrow{f} O$

SOTA

1) Image — caption
2) Text — Image
3) Image — class (MP1)

DL → Universal function Approximator

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 - What's an example application for this?

Applications of Deep Learning

Applications

1. Self-driving cars
2. Sentiment analysis
3. Text Summarization - What's an example application for this?
4. Arrythmia detection - P~~ossible assignment for this course~~!

Applications of Deep Learning

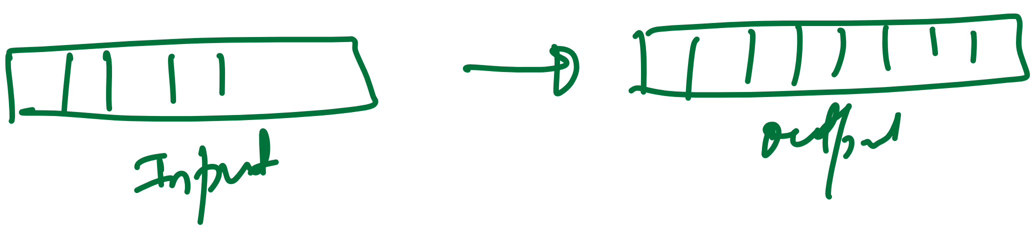
Applications

1. Self-driving cars
2. Sentiment analysis
3. Text Summarization - What's an example application for this?
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 - What's an example application for this?
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 - What's an example application for this?
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 - What's an example application for this?
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 - What's an example application for this?
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!

Email auto-complete

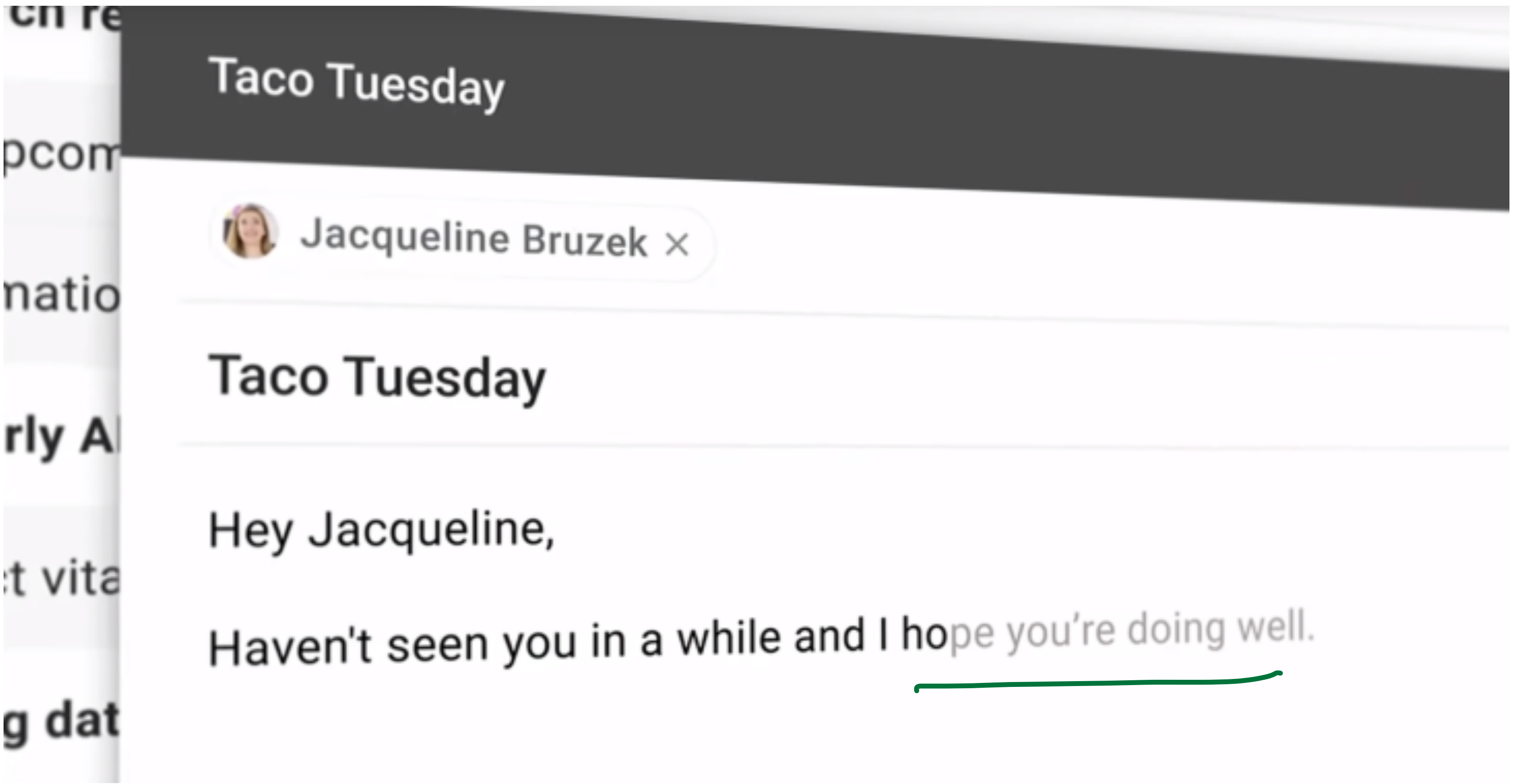
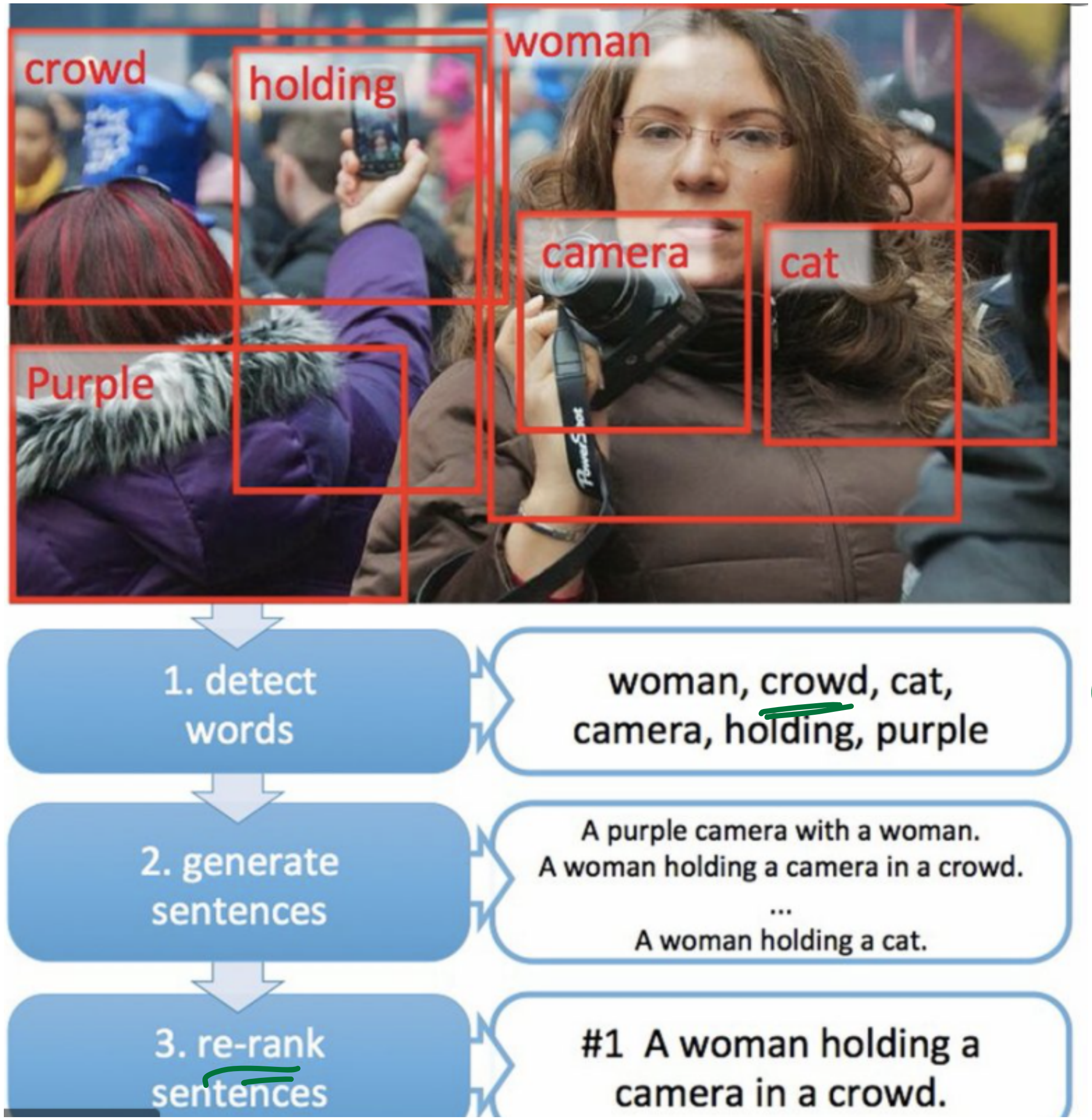


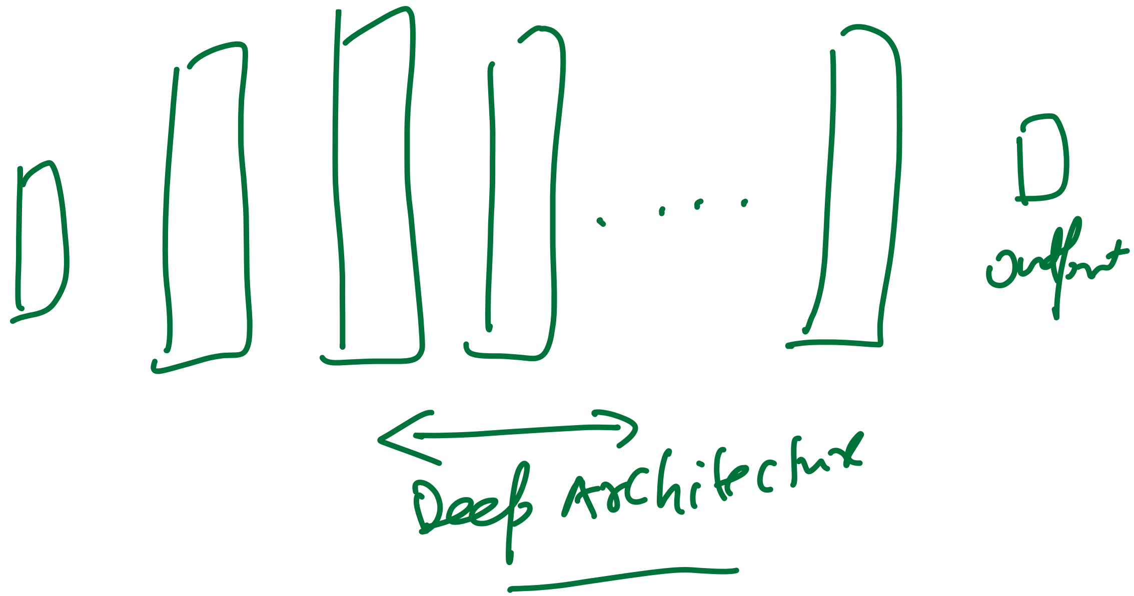
Image to Text!



(object detection)
to word detection
to Sentence descrpt

Deep Learning vs Neural Networks!

Lots of Data "Deep Architecture"



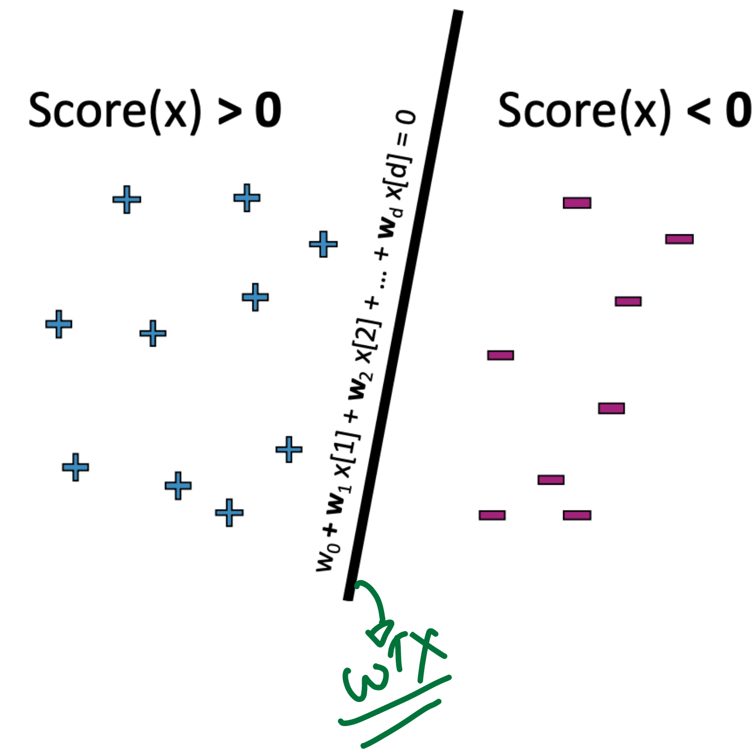
Build up to NN/Deep Learning

1. Perceptron - Most simplest neural representation
2. Logistic Regression
3. Multi-layer perceptron (MLP)
4. NN and Deep Learning

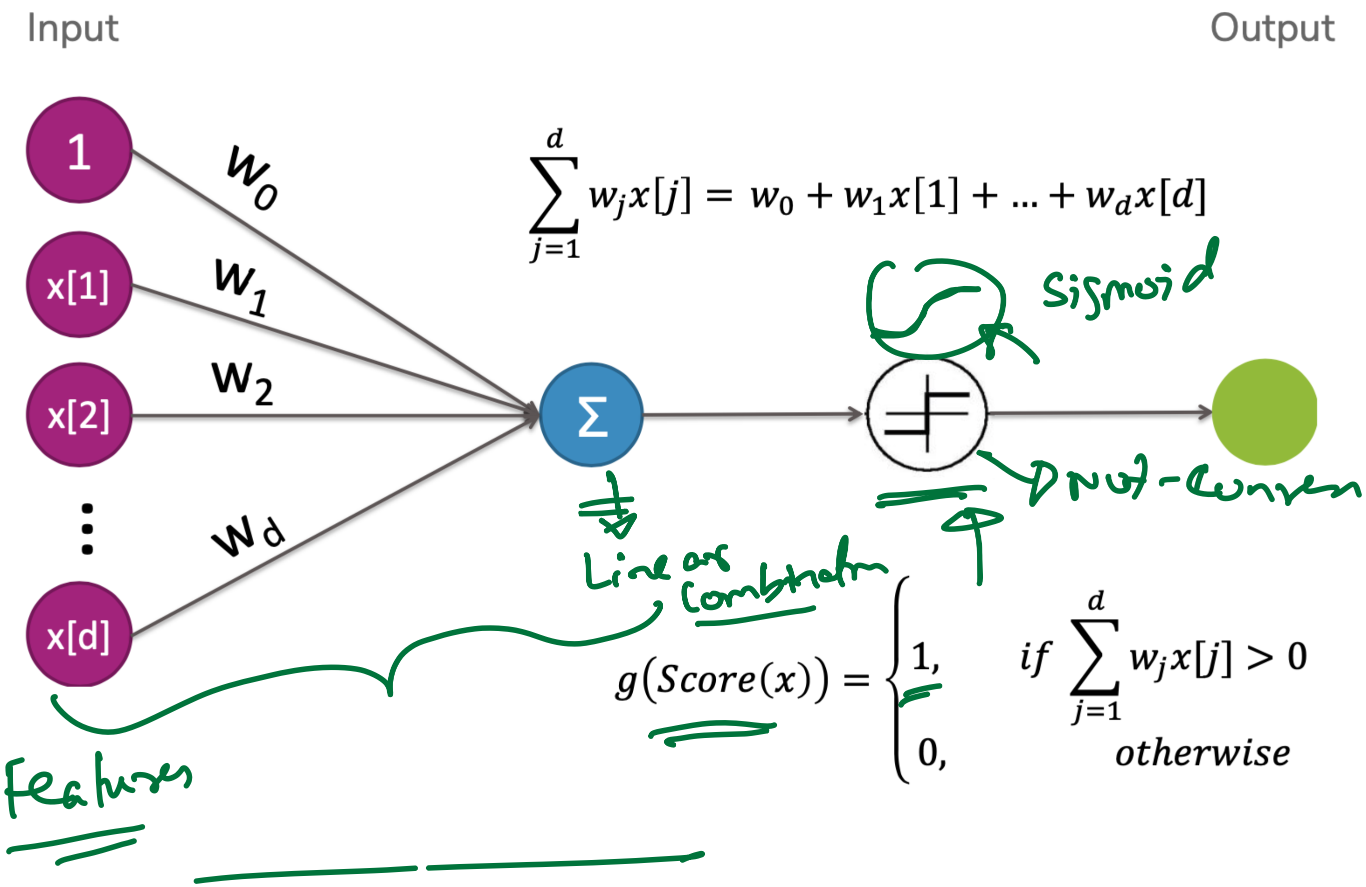
Build-up

Perceptron

$\text{Score}(x) = w_0 + \mathbf{w}_1\, x[1] + \mathbf{w}_2\, x[2] + \ldots + \mathbf{w}_d\, x[d]$



Perceptron



$$\sum_{j=1}^{d} w_j x[j] = w_0 + w_1 x[1] + \ldots + w_d x[d]$$

$$g(Score(x)) = \begin{cases} 1, & if \sum_{j=1}^{d} w_j x[j] > 0 \\ 0, & otherwise \end{cases}$$

ICE #1

Convex Concave Non-Convex Global Minimum

Perceptron vs Logistic Regression?

What's the difference between the Perceptron architecture and Logistic Regression Model we looked at previously?

1. They are both the same
2. Perceptron is a non-linear model while Logistic Regression is a linear model
3. Perceptron and Logistic Regression differ in the activation function that is used
4. Perceptron leads to a non-convex loss function while Logistic Regression yields a convex loss function

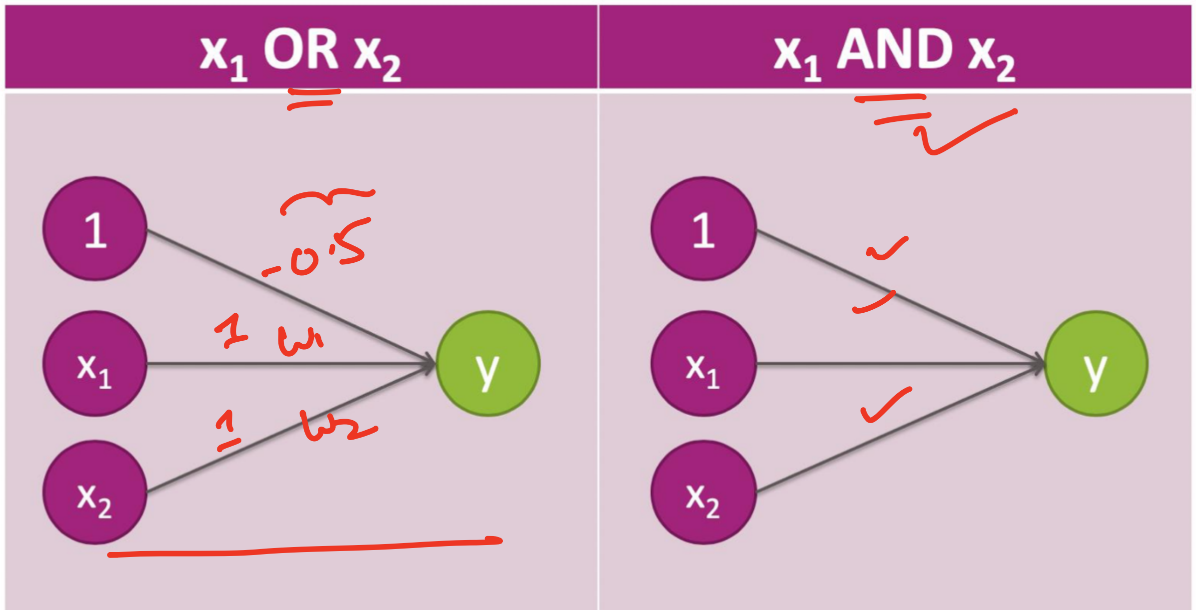
OR and AND Functions

What can a perceptrons represent?

1
0

x_1 1

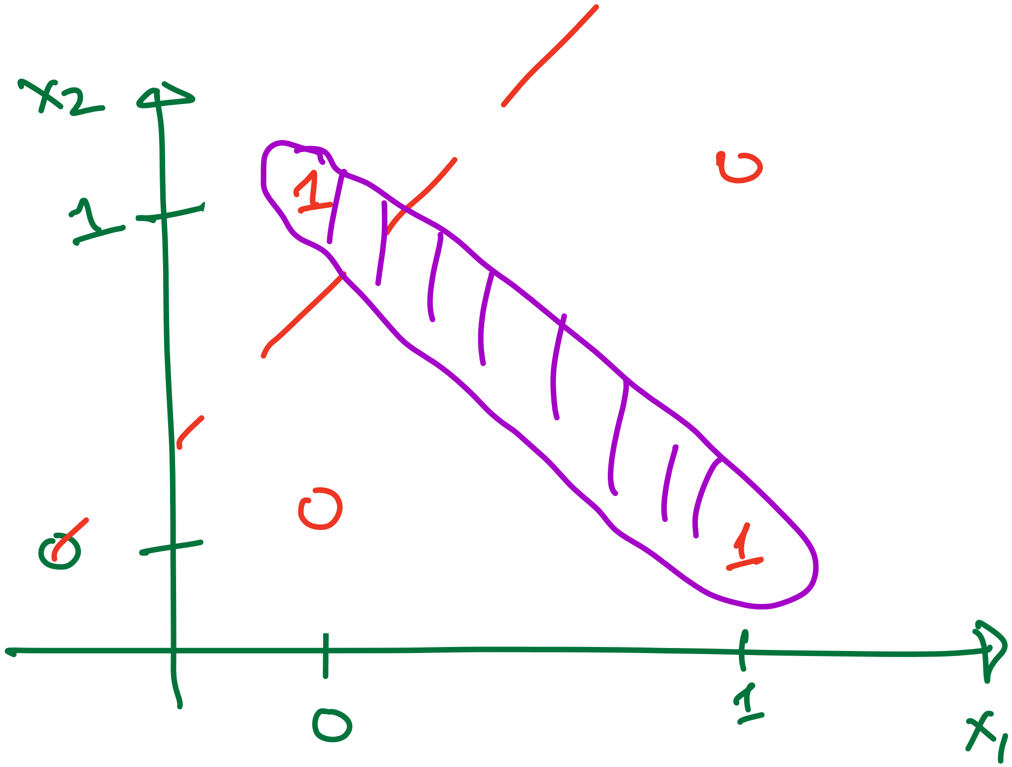
x_2 0



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

Linear Model

Non-Linear Model

This is a 2-layer neural network

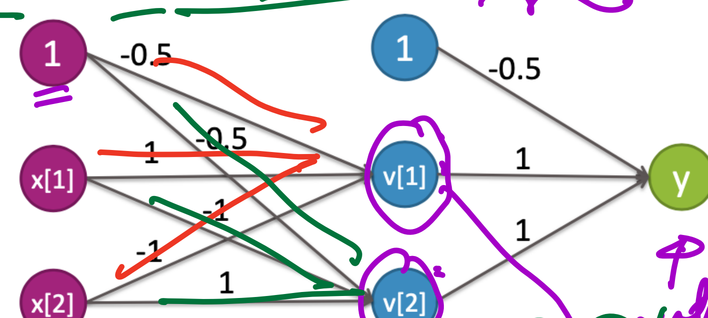
v_1 v_2 → Linear model

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

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

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

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



Input

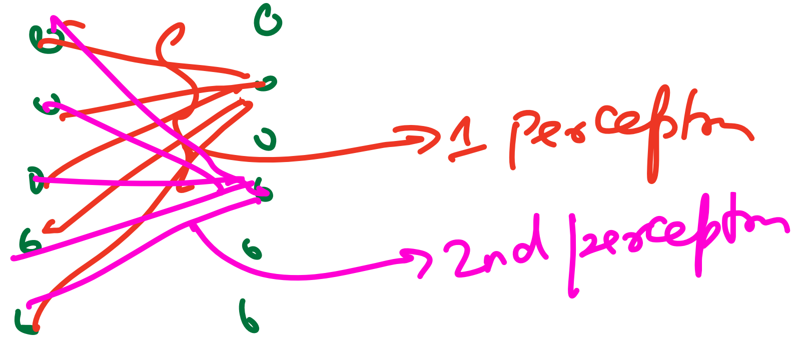
Hidden

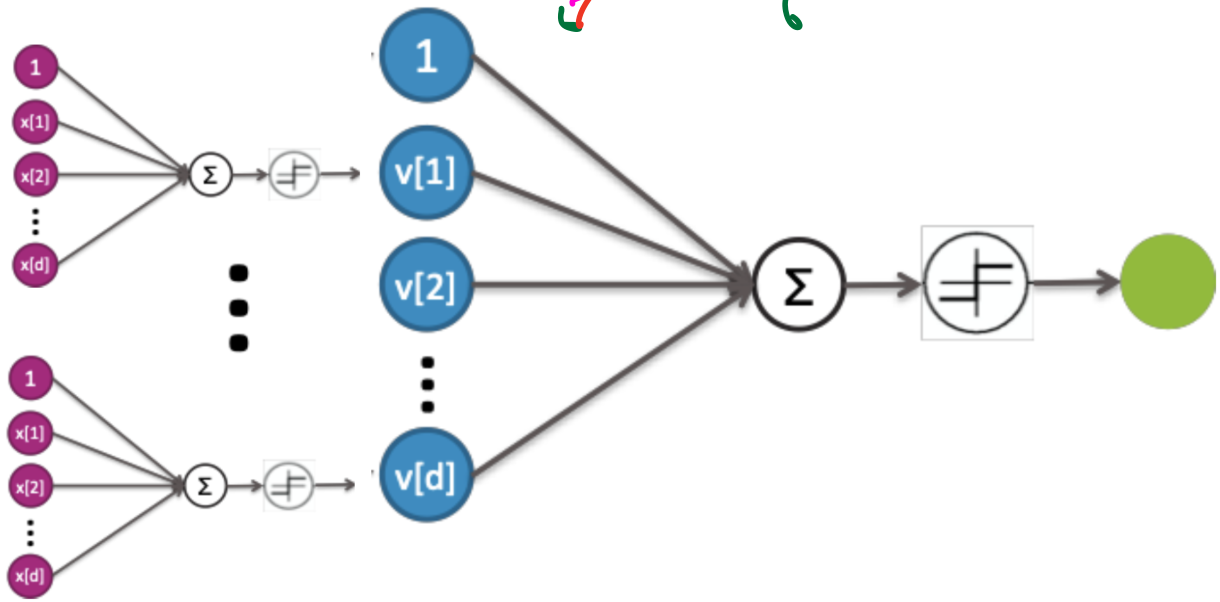
output layer

Non-Linear Combination of input x!)

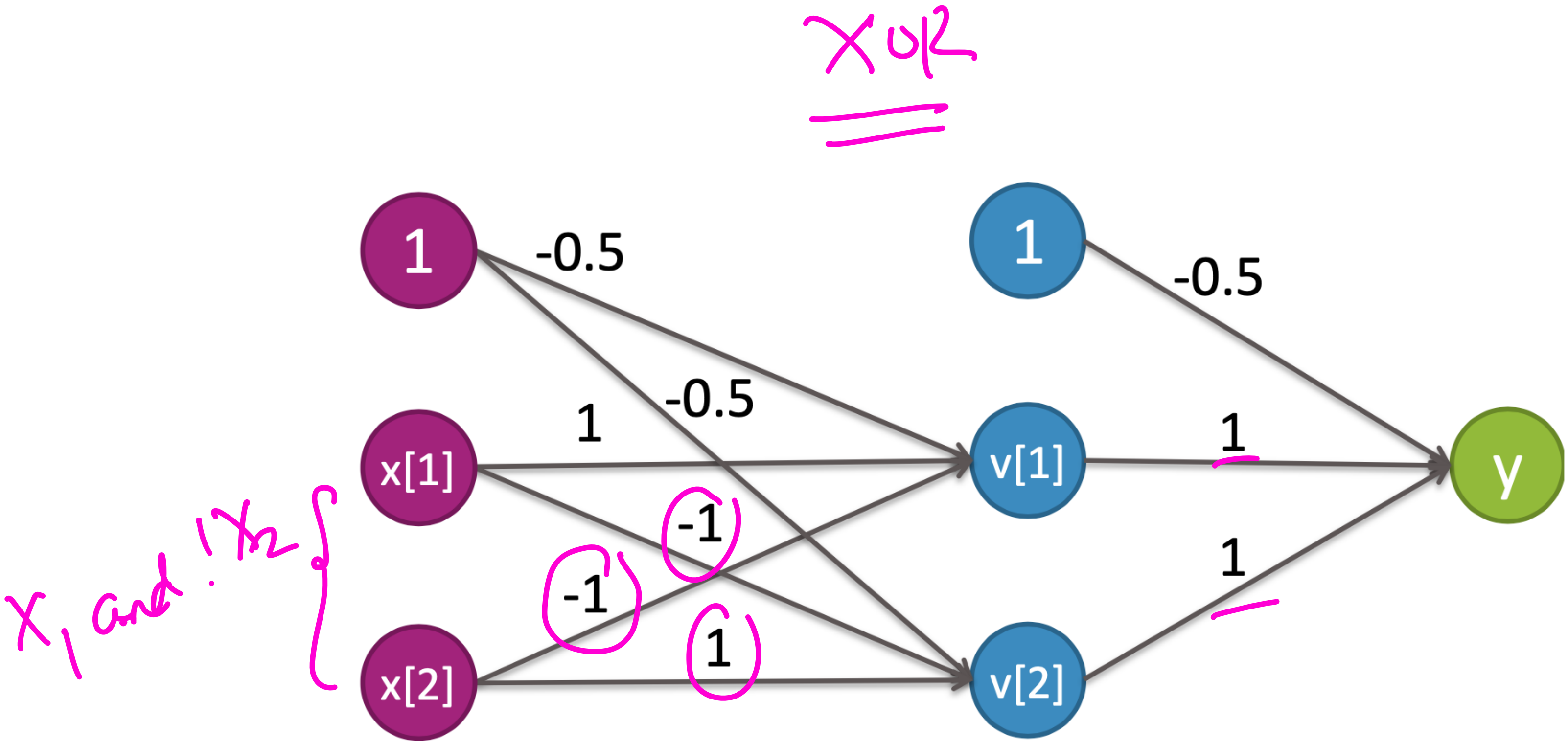
2 layer NN
1 hidden layer NN

Multi-Layer Perceptron (MLP)



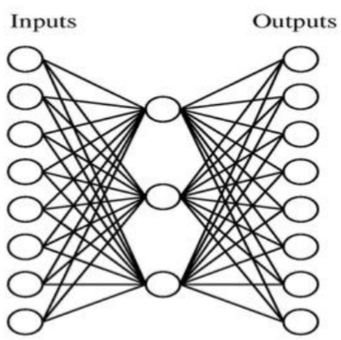


Multi-Layer Perceptron (MLP)



2 Layer Neural Network

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



$w^T x$

Single

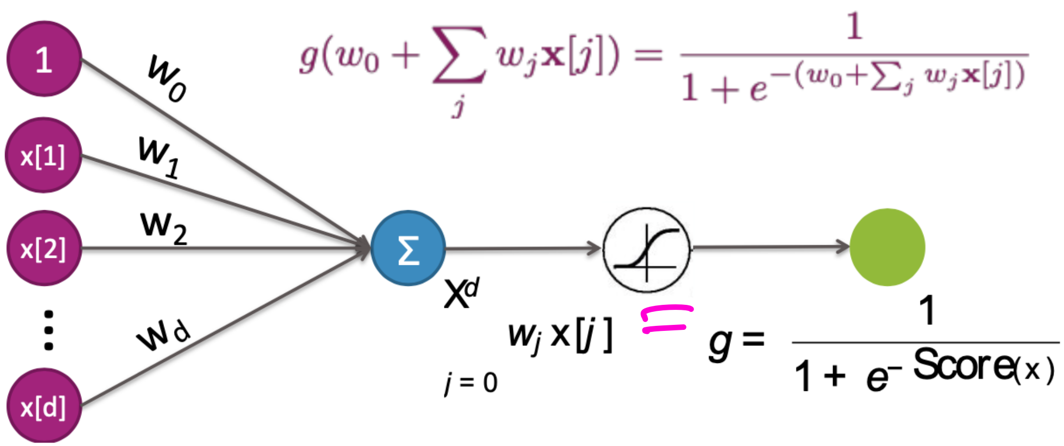
$$out(x) = g\left(w_0 + \sum_j w_j x[j]\right)$$

1-hidden layer

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

g – Activation function has to be non-linear

Perceptron to Logistic Regression



Choices for Non-Linear Activation Function

Probability

output

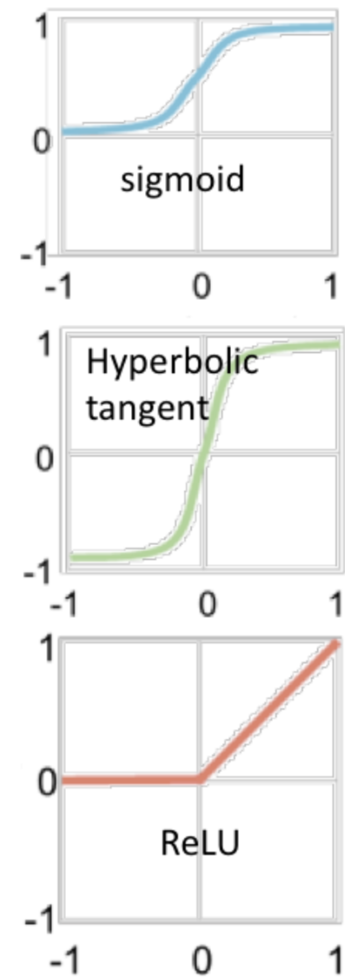
•Sigmoid
-Historically popular, but (mostly) fallen out of favor
•Neuron's activation saturates
(weights get very large -> gradients get small)
•Not zero-centered -> other issues in the gradient steps
-When put on the output layer, called "softmax" because
interpreted as class probability (soft assignment)

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

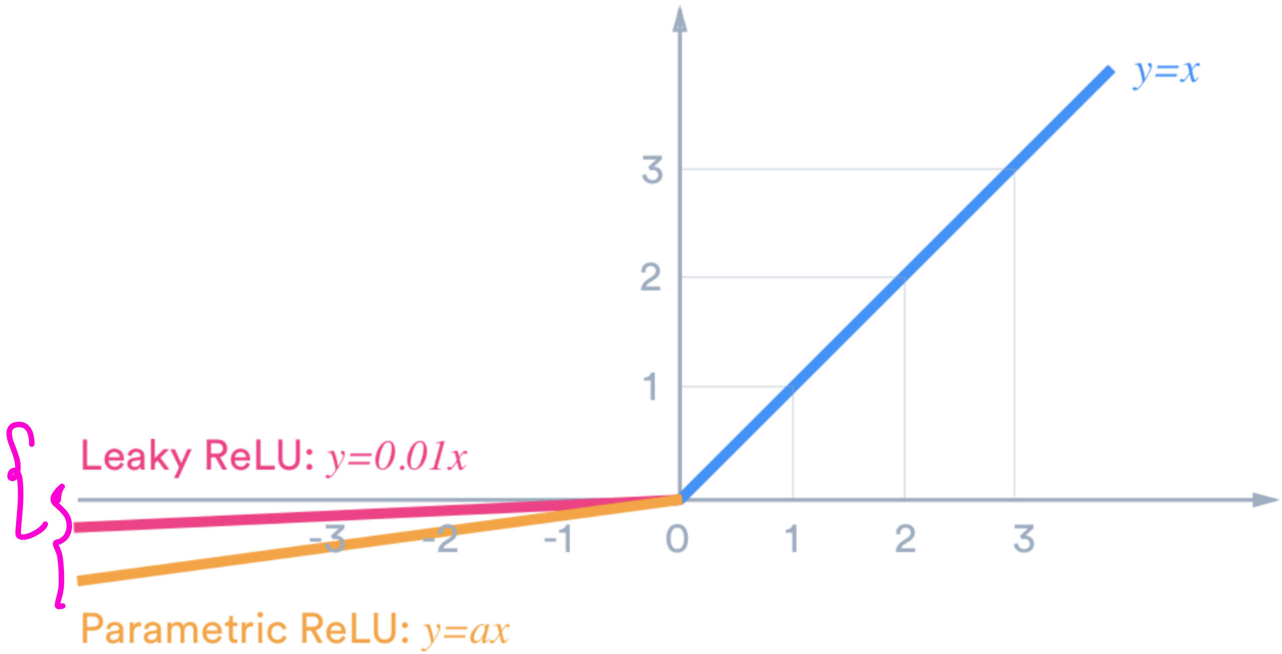
Hidden Layer

•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

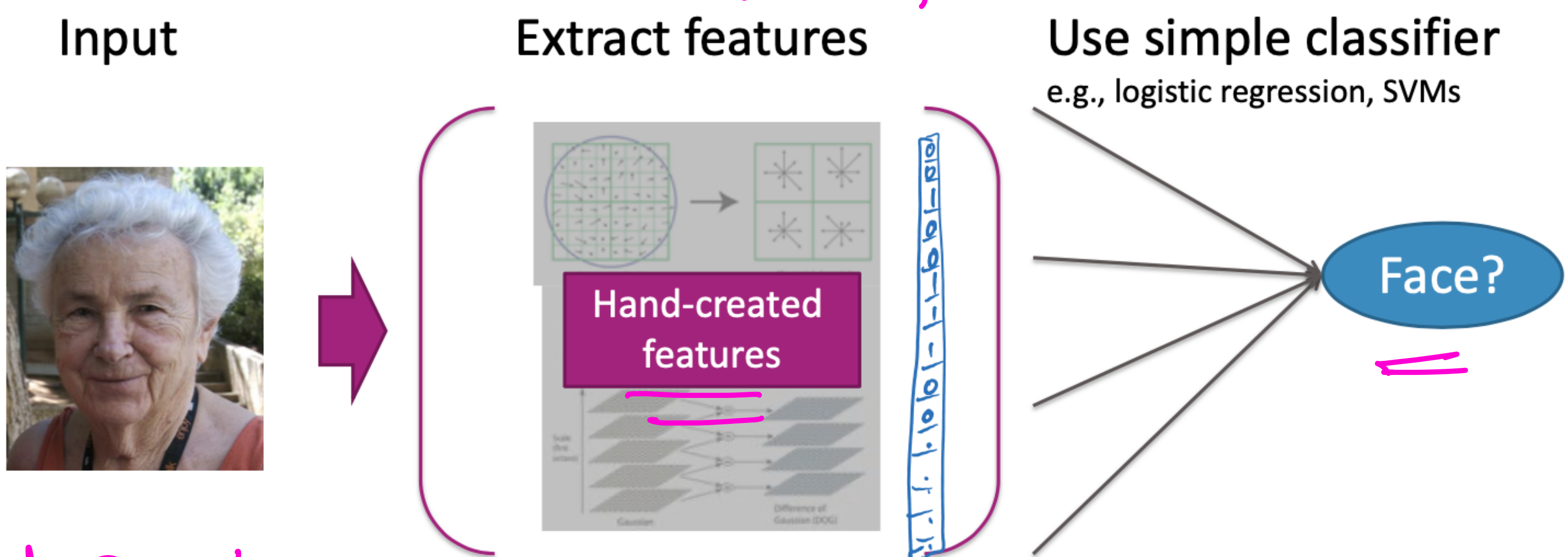


RELU vs Leaky RELU



Computer vision before deep learning

Feature Engineering



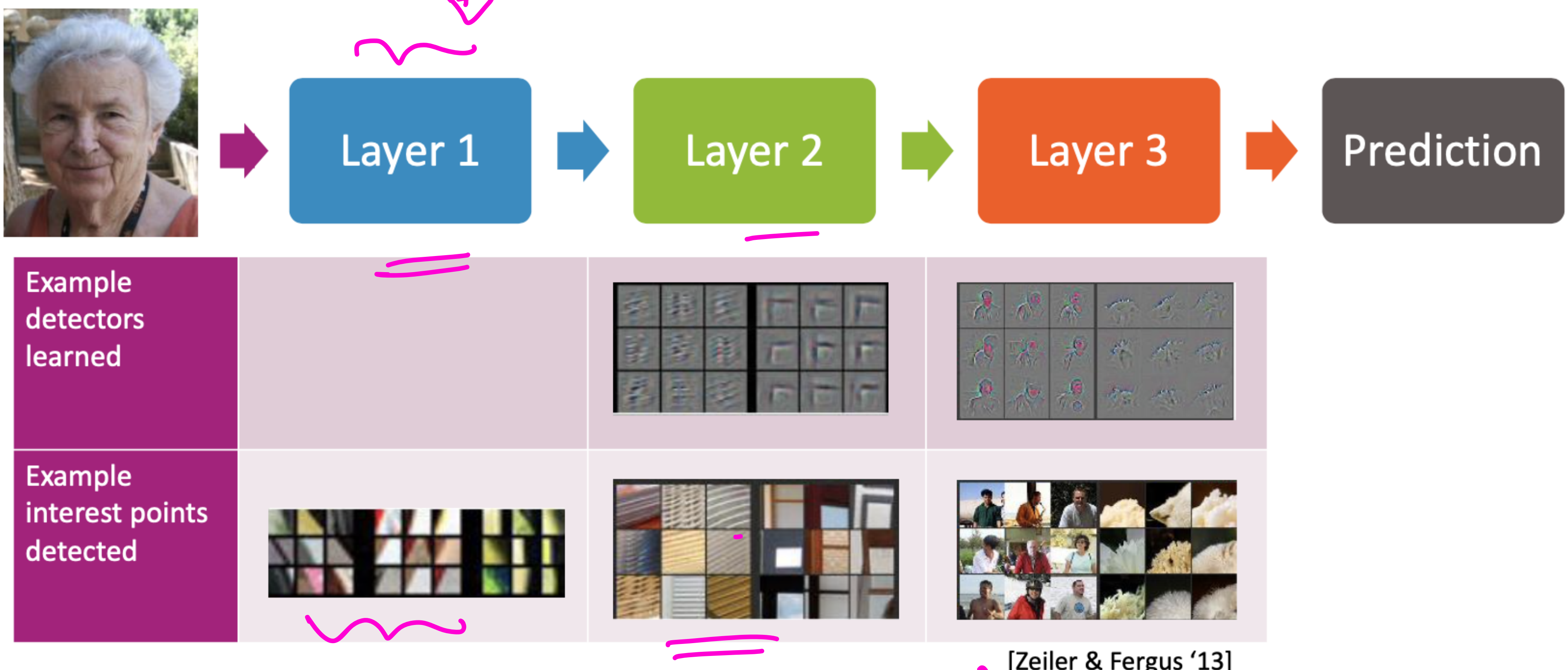
SIFT
Convolution

Feature vector

Explainability of AI | Interpretability

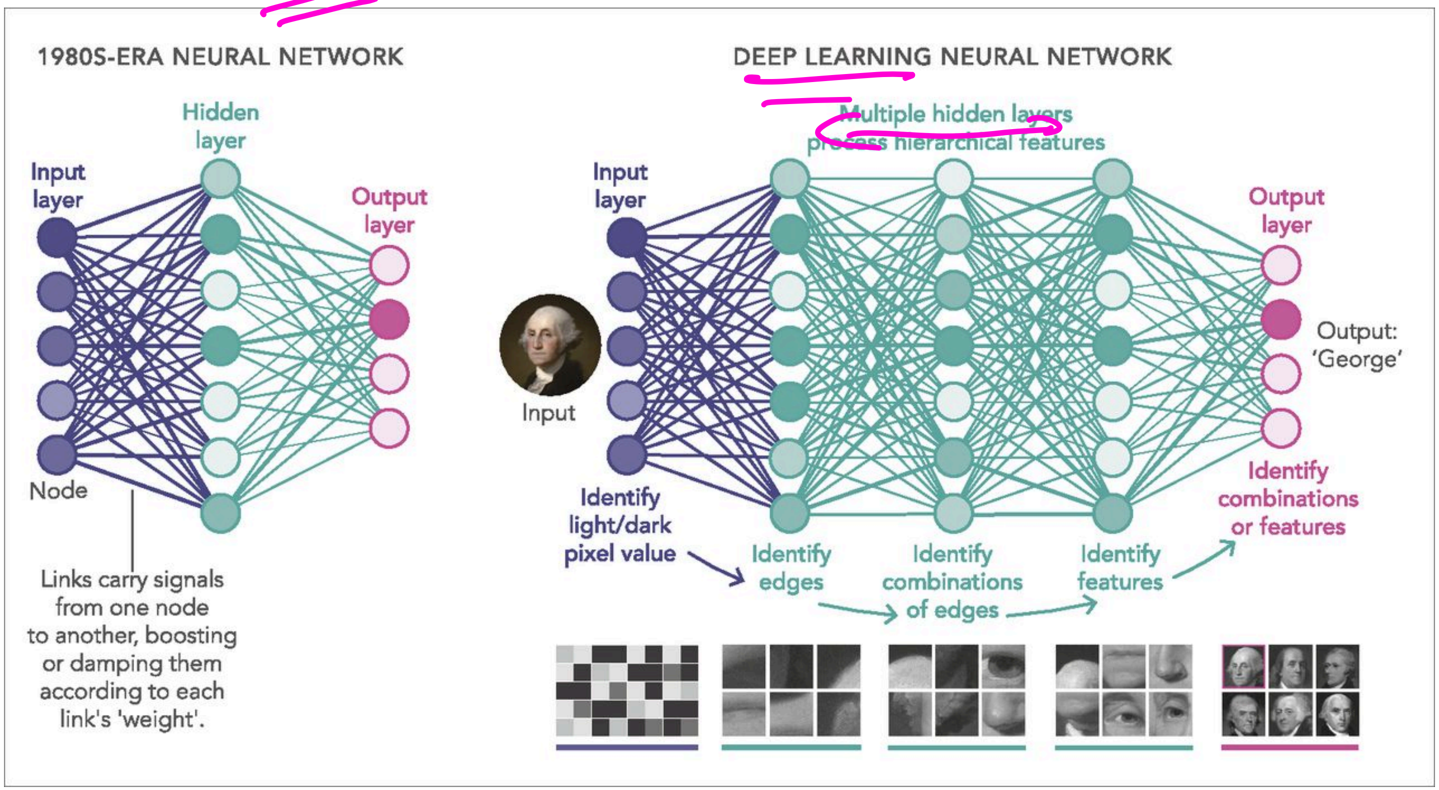
Computer vision after deep learning

Coarse Finer . . Higher



Feed-forward Deep Learning Architecture Example

1 Hidden
not DL



Feed-forward Deep Learning Architecture Example

tsne

512

100x100

500

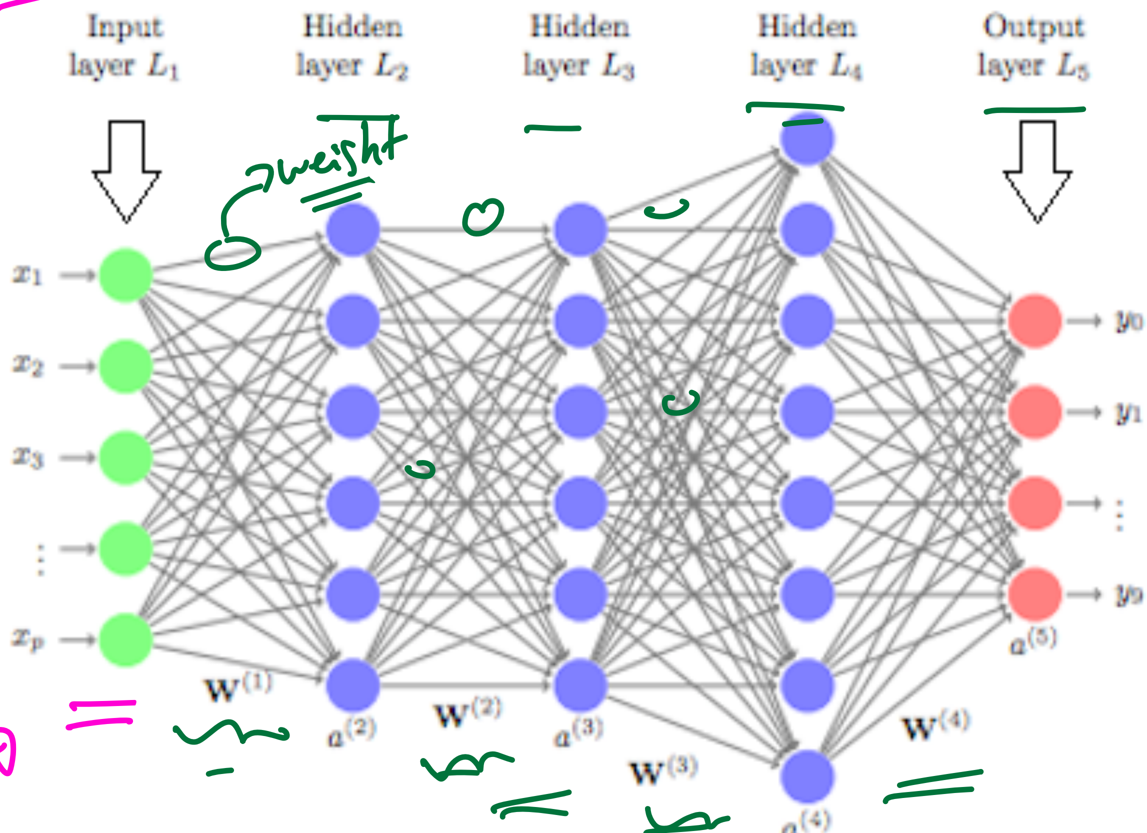
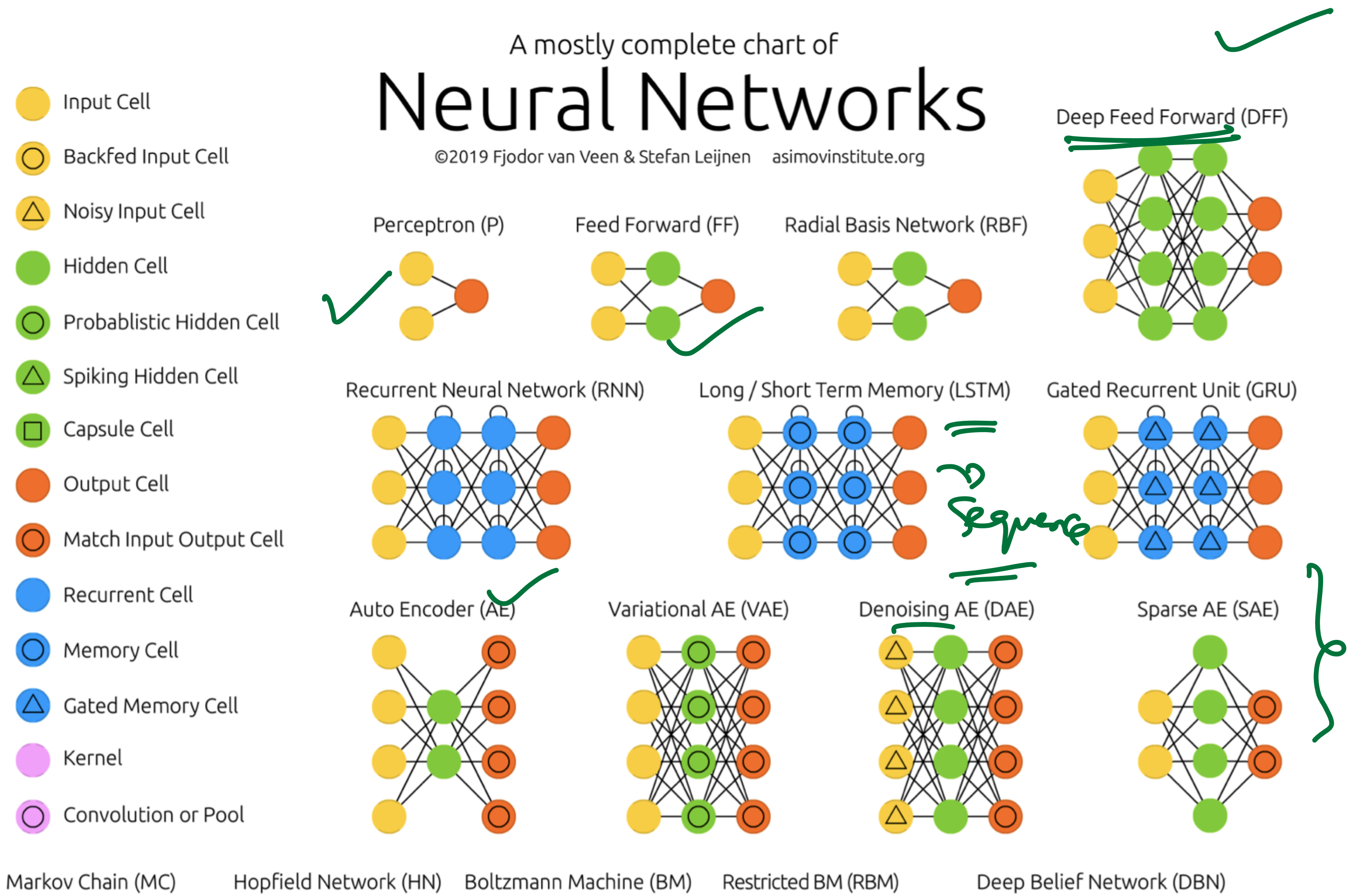


Image Embedding

1. Train Faster
2. Fewer parameters
3. Avoid overfitting

Other Neural Network Architectures

Neural Networks Zoo



Hyper-parameters in Deep Learning

ICE #2: 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. None of the above
5. 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

500 1000

η δ

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

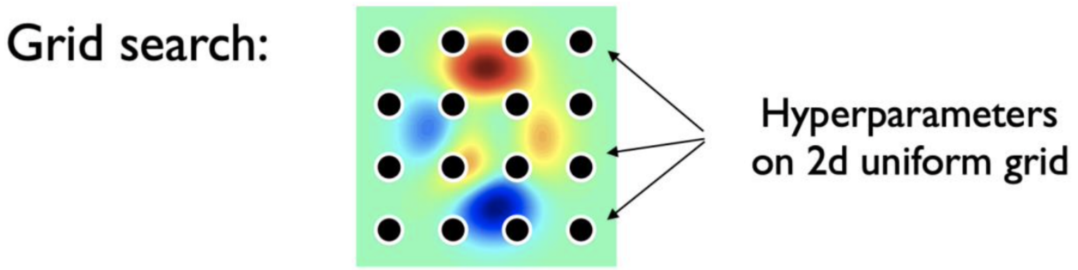
Hyper-parameters in Deep Learning

Hyper-parameters

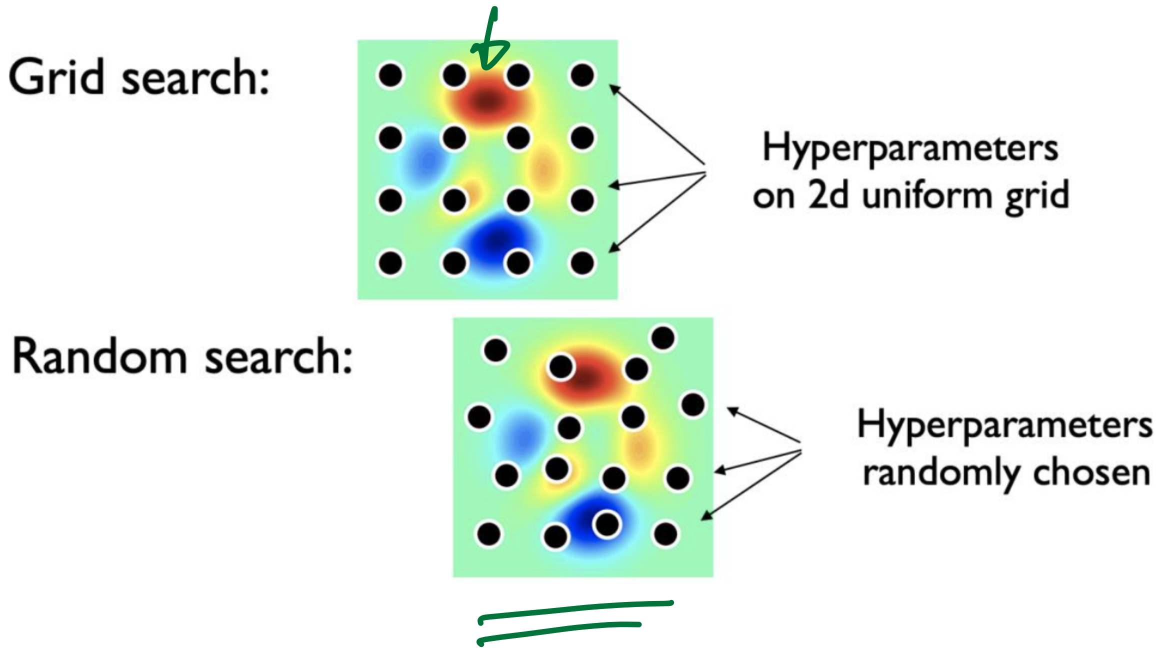
1. Learning rate
2. Number of Hidden Layers
3. Number of neurons per hidden layer
4. Type of non-linear activation function used
5. Anything else?

3 typical HPs

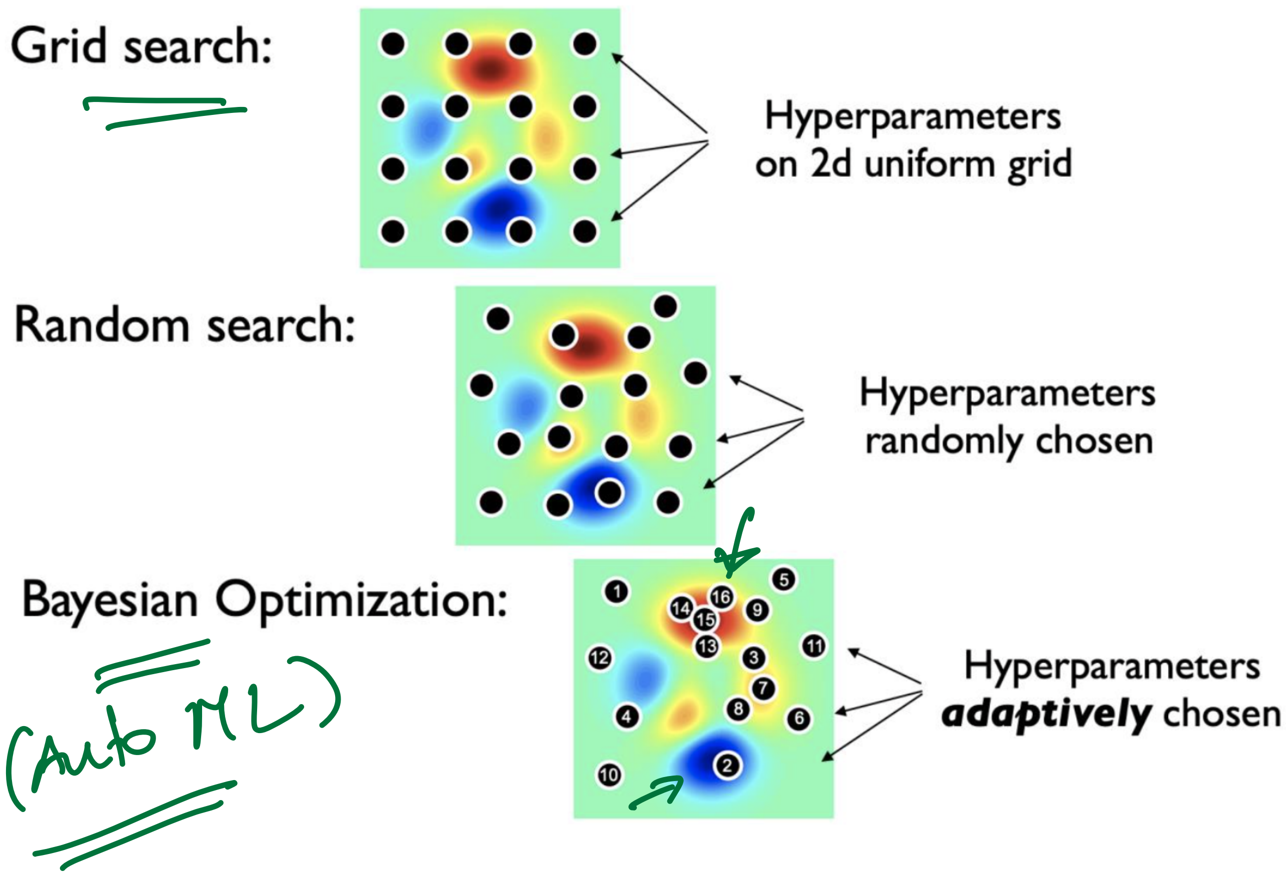
Hyper-parameter tuning methods



Hyper-parameter tuning methods



Hyper-parameter tuning methods



ICE #3

Compute the number of parameters in DNN model

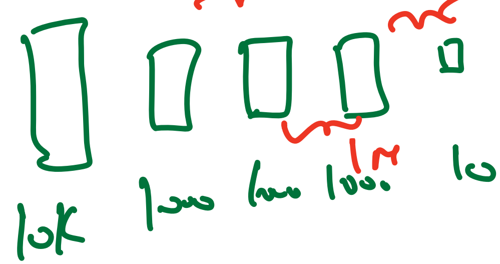
Consider a DNN model with 3 hidden layers where each hidden layer has 1000 neurons. Let the input layer be raw pixels from a 100x100 image and the output layer has 10 dimensions, let's say for a 10 class image classification example. How many total parameters exist in the DNN model?

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

$W^1 \rightarrow 10k \times 1000 = 10MM$

$1k \times 1k = 1M$

$\rightarrow 10k$



$10MM + 1MM + 1MM + 10k \approx 12MM?$

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

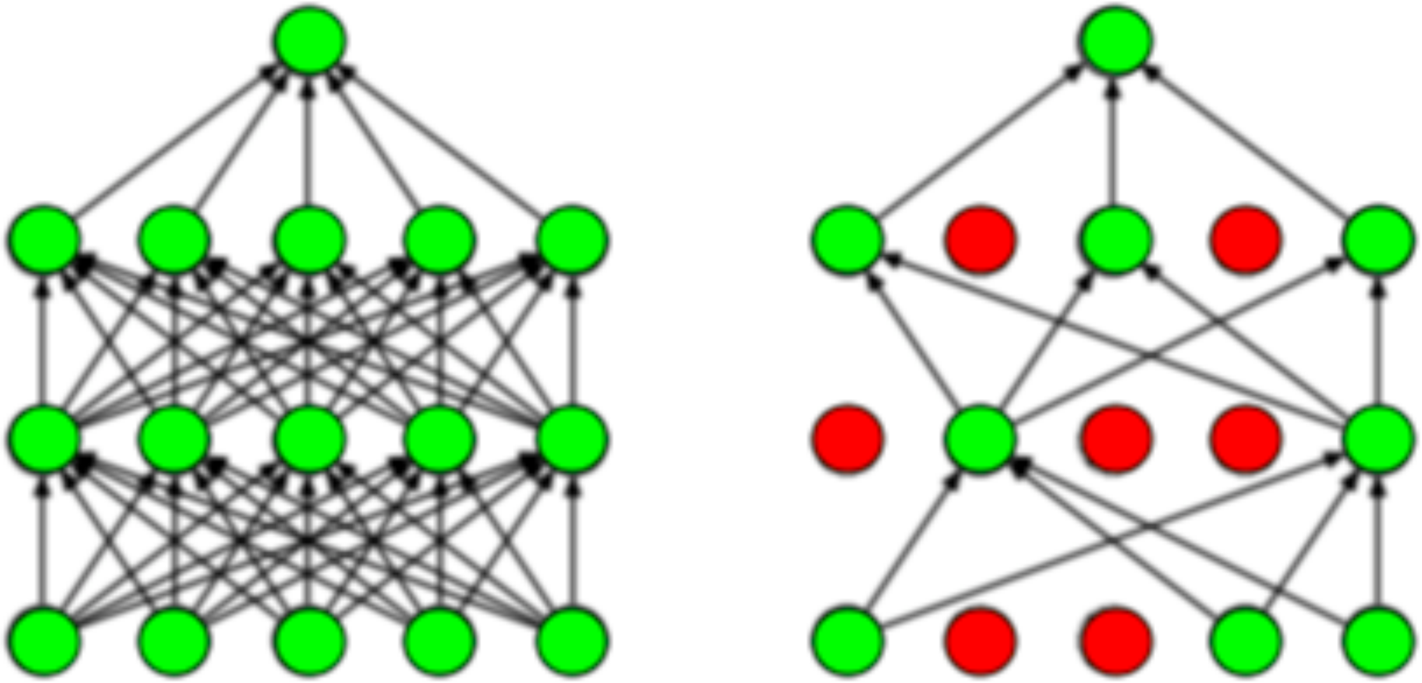
1. A DNN model with 100 million parameters and only 100k data points or even a million data points will overfit unless we take care of over-fitting.
2. Weight regularization can help - ℓ_1, ℓ_2
3. More common over-fitting strategy for DL?
4. Dropouts!
5. Early stopping is also a great strategy! Stop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

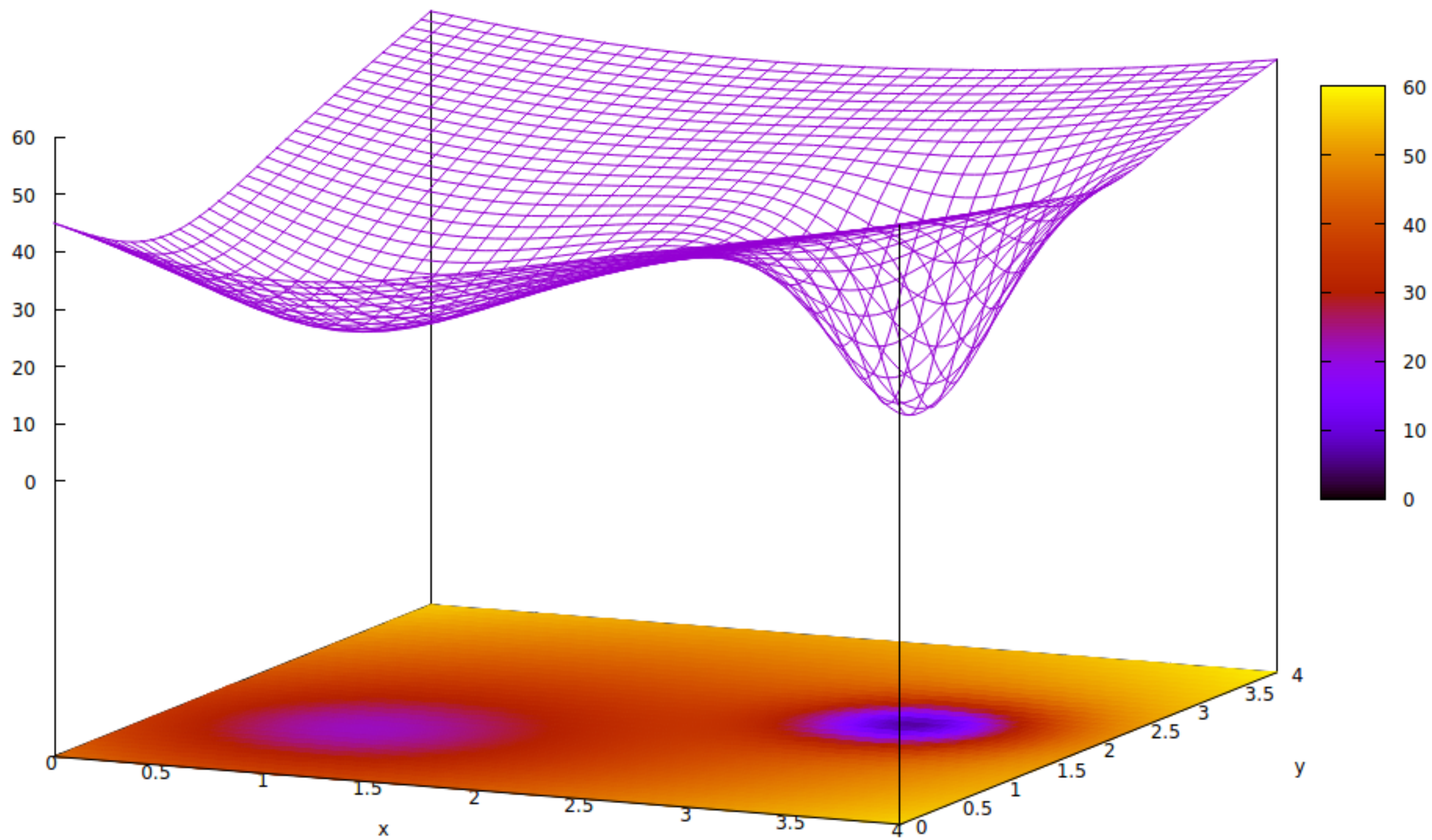
Taking care of Over-fitting: Dropouts



(a) Standard Neural Net

(b) After applying dropout.

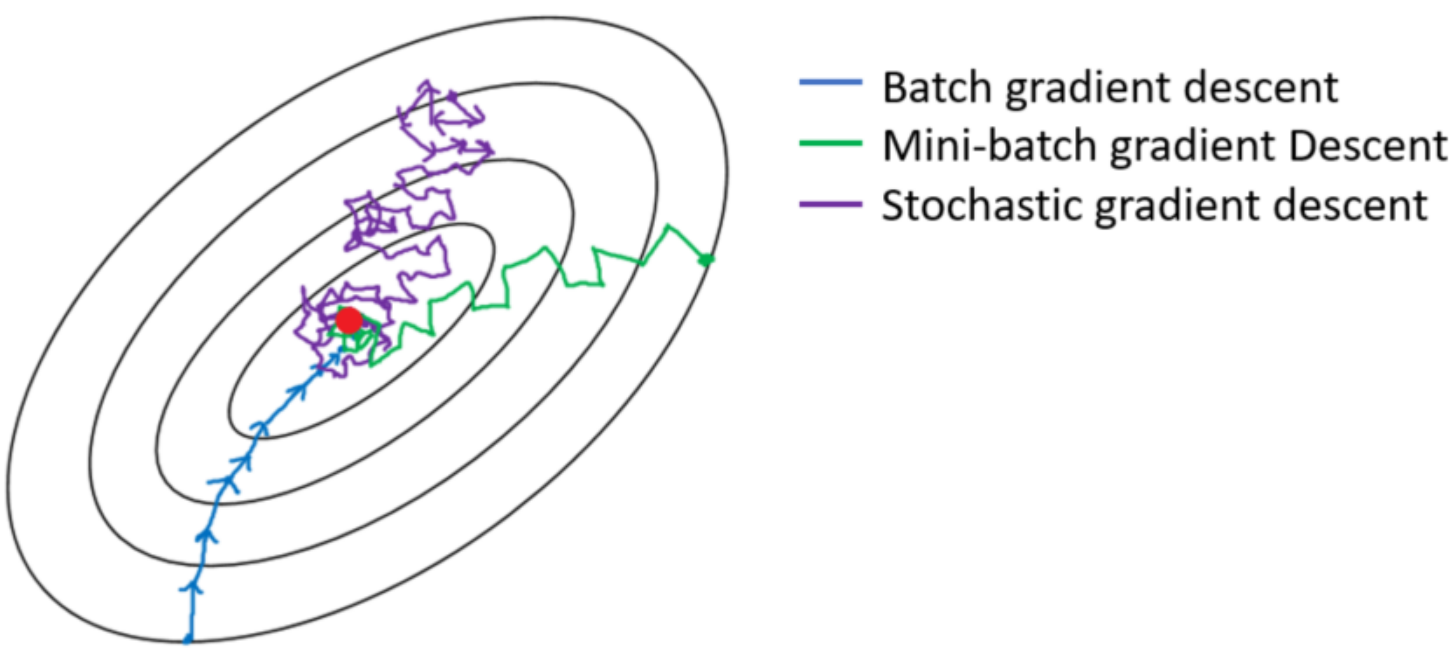
Good vs Bad Local minima



Algorithmic foundations to Machine Learning

Underlying Engine behind ML Training

`(Mini-batch) Stochastic Gradient Descent` Almost every model and problem-space in ML uses SGD of some kind - Clustering, Regression, Deep Learning, Computer Vision and NLP to name a few. Almost every algorithm in every library - `Scikit-learn`, `Keras`, `Pytorch`, etc uses **mini-batch SGD under the hood**.



So what is Gradient Descent?

Fundamentally

Take a convex/non-convex function, f. GD allows you to find a local optimum to f.

So what is Gradient Descent?

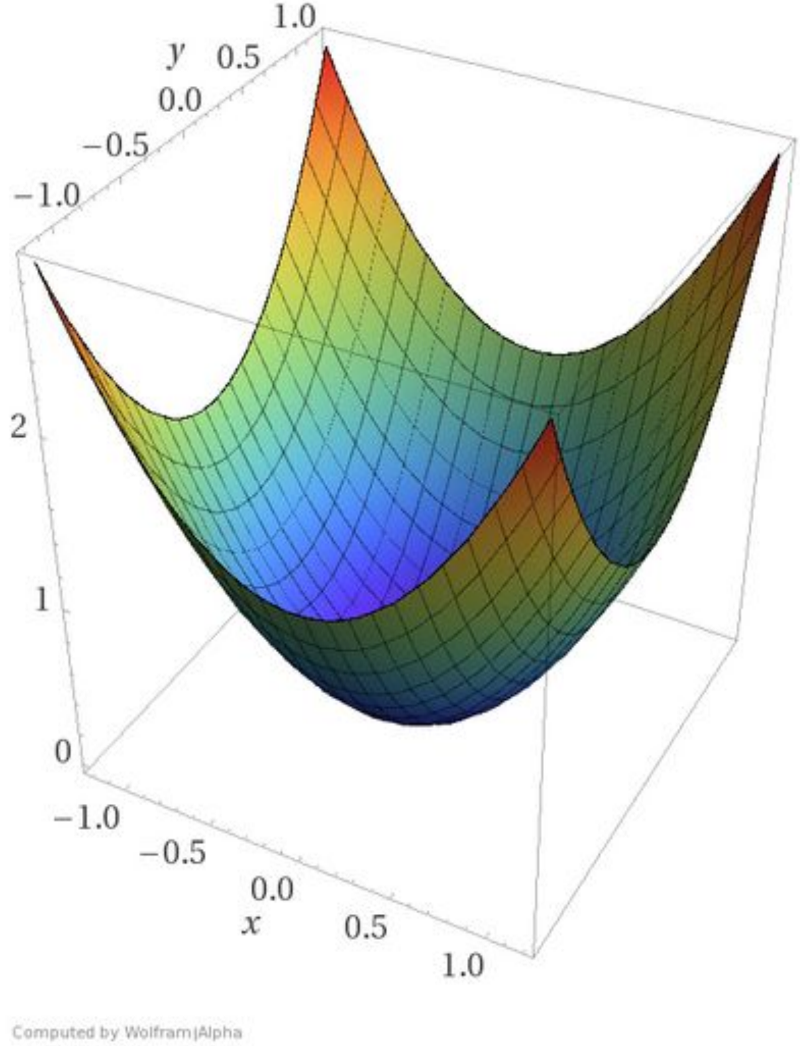
Fundamentally

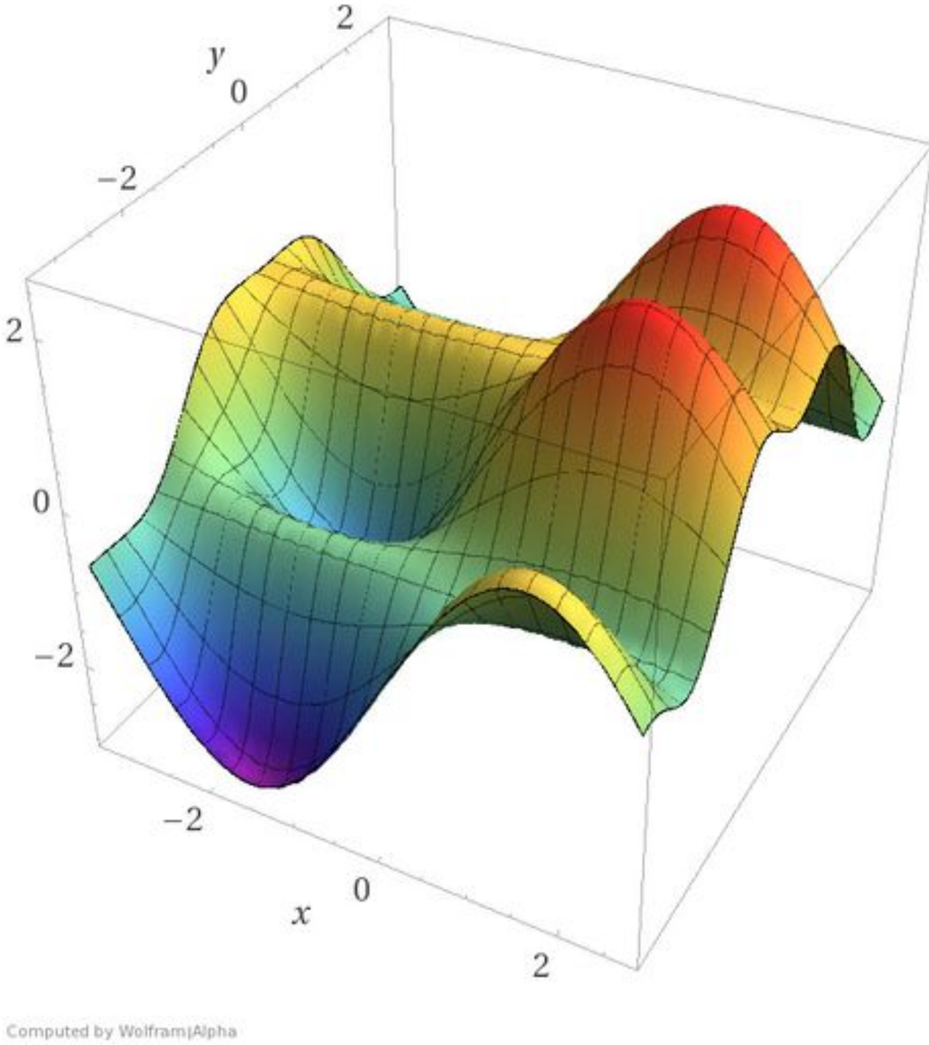
Take a convex/non-convex function, f. GD allows you to find a local optimum to f.

Why is this important?

Consider the Linear Regression problem. $\hat{w}$ is a local optimum to the function $f(w) = \frac{1}{2}\|Xw - y\|_2^2 + \lambda\|w\|_2^2$

Negative Gradient helps you view the direction of descent





Gradient Descent

Batch Gradient Descent

Let us say we want to minimize $L(w)$ - Loss Function and find the best $\hat{w}$ that does that.

1. **Initialize** $w = w_0$ (maybe randomize)

Gradient Descent

Batch Gradient Descent

Let us say we want to minimize $L(w)$ - Loss Function and find the best $\hat{w}$ that does that.

1. **Initialize** $w = w_0$ (maybe randomize)
2. **Gradient Descent** $w \leftarrow w - lr * \nabla L(w)$

Gradient Descent

Batch Gradient Descent

Let us say we want to minimize $L(w)$ - Loss Function and find the best $\hat{w}$ that does that.

1. **Initialize** $w = w_0$ (maybe randomize)
2. **Gradient Descent** $w \leftarrow w - lr * \nabla L(w)$
3. **Iterate** Repeat step 2 until w converges, i.e.

$$\|w^{k+1} - w^k\| / \|w^k\| \leq 10^{-3}$$

ICE #4

Derivative (1 min)

Find the derivative of w^2

- a) $2w$
- b) w
- c) $0.5w$
- d) 0

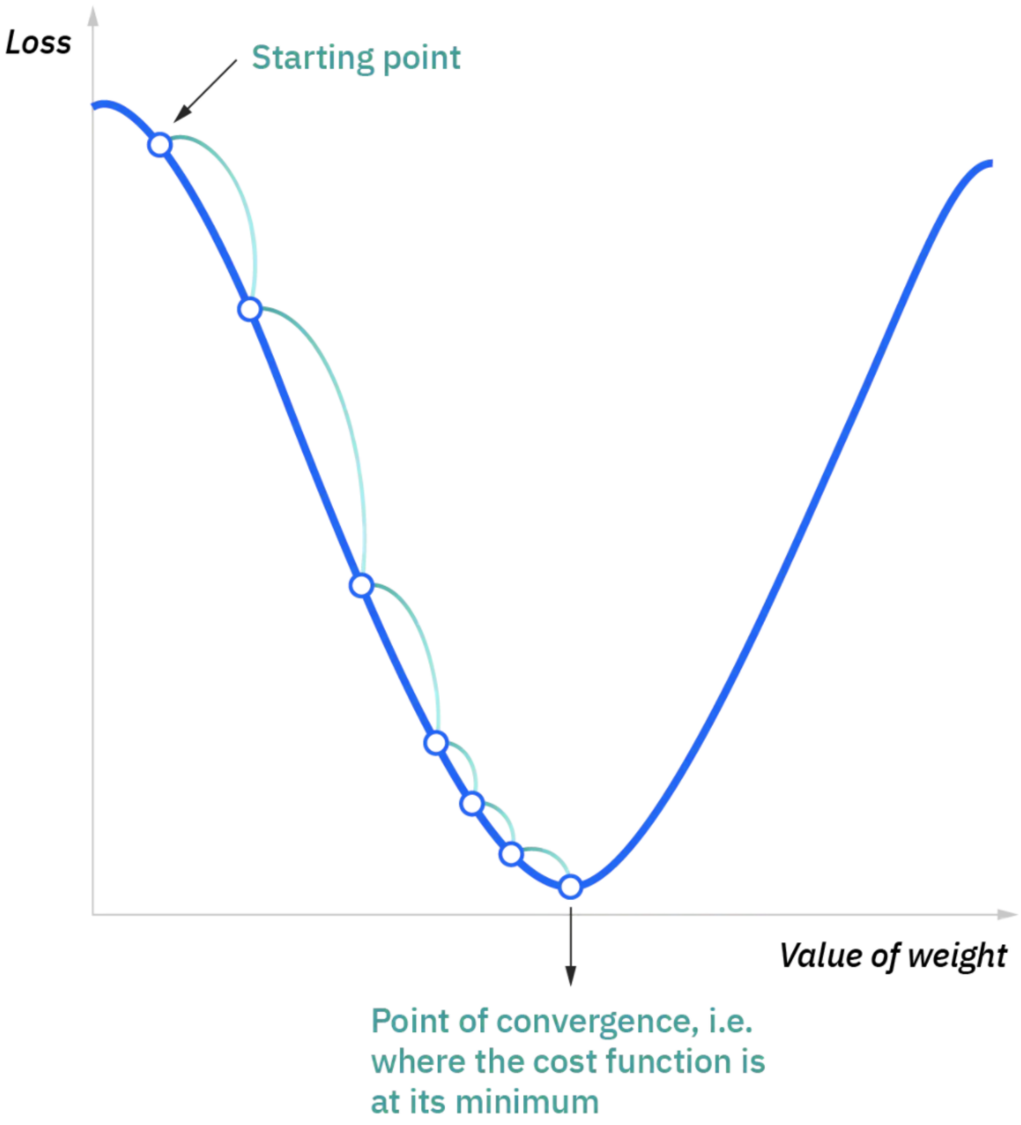
ICE #5

Gradient of Ridge Regularizer (2 mins)

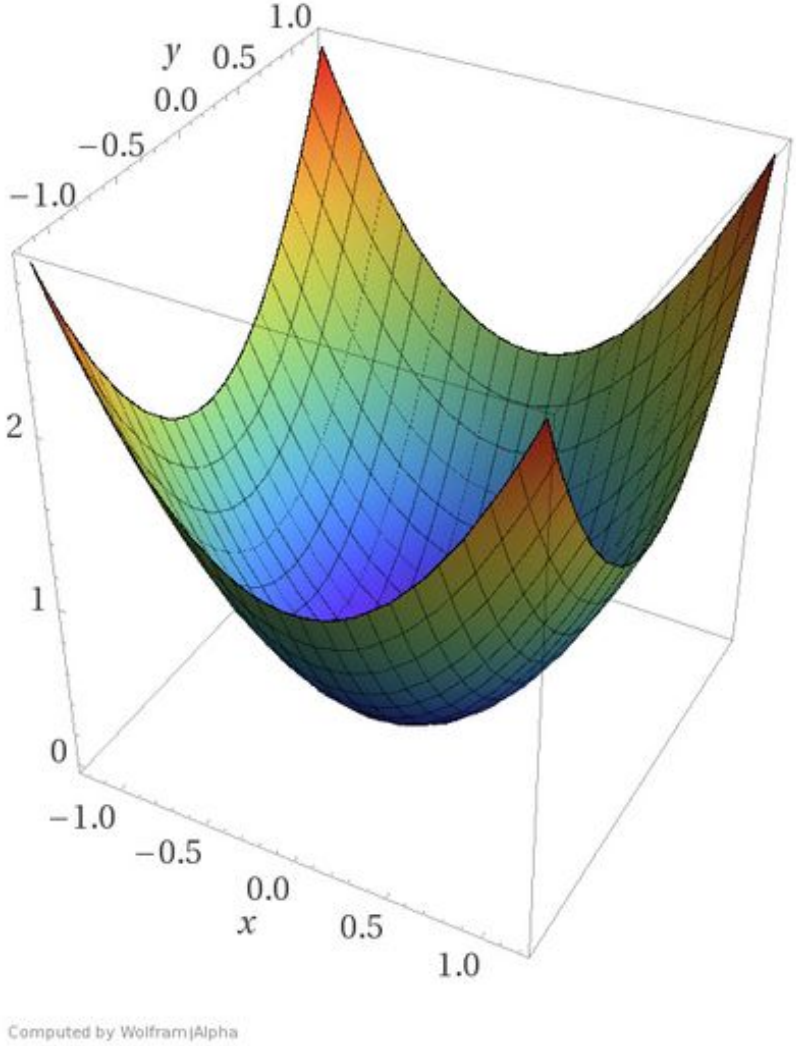
Find the gradient of the regularization function, $R(w) = \lambda \|w\|_2^2$. I.e. obtain the expression for, $\nabla_w R(w)$?

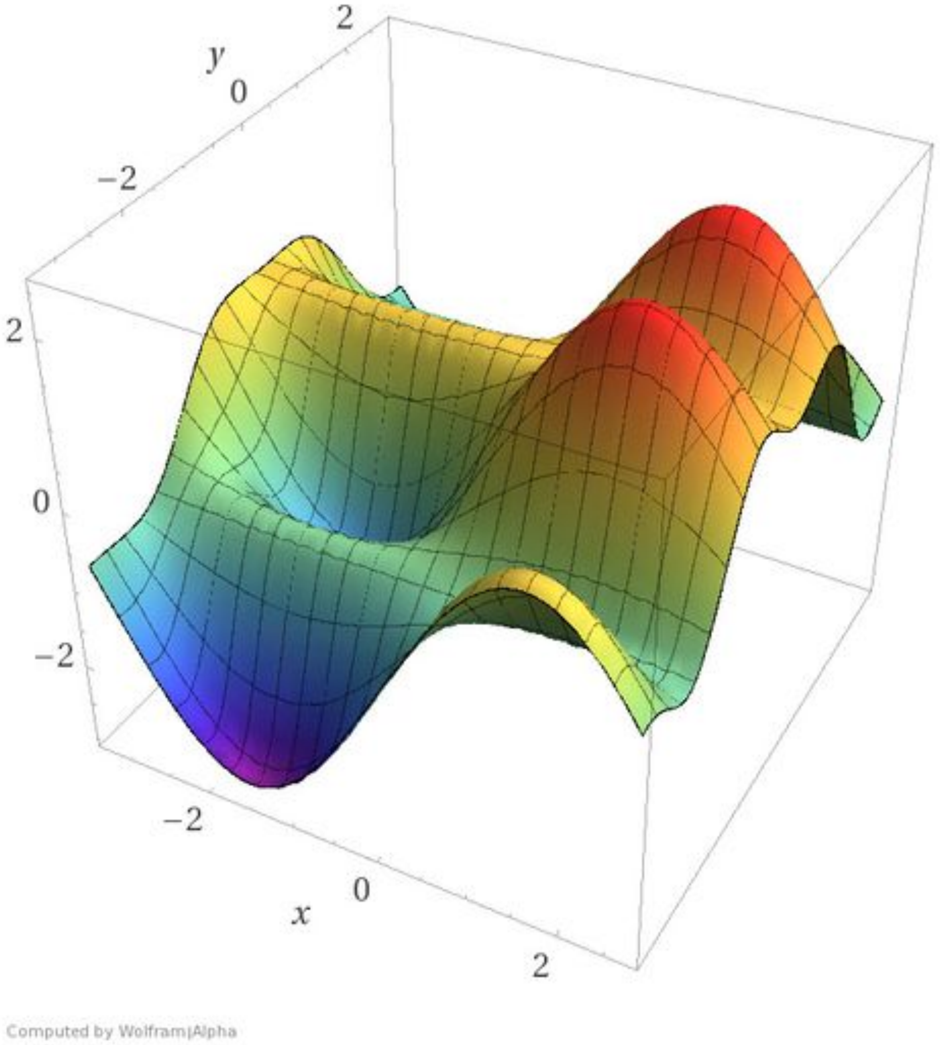
- a) $2\lambda\|w\|_2$
- b) $\lambda\|w\|_2 w$
- c) $2\lambda w$
- d) $2\lambda\|w\|_2 w$

GD in one dimension



Loss function in 2 dimensions





Gradient Descent Properties

1. Gradient Descent converges to a local minimum

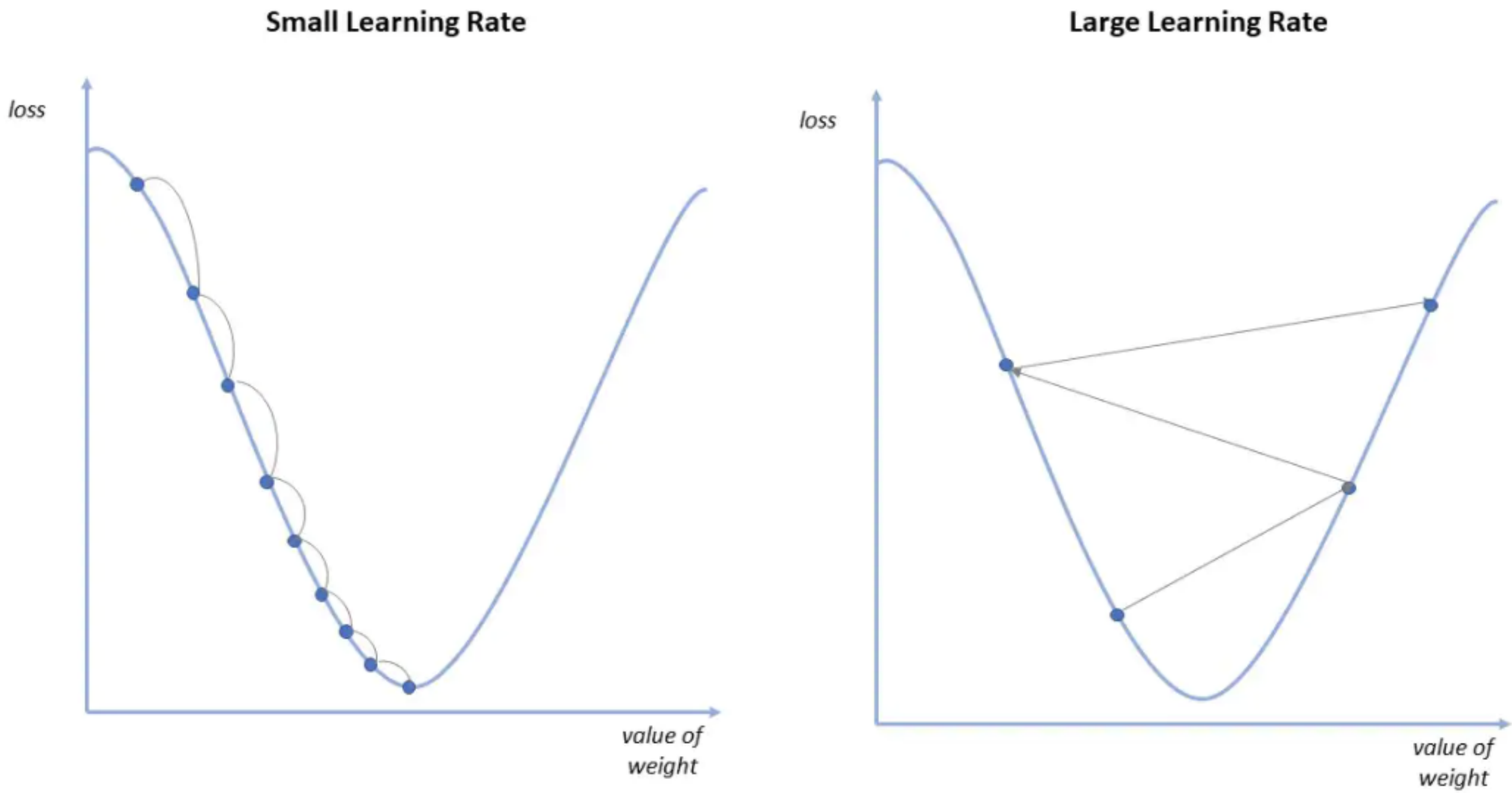
Gradient Descent Properties

1. Gradient Descent converges to a local minimum
2. If L is a convex function, all local minima become a global minima!

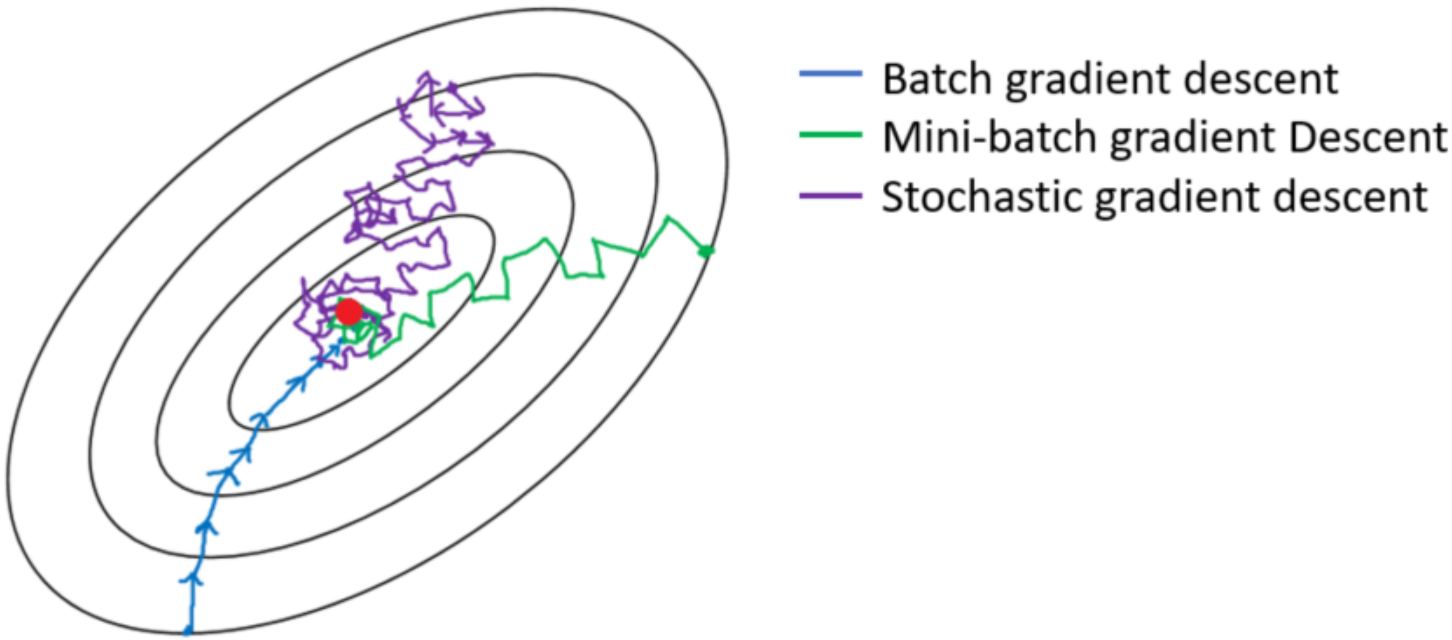
Gradient Descent Properties

1. Gradient Descent converges to a local minimum
2. If L is a convex function, all local minima become a global minima!
3. Wherever we start, gradient descent usually finds a local minima closest to the start.

Effect of Learning Rate



SGD behavior in search space



SGD in practice - mini-batch SGD!

mini-batch SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w. Let B be the number of batches and k be the batch size.

1. **Initialize** $w = w_0$ (randomize)

SGD in practice - mini-batch SGD!

mini-batch SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w. Let B be the number of batches and k be the batch size.

1. **Initialize** $w = w_0$ (randomize) Pick a batch of k data points at random between 1 and N: $i_1, i_2, \ldots, i_k$!

SGD in practice - mini-batch SGD!

mini-batch SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w. Let B be the number of batches and k be the batch size.

1. **Initialize** $w = w_0$ (randomize) Pick a batch of k data points at random between 1 and N: $i_1, i_2, \ldots, i_k$!
2. **Gradient Descent** $w^{k+1} \leftarrow w^k - lr * \sum_{j=1}^{k} \nabla_w L_{i_j}(w^k)$

SGD in practice - mini-batch SGD!

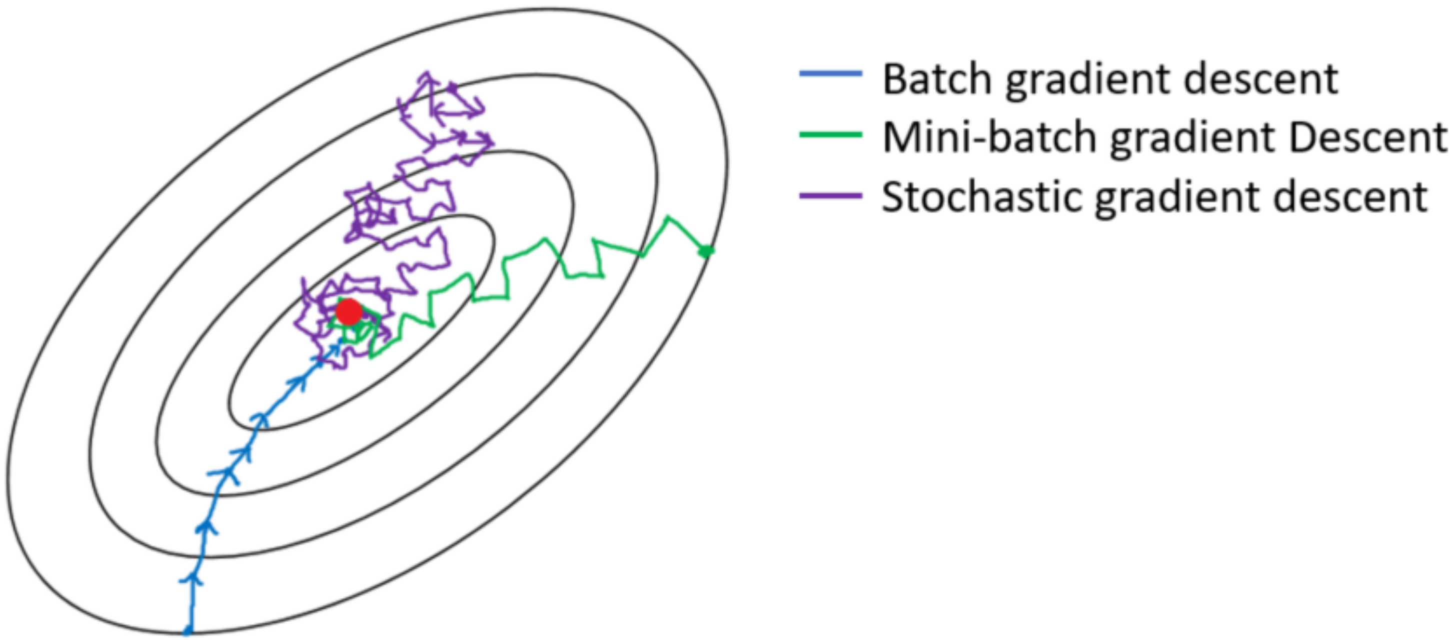
mini-batch SGD

Let $L(w) = \sum_{i=1}^{N} L_i(w)$ where L_i is a function of only the *ith* data point (x_i, y_i) and parameter w. Let B be the number of batches and k be the batch size.

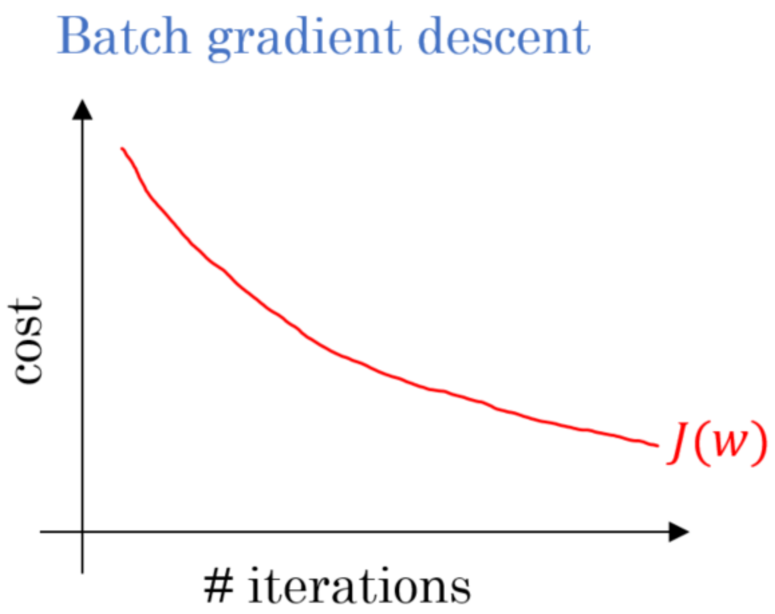
1. **Initialize** $w = w_0$ (randomize) Pick a batch of k data points at random between 1 and N: $i_1, i_2, \ldots, i_k$!
2. **Gradient Descent** $w^{k+1} \leftarrow w^k - lr * \sum_{j=1}^{k} \nabla_w L_{i_j}(w^k)$
3. **Iterate** Repeat step 2 and 3 until w converges, i.e.

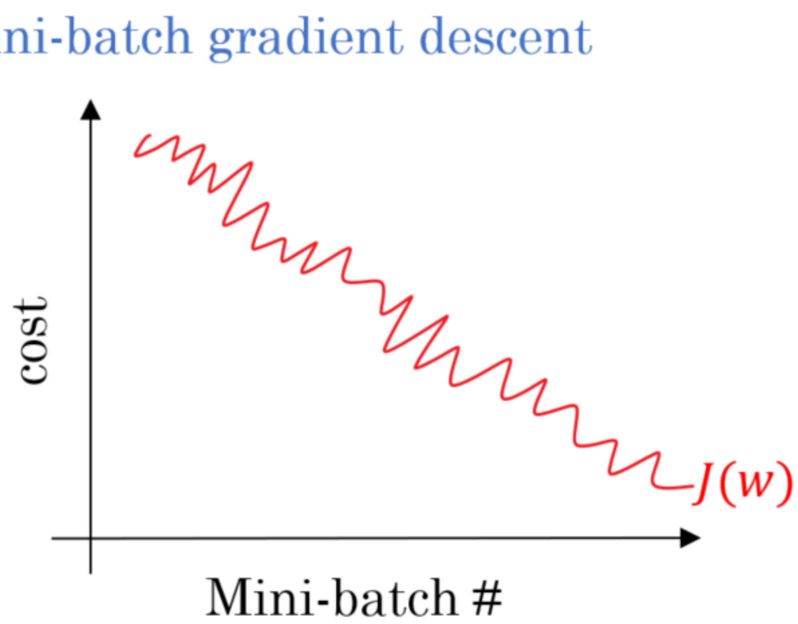
$$\|w^{k+1} - w^k\| / \|w^k\| \leq 10^{-3}$$

GD behavior in the search space



GD vs Mini-batch convergence behavior

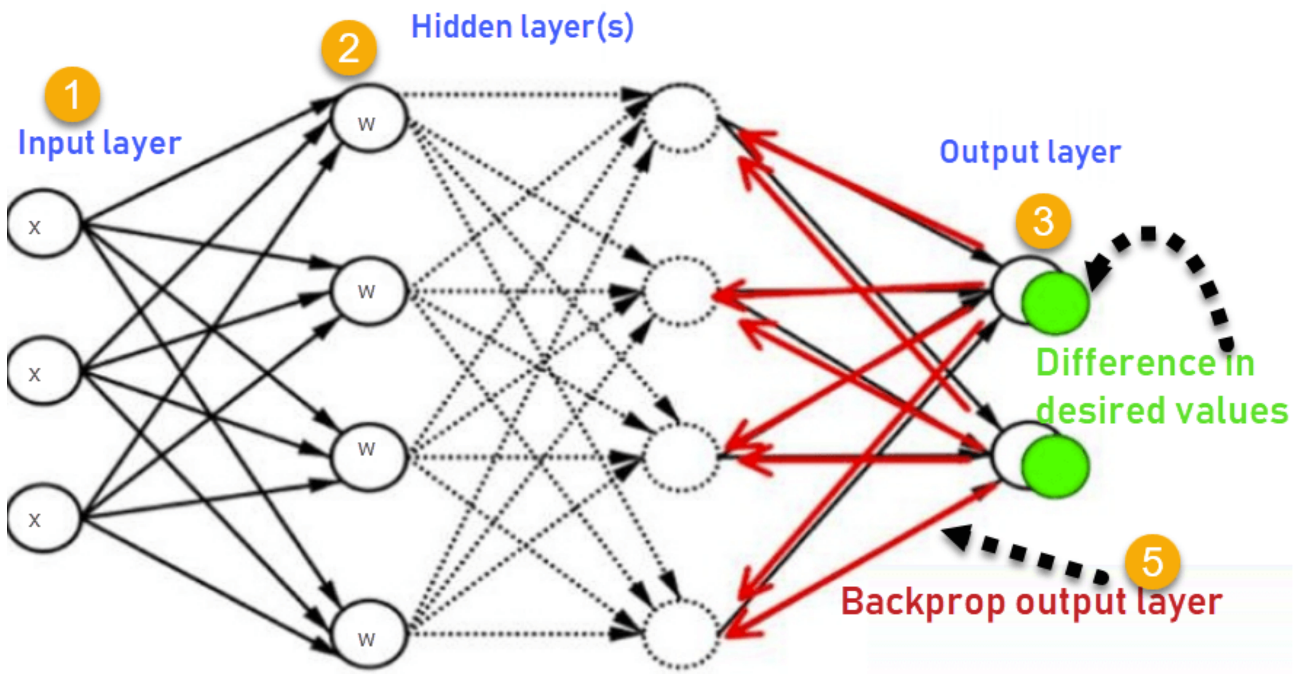




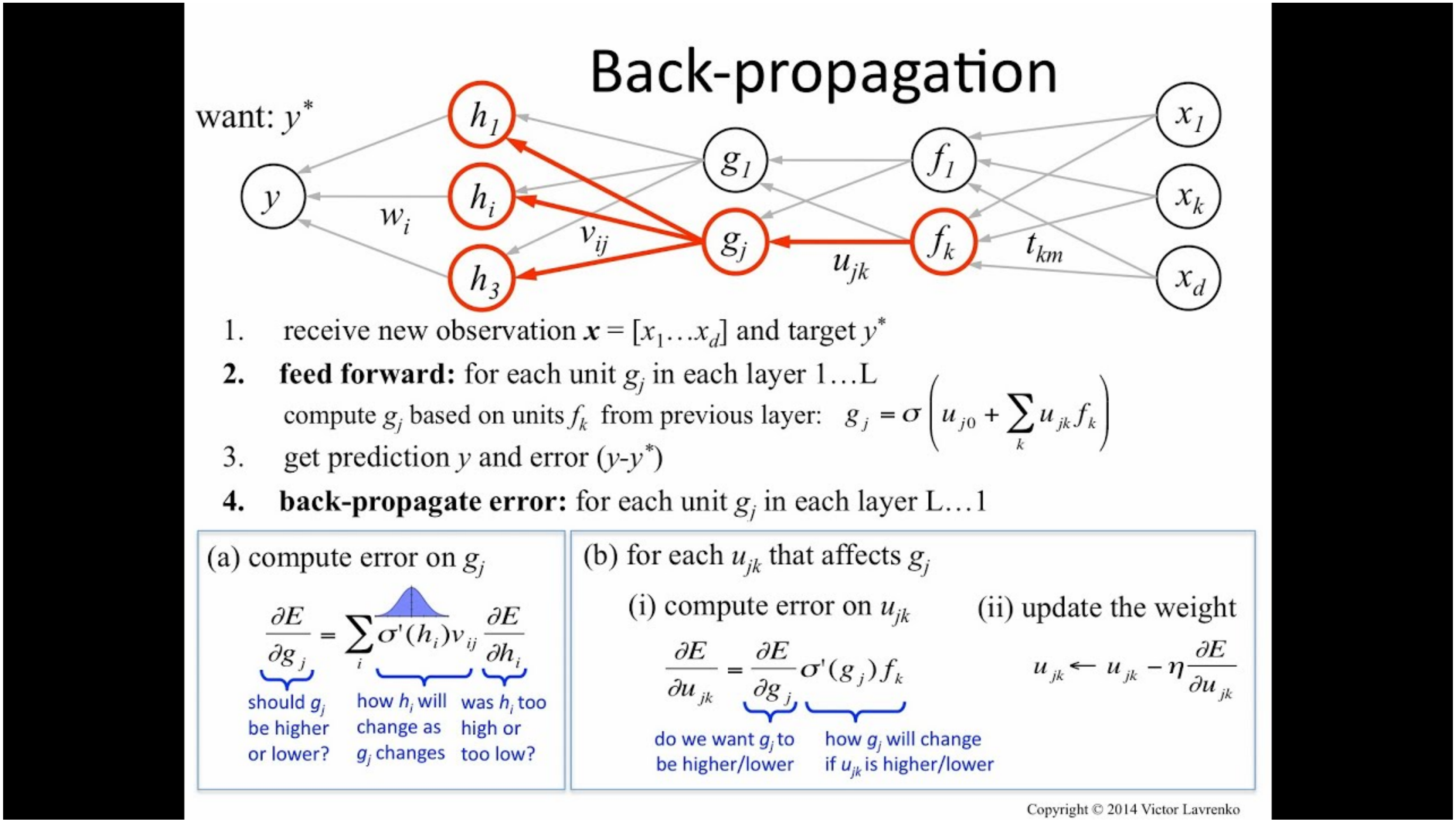
GD vs mini-batch SGD

Factor	GD	Mini-batch SGD
Data	All per iteration	Mini-batch (usually 128 or 256)
Randomness	Deterministic	Stochastic
Error reduction	Monotonic	Stochastic
Computation	High	Low
Memory big data	Intractable	Tractable
Convergence	Low relative error	Few "passes" on data
Local Minima traps	Yes	No

Forward Propagation vs Back-propagation in NN



Back Propagation explained



ICE #6

Back Prop

How is back-prop related to Gradient Descent?

1. Back-propagation is an alternative to Gradient Descent for Neural Networks
2. Back-propagation comptues the gradient that can then be used in gradient descent
3. Back-prop is the same as gradient descent for neural networks
4. Back-prop is a different concept from Gradient Descent

Mini-Project Pointers

1. Form a team of 2 (today if you can!) Share your team on the spreadsheet in discord

Mini-Project Pointers

1. Form a team of 2 (today if you can!) Share your team on the spreadsheet in discord
2. Play with the architecture details in your modeling process. Start simple and add more layers, more neurons per layer if it's help your validation metrics. Hyper-parameters are tuned on validation set

Mini-Project Pointers

1. Form a team of 2 (today if you can!) Share your team on the spreadsheet in discord
2. Play with the architecture details in your modeling process. Start simple and add more layers, more neurons per layer if it's help your validation metrics. Hyper-parameters are tuned on validation set
3. **Two Deadlines for Mini-Project:** First one is November 6th as a check-point with deliverables including baseline and your first NN model. Second includes full report, best Kaggle submission, all metrics and CNN model that is due November 13th.

Mini-Project Pointers

1. Form a team of 2 (today if you can!) Share your team on the spreadsheet in discord
2. Play with the architecture details in your modeling process. Start simple and add more layers, more neurons per layer if it's help your validation metrics. Hyper-parameters are tuned on validation set
3. **Two Deadlines for Mini-Project:** First one is November 6th as a check-point with deliverables including baseline and your first NN model. Second includes full report, best Kaggle submission, all metrics and CNN model that is due November 13th.
4. Will work with PyTorch for this Mini-Project - Get yourself familiar with tutorials on this (Will be covered in quiz section as well)

Summary

1. Introduction to Neural Networks
2. Neural Network Architecture and its components
3. Backpropagation in Neural Networks
4. Overfitting in Neural Networks