Computer Vision: Fall 2022 — Lecture 9

Dr. Karthik Mohan

Univ. of Washington, Seattle

October 28, 2022

Check-In

1. Mini Project 1 Assigned

Check-In

1. Mini Project 1 Assigned
2. Identify your team mate through the spreadsheet

Check-In

1. Mini Project 1 Assigned
2. Identify your team mate through the spreadsheet
3. Other thoughts/questions?

References

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

Today

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

} mechanics of DL

Introduction to Deep Learning

Deep Learning

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

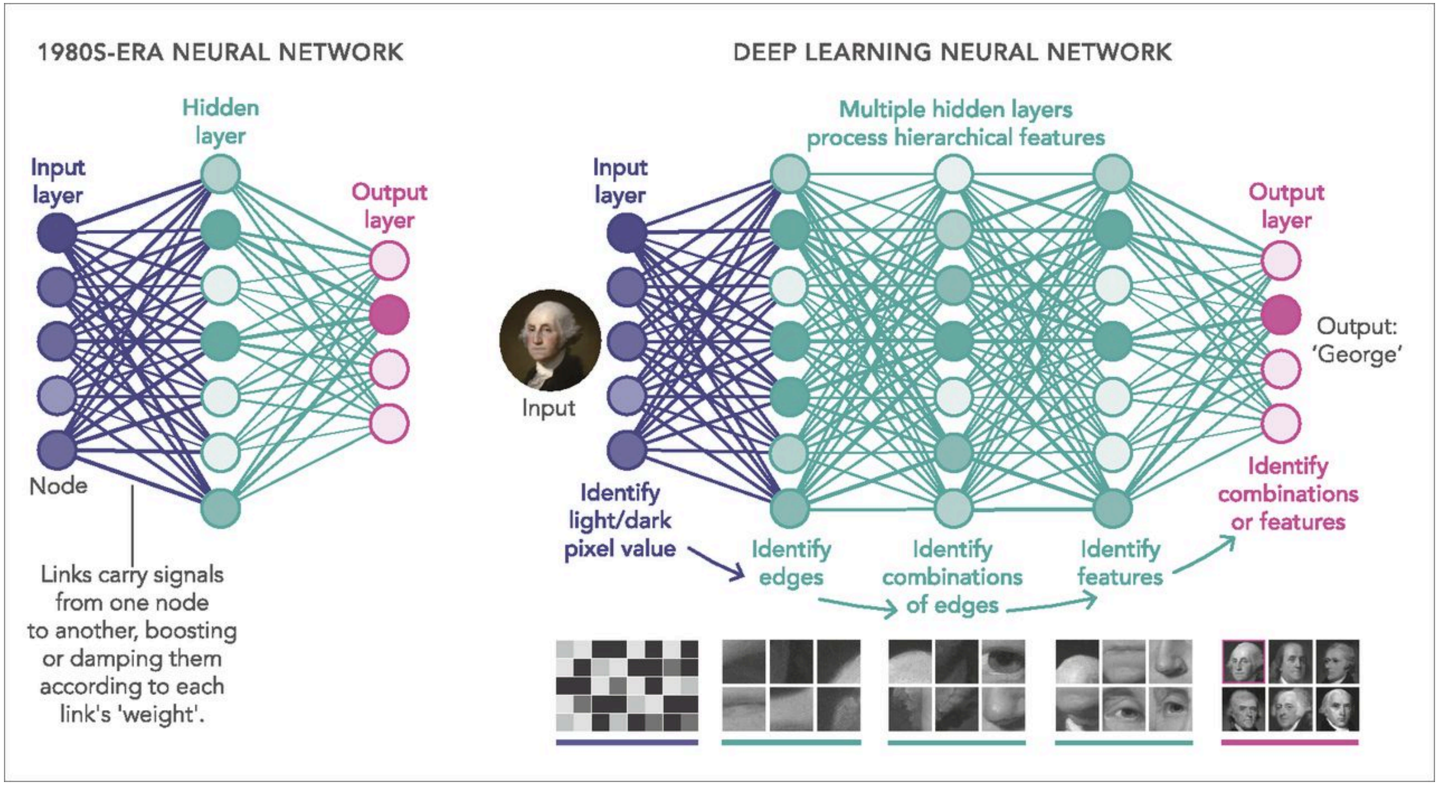
Introduction to Deep Learning

Deep Learning

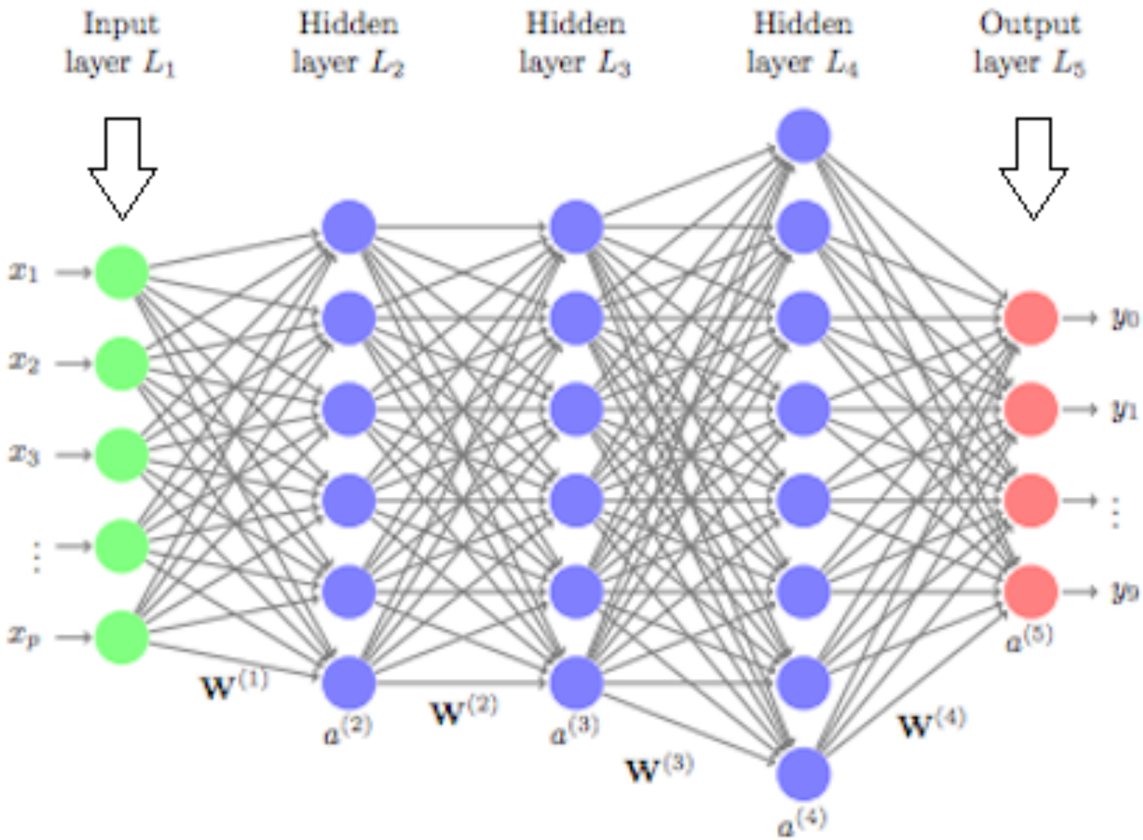
1. Lot of buzz around Deep Learning in the past decade!
2. Deep Learning refers to Neural Networks that is a loose approximation of how the brain works

Feed-forward Deep Learning Architecture Example

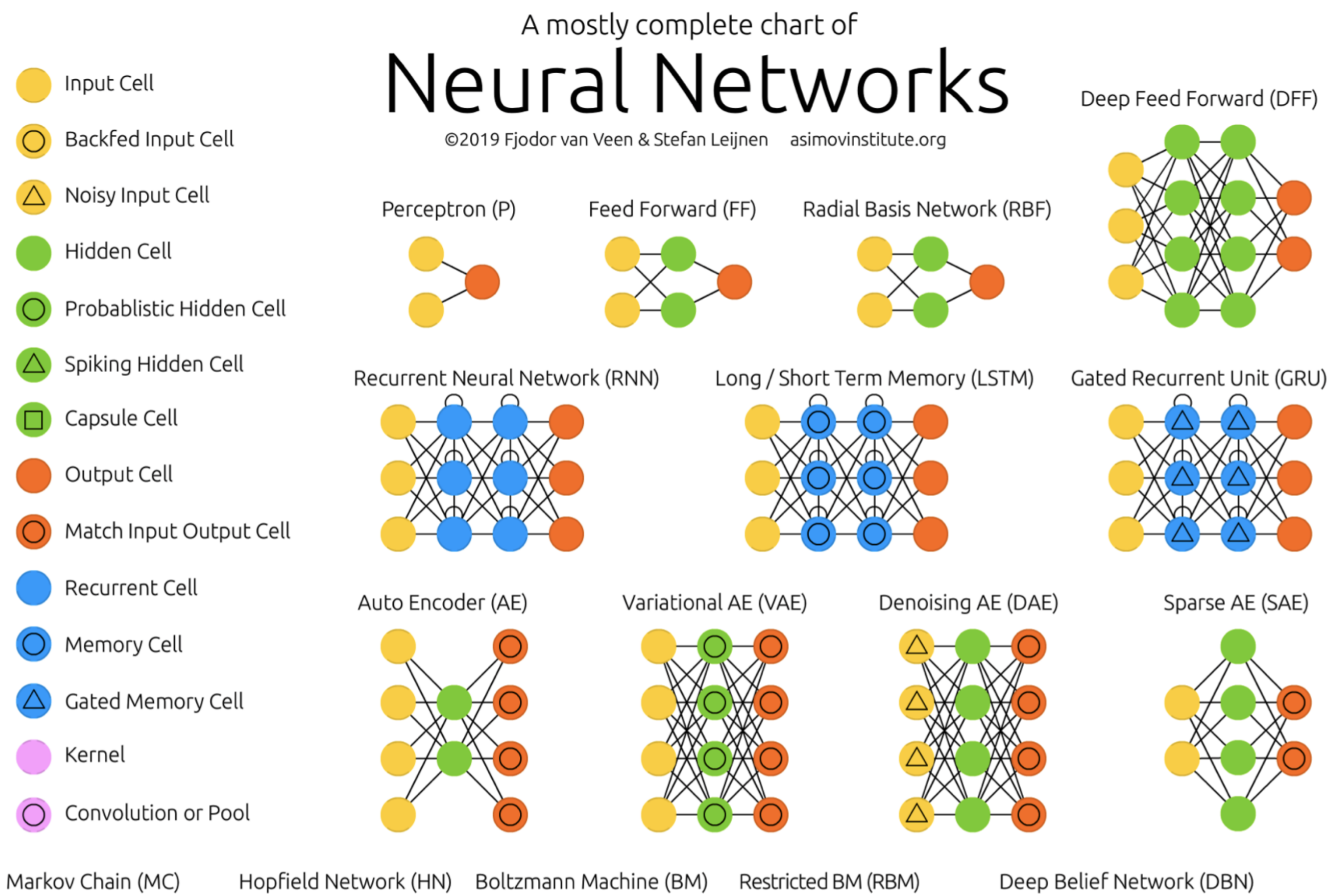
Feed Forward



Feed-forward Deep Learning Architecture Example



Neural Networks Zoo



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

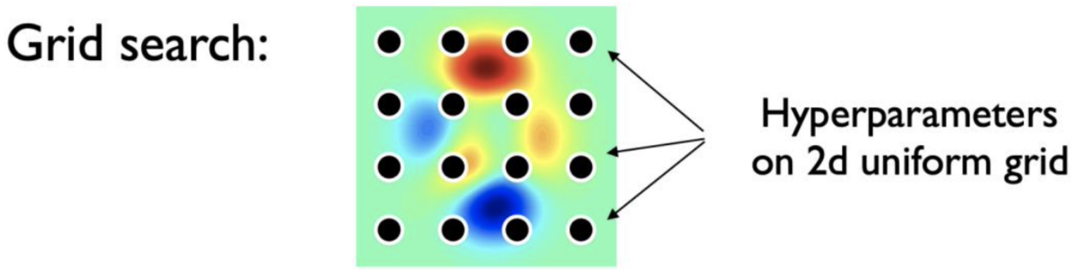
1. Learning rate
2. Number of Hidden Layers
3. Number of neurons per hidden layer
4. Type of non-linear activation function used

Hyper-parameters in Deep Learning

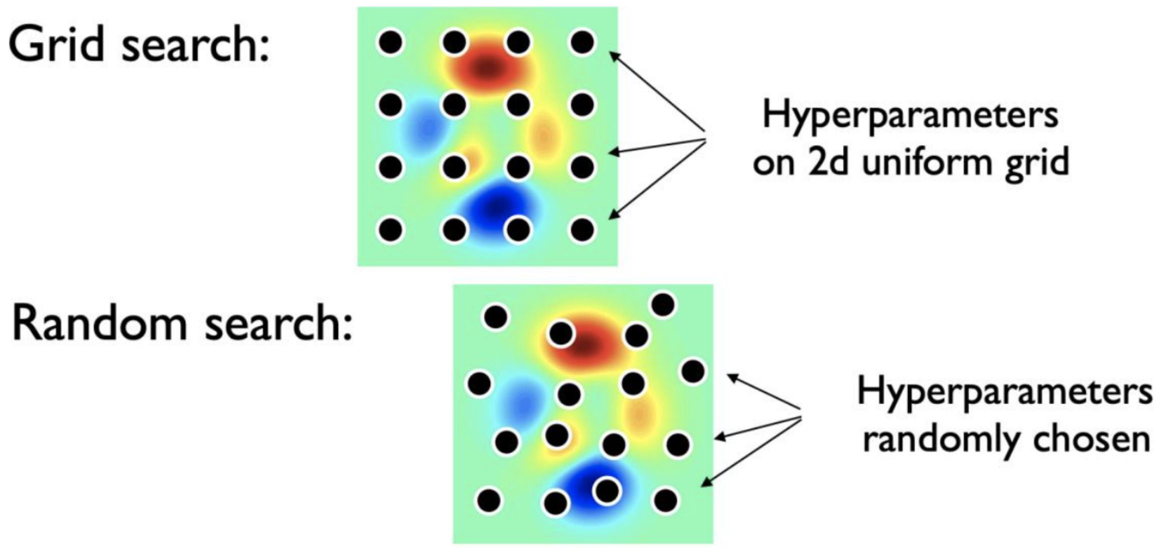
Hyper-parameters

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?

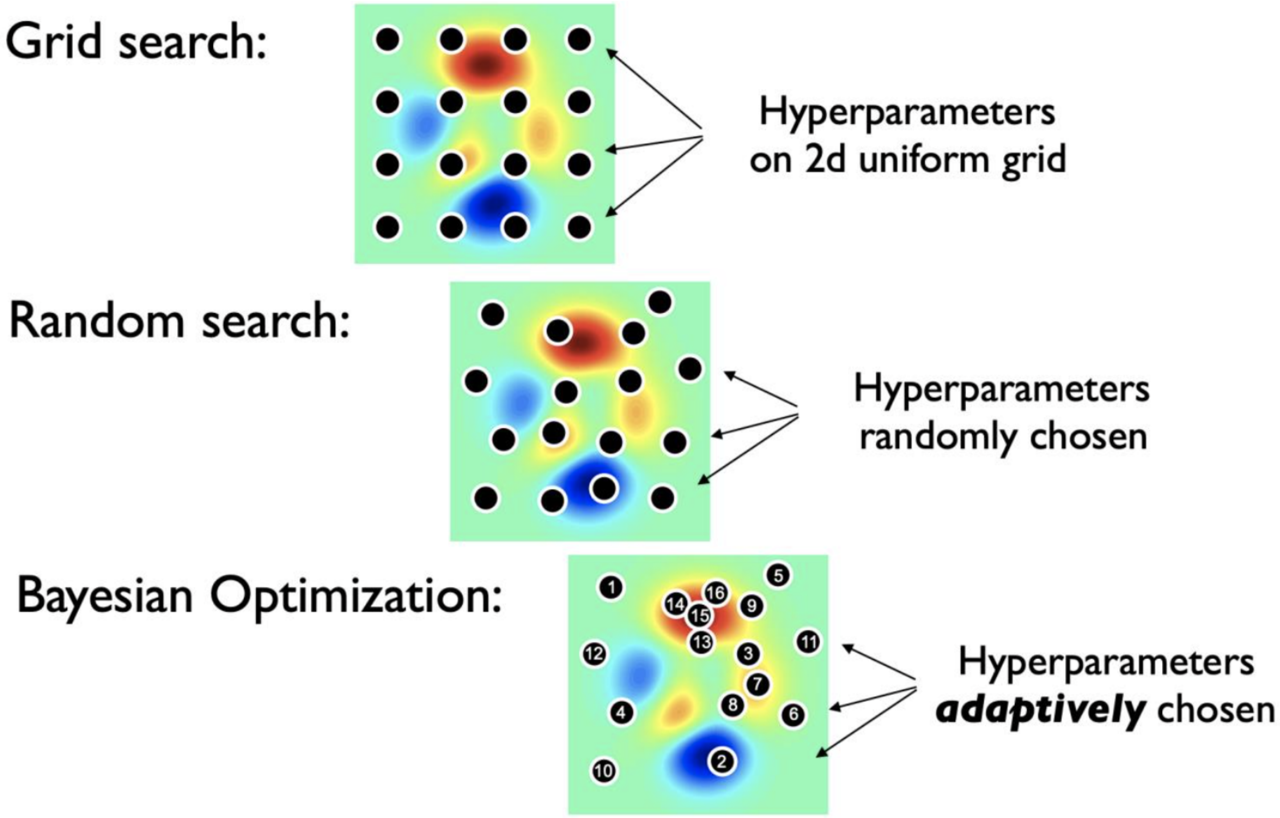
Hyper-parameter tuning methods



Hyper-parameter tuning methods



Hyper-parameter tuning methods



ICE #0

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

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

$$\min \ \ell(w) + \underbrace{\lambda \|w\|_{1/2}}_{\text{Regularization}}$$

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

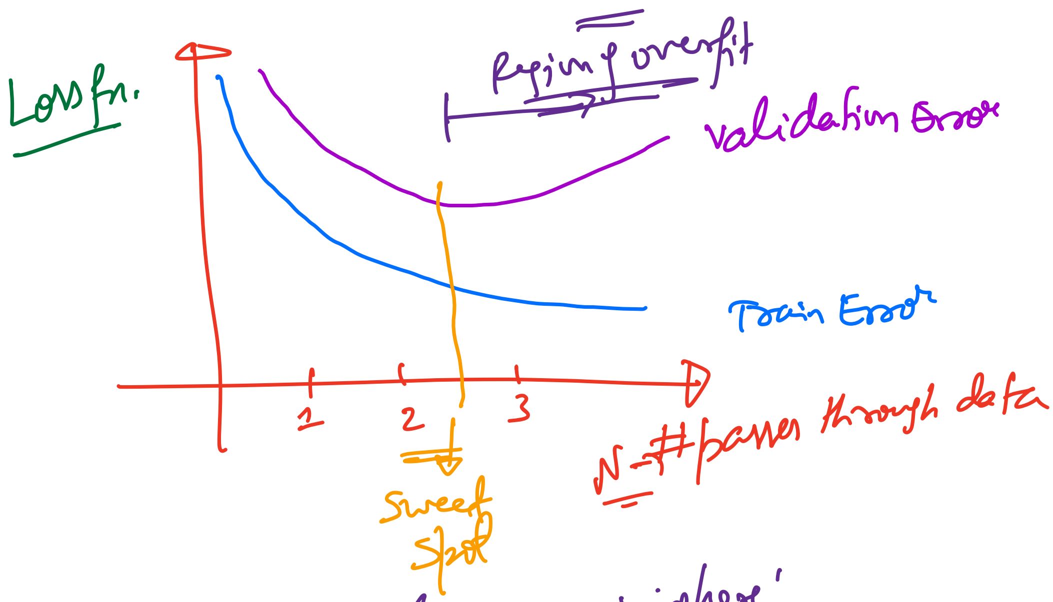
How to handle over-fitting in DNNs

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??



↳ stop training here!

= "Early stopping"

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

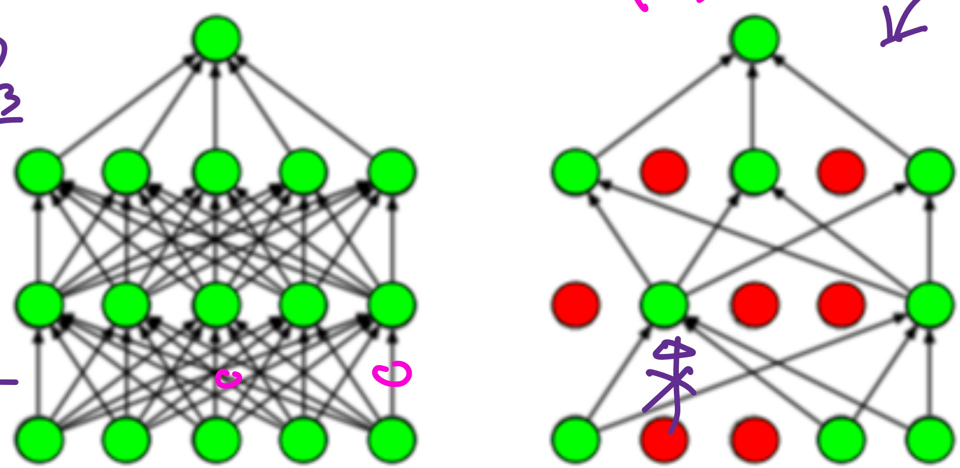
Dropout helps overfit
— Makes Training Faster
Ensemble of models

Specific kind of network

$W^{(3)}$ 3

$W^{(2)}$ 2

$W^{(1)}$ 1



(a) Standard Neural Net (b) After applying dropout.

Hyper parameter ← Dropout = 0.6

ICE #1

Size of Dataset $\overset{?}{\sim}$ Size of parameters / weights?

Overfitting

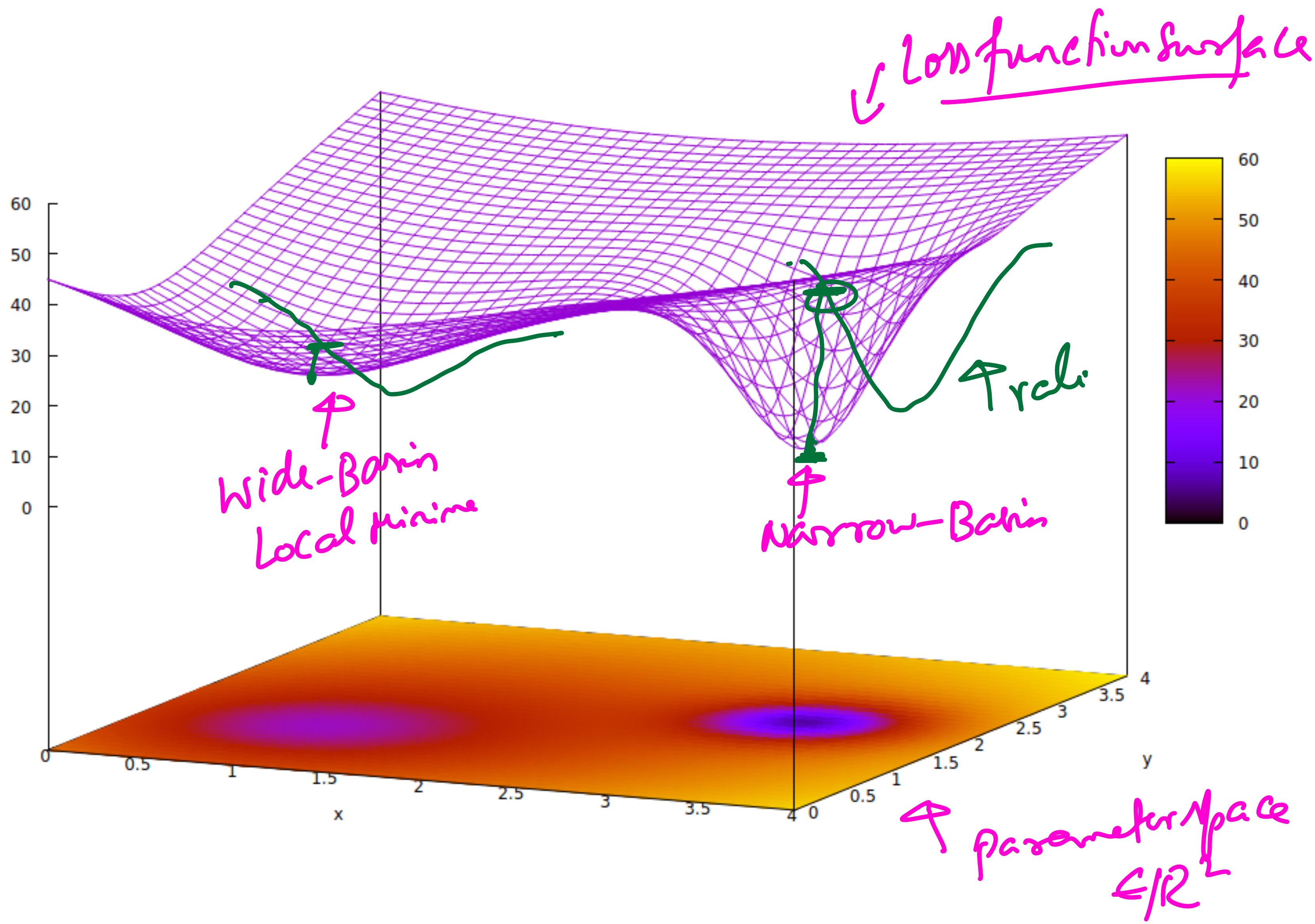
Say you trained your deep learning model on an image data set to classify images as a cat, dog or a human. You realize you are probably overfitting as you are doing much better on training set as compared to the validation and test set. Which of these strategies would probably help reduce overfitting?

→ Data Augmentation

1. Create variations of the images in the training set using image processing techniques and adding it to the training set ✓
2. Using ℓ_1 regularization on the weights ✓
3. Down-sampling the training data set and retraining ✗
4. Error analysis to understand which parts of the validation data does the model perform poorly on? ✓

Good vs Bad Local minima

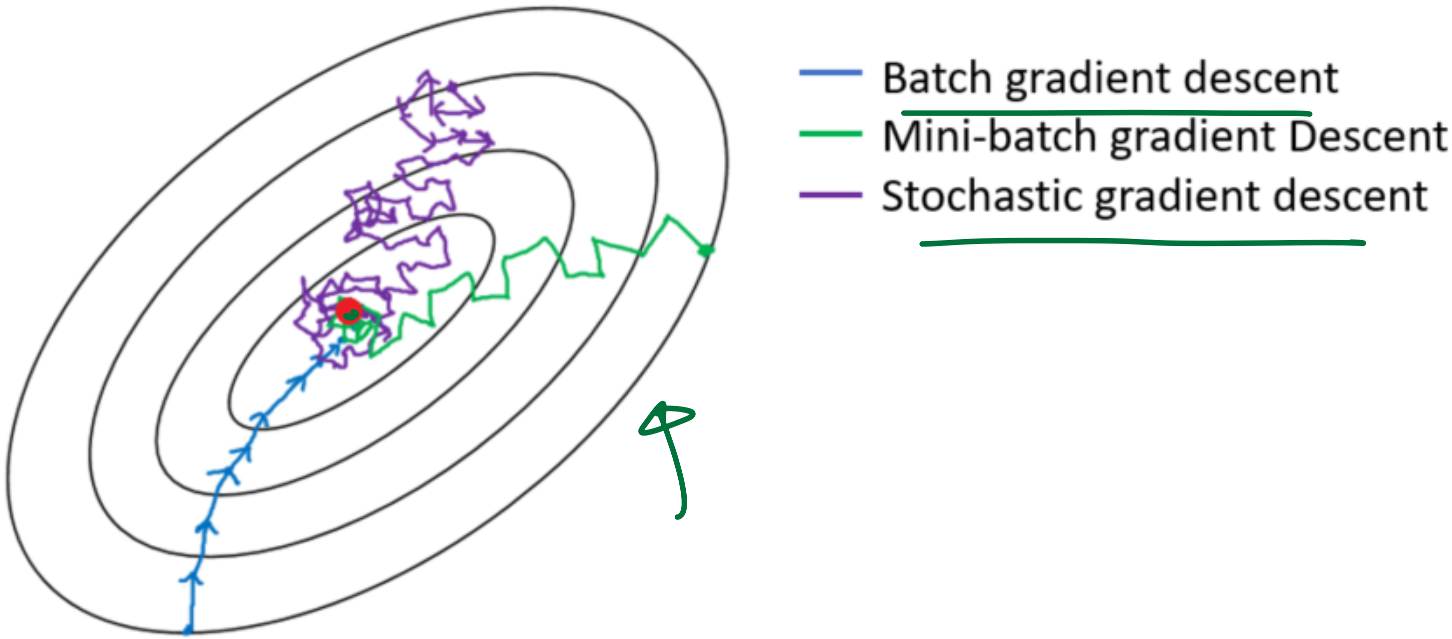
(Mechanics of DL)



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

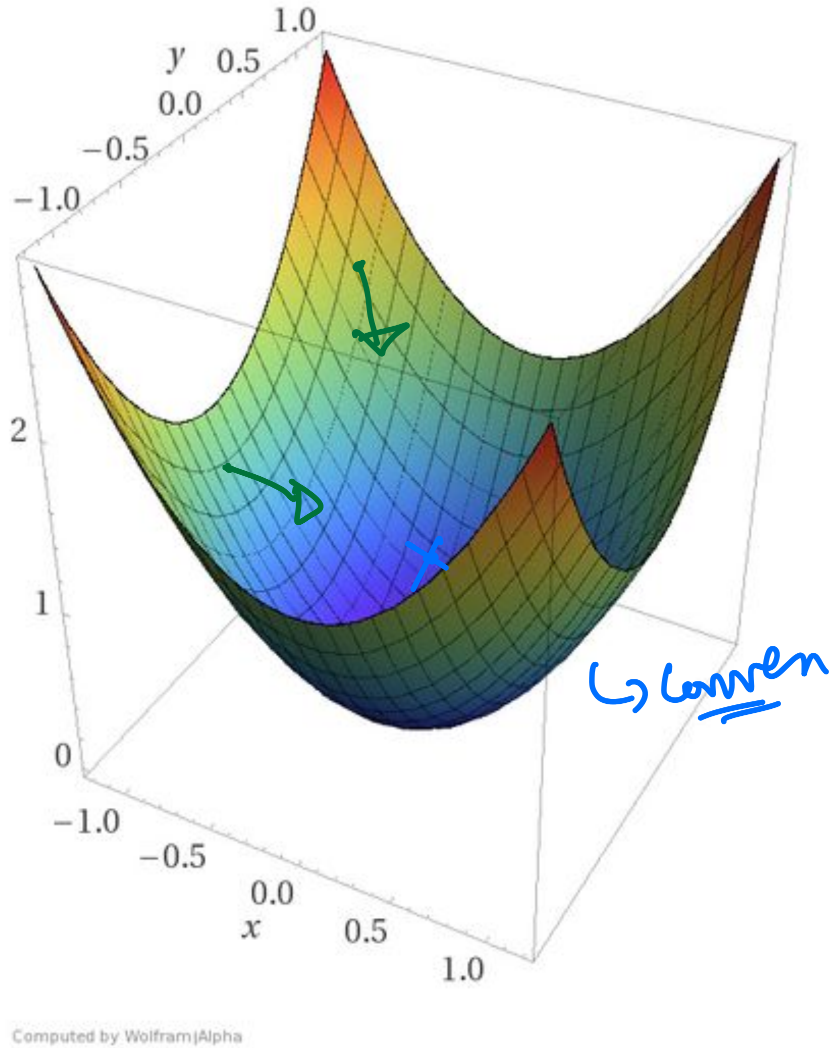
Take a convex/non-convex function, f. GD allows you to find a local optimum to f.

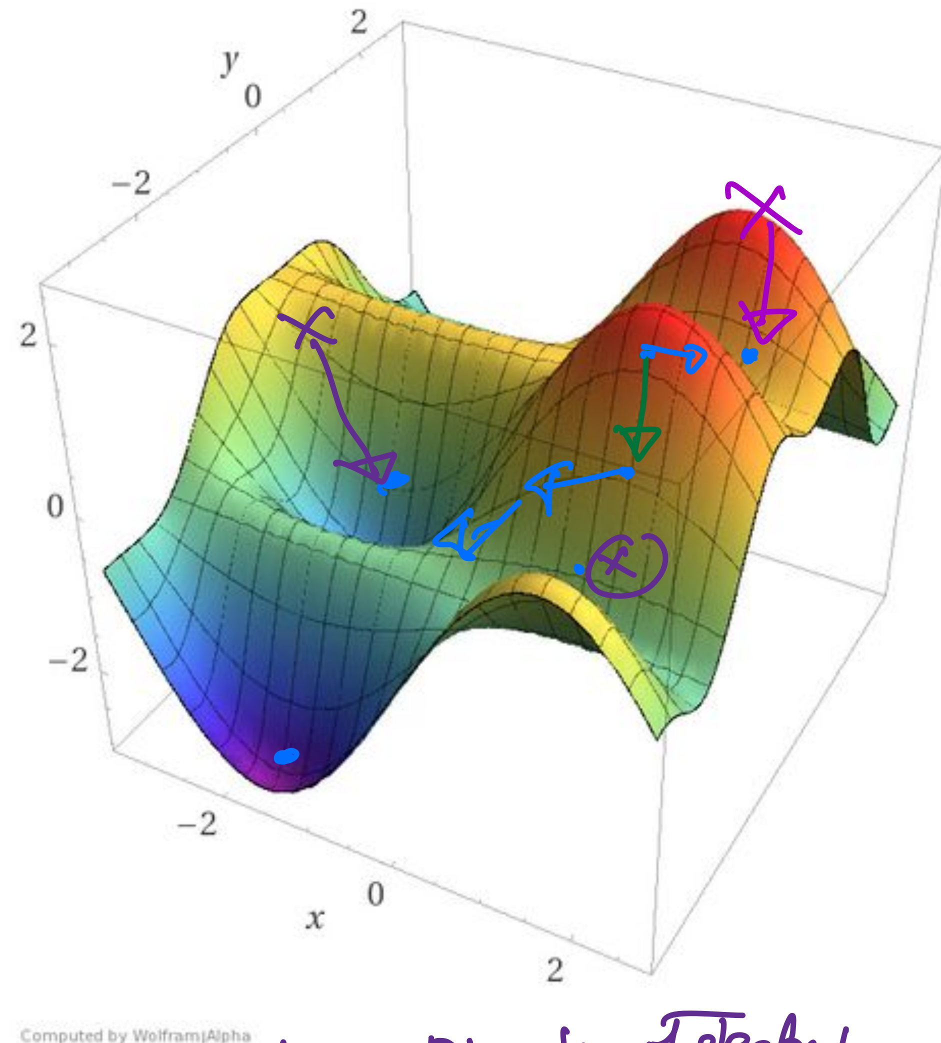
Why is this important?

Consider the Logistic Regression problem. $\hat{w}$ is a local optimum (and global optimum) to the function

$$l(w) = \underbrace{\sum_i y_i \log(1 + e^{-w^T x^i}) + \sum_i (1 - y_i) \log(1 + e^{-w^T x^i})}_{\text{Binary Cross-Entropy}}$$

Negative Gradient helps you view the direction of descent





Negative Gradient = Direction of steepest 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** $w \leftarrow w - \underline{lr} * \underbrace{\nabla L(w)}$

Learning Rate / step size

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 #2

Derivative (1 min)

Find the derivative of w^2

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

ICE #3

Pytorch: Automatic Differentiation

$$\lambda(w_1^2 + w_2^2 + \cdots + w_d^2)$$

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, $\nabla_w R(w)$?

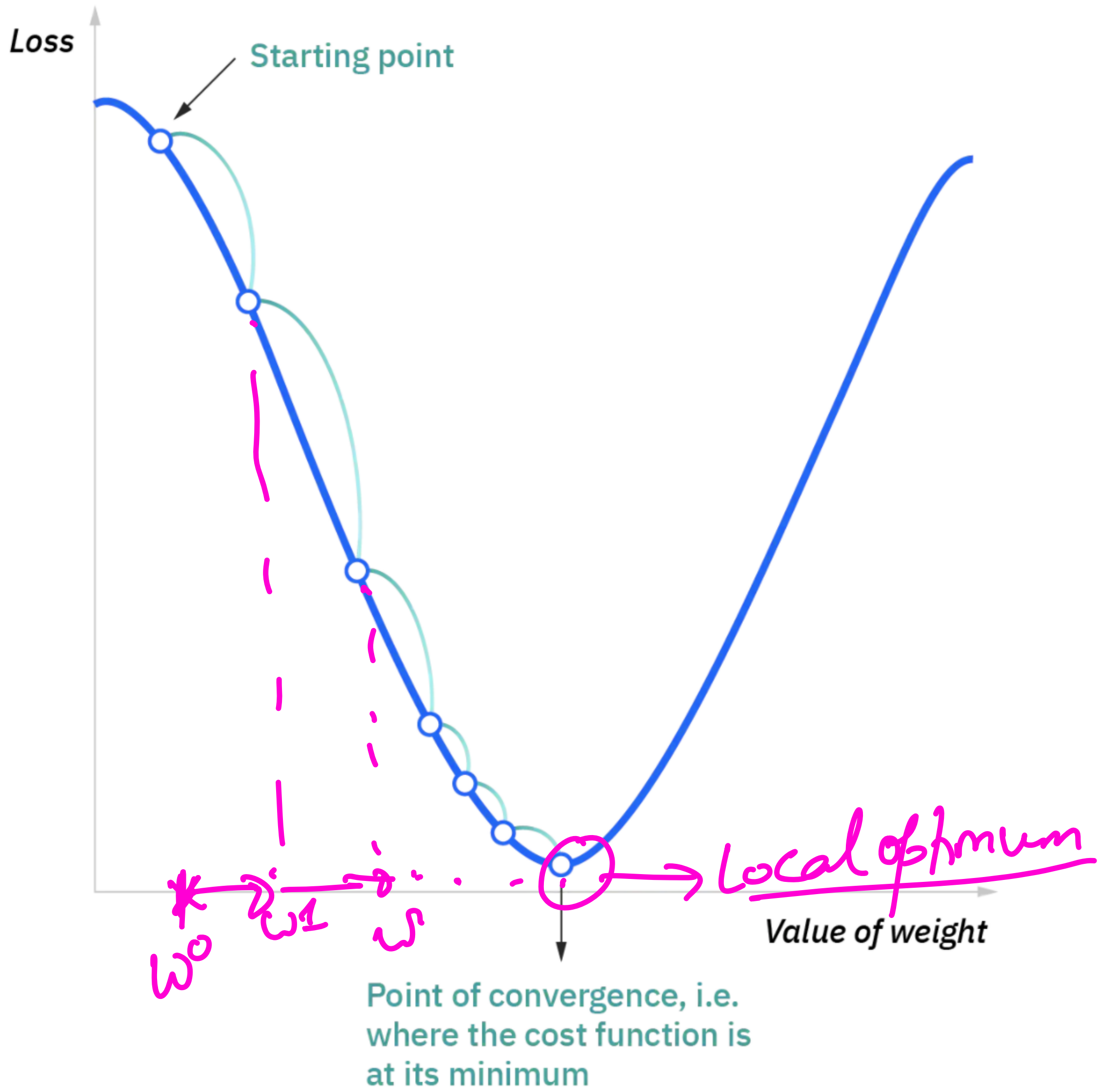
- a) $2\lambda\|w\|_2$
- b) $\lambda\|w\|_2 w$
- c) $2\lambda w$
- d) $2\lambda\|w\|_2 w$

$$\frac{\partial}{\partial w_1} R(w) = \frac{\partial}{\partial w_1}(\lambda w_1^2) = 2\lambda w_1$$

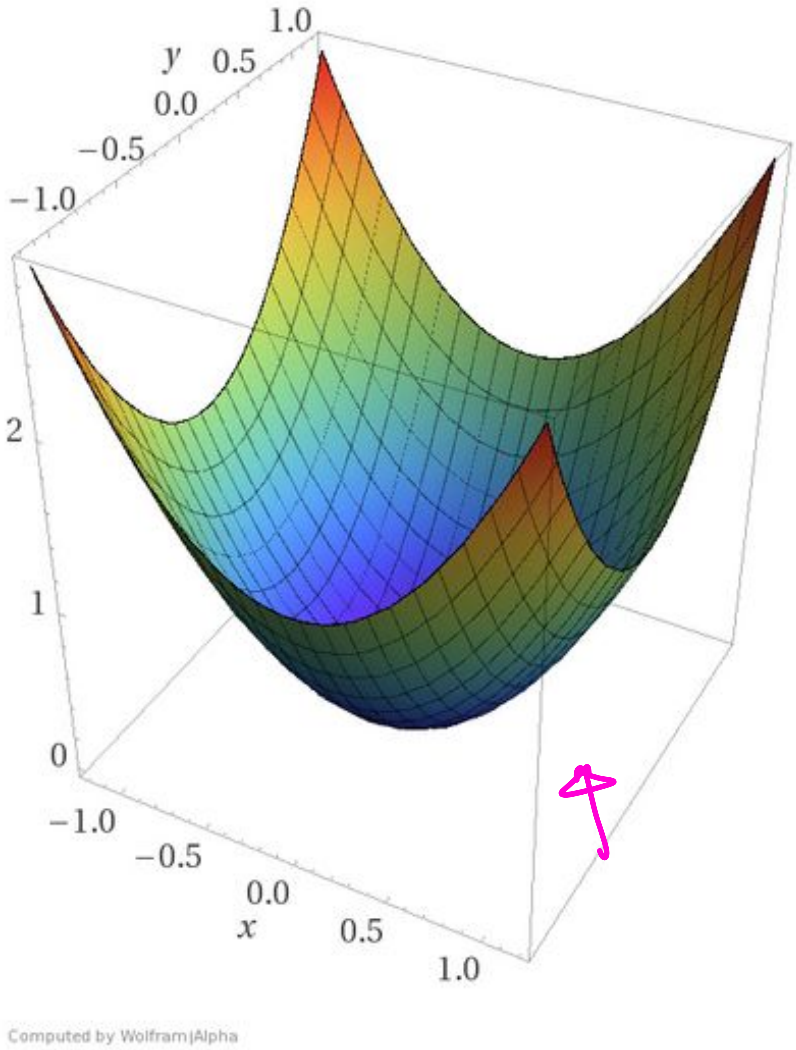
$$\frac{\partial}{\partial w_j} R(w) = 2\lambda w_j$$

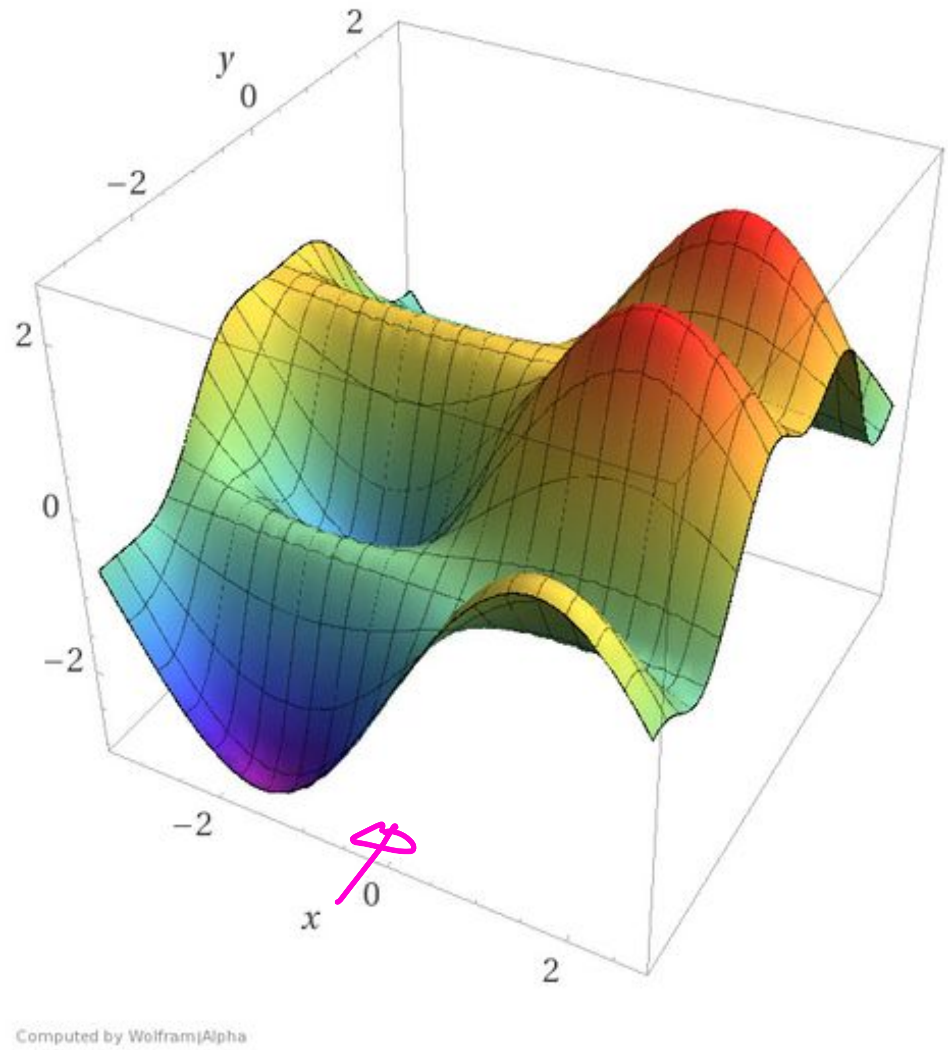
$$\begin{bmatrix} 2\lambda w_1 \\ 2\lambda w_2 \\ \vdots \\ 2\lambda w_d \end{bmatrix} = 2\lambda w$$

GD in one dimension



Loss function in 2 dimensions





Gradient Descent Properties

1. Gradient Descent converges to a local minimum

Gradient Descent Properties

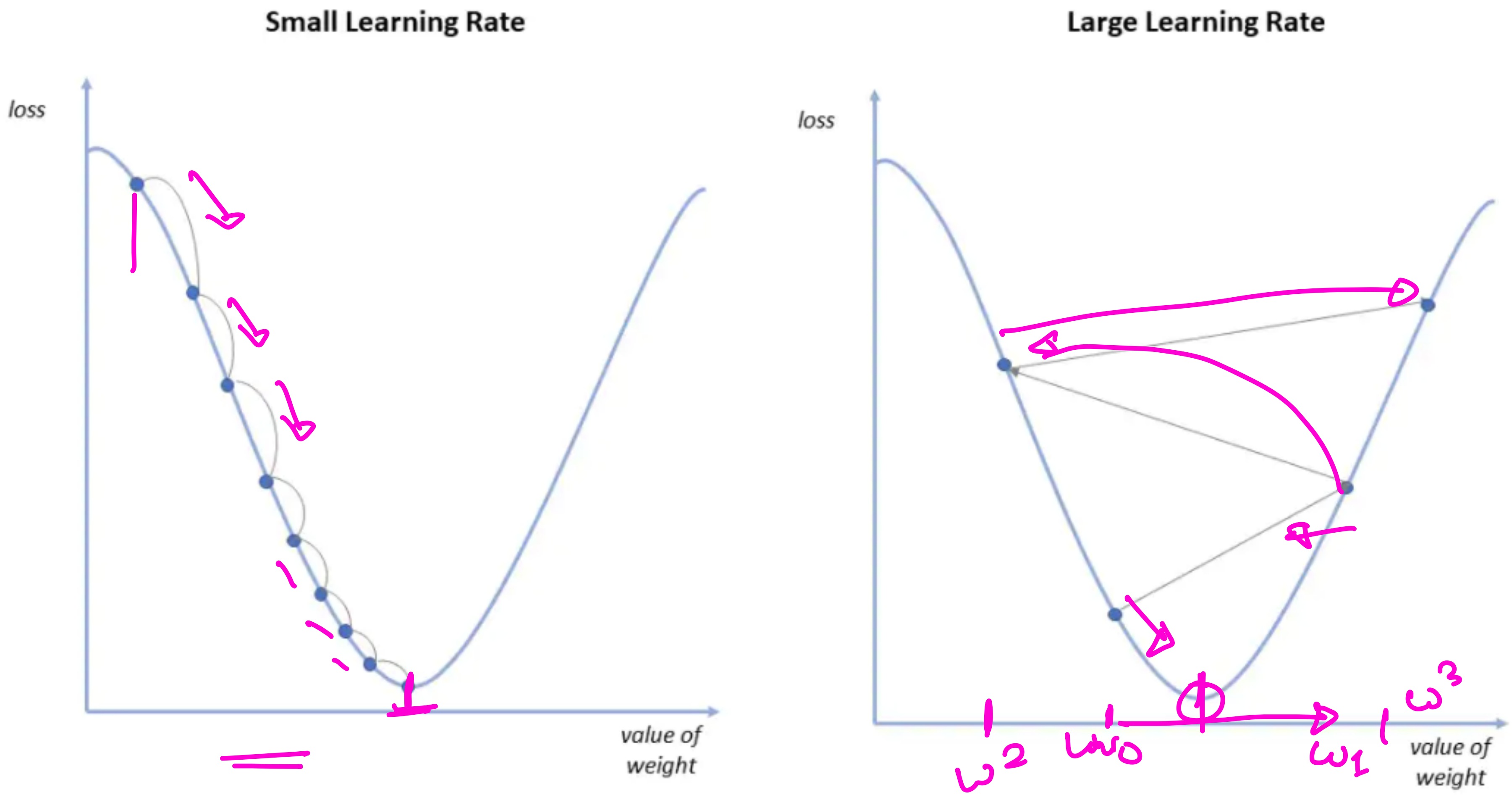
1. Gradient Descent converges to a local minimum
2. If L is a convex function, all local minima become a global minima!

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.

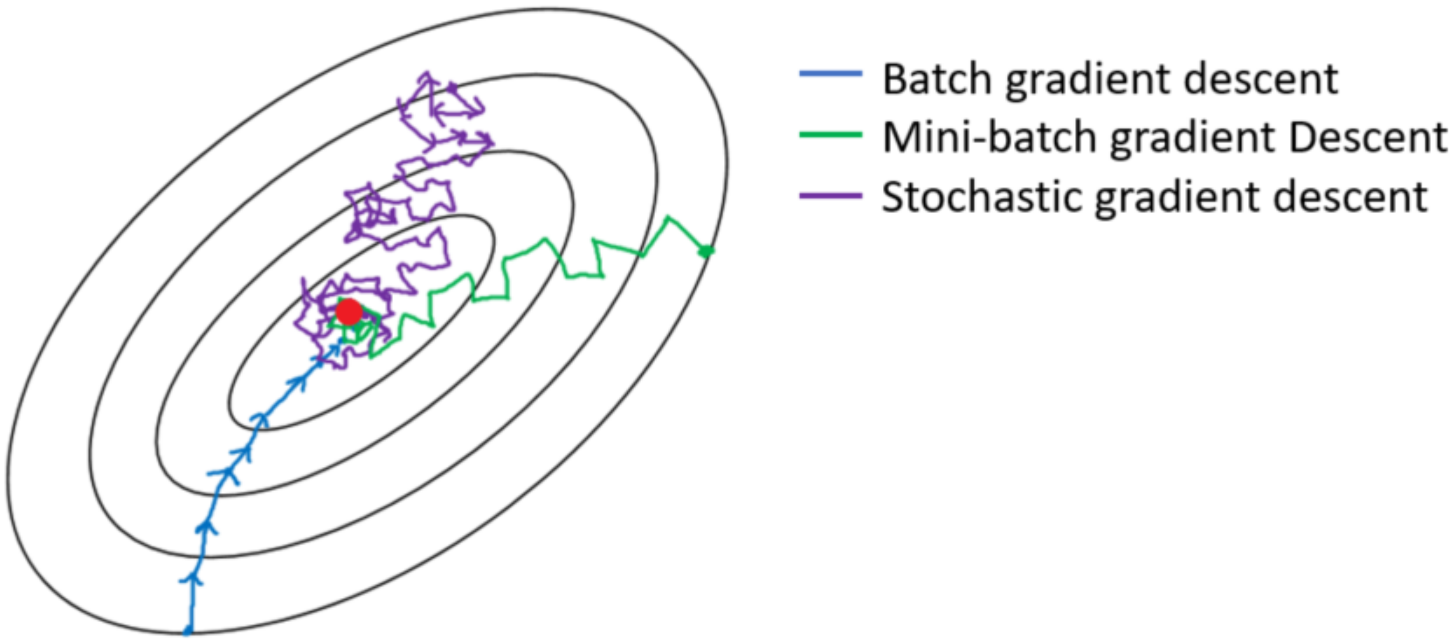
Pick starting point at Random

Effect of Learning Rate

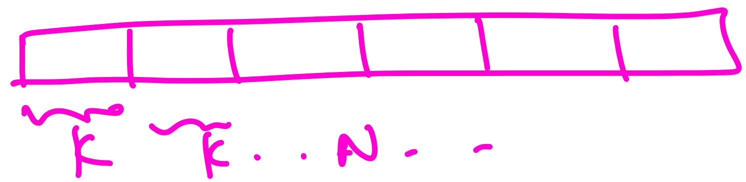


(Adaptive Learning Rate)

SGD behavior in search space



SGD in practice - mini-batch SGD!



B Batches of k

$B \times k = N$

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.

$k = 128$ or 256

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

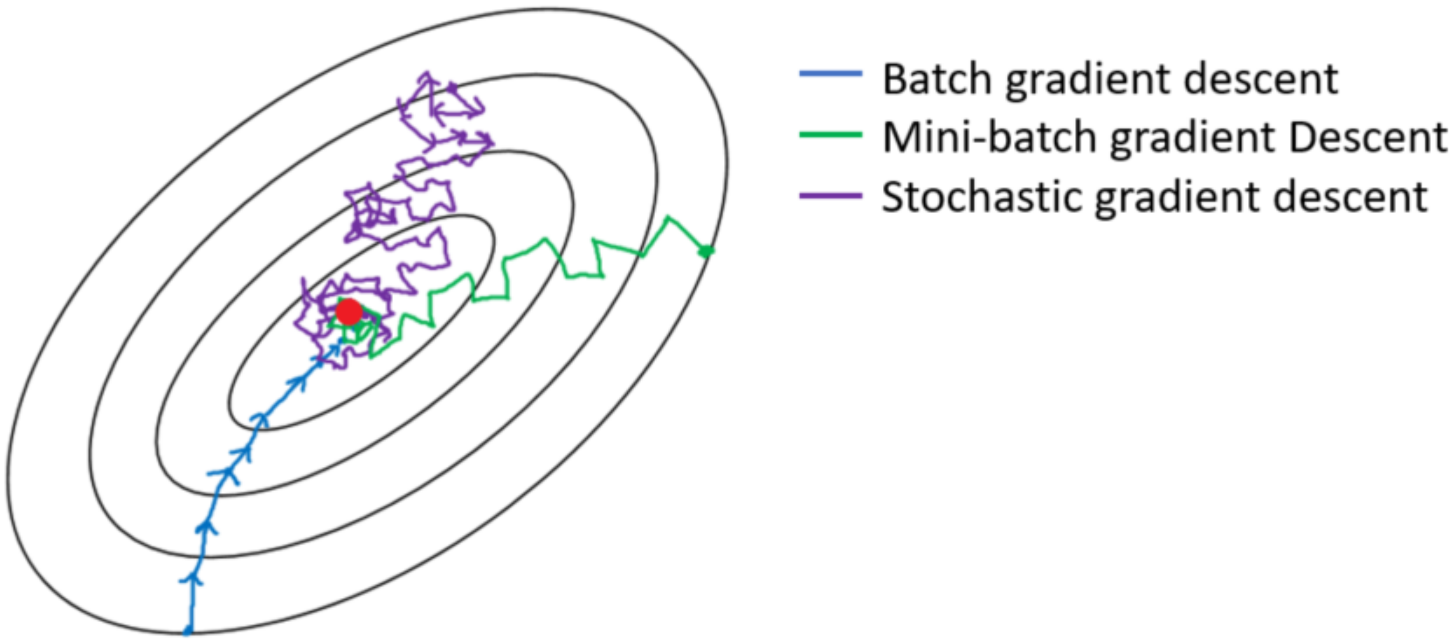
How many passes through the data?

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.

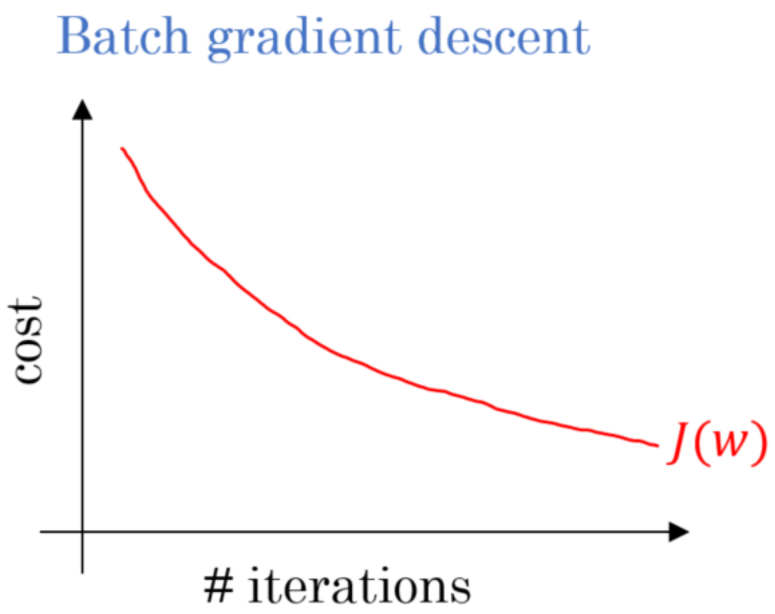
1. **Initialize** $w = w_0$ (randomize) Pick a batch of k data points at random between 1 and N: $i_1, i_2, \ldots, i_k$!
2. **Gradient Descent** $w^{k+1} \leftarrow w^k - lr * \sum_{j=1}^{k} \nabla_w L_{i_j}(w^k)$
3. **Iterate** Repeat step 2 and 3 until w converges, i.e.

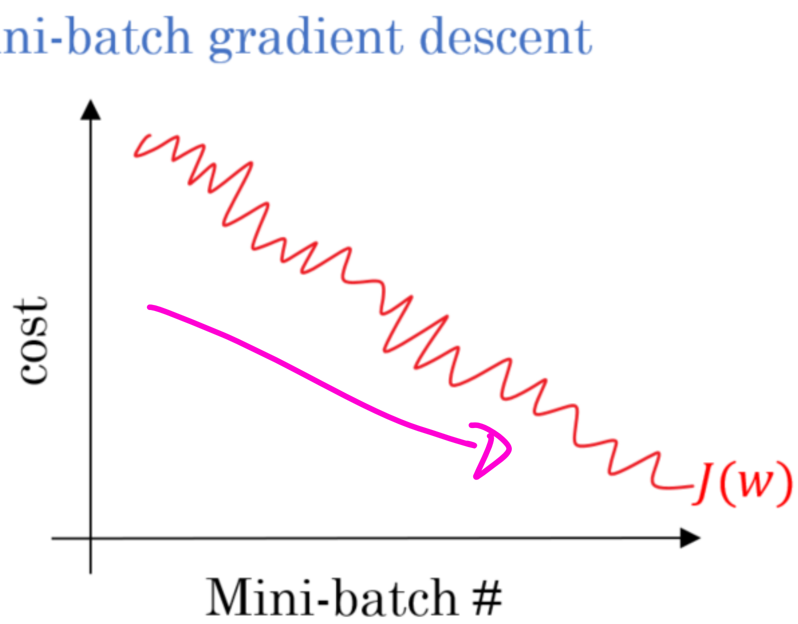
$$\|w^{k+1} - w^k\| / \|w^k\| \leq 10^{-3}$$

GD behavior in the search space



GD vs Mini-batch convergence behavior





GD vs mini-batch SGD

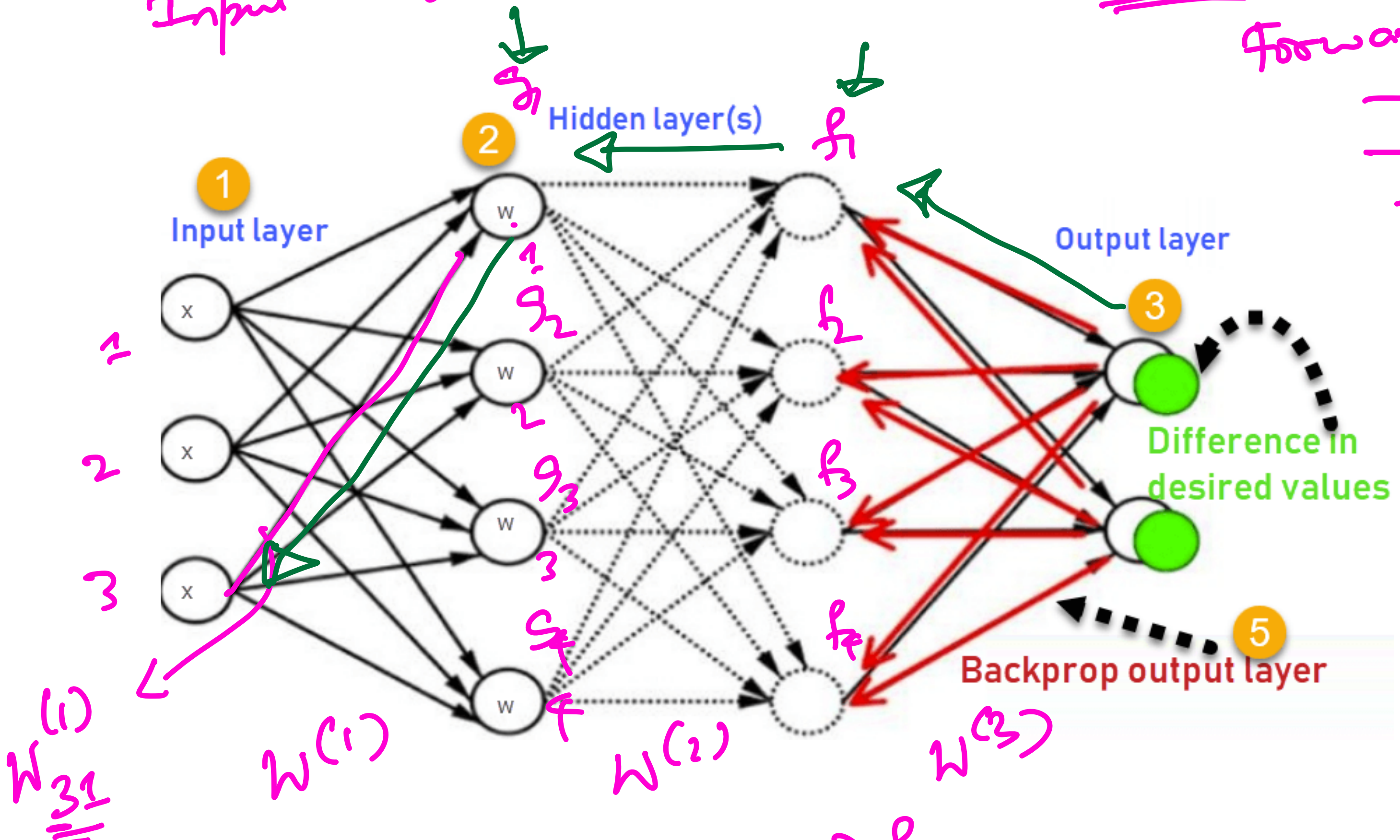
In practice

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

Input → → → output

Forward pass
→ Prediction
→ Function value



Back Propagation = Computing Gradient using chain Rule for derivatives

$$\frac{\partial L}{\partial W^{(1)}_{31}} = \frac{\partial L}{\partial f_1} \times \underbrace{\frac{\partial f_1}{\partial W^{(1)}_{31}}}_{} + \cdots$$

$$\frac{\partial f_1}{\partial W^{(1)}_{31}} \rightarrow \frac{\partial f_1}{\partial S_1} \times \frac{\partial S_1}{\partial W^{(1)}_{31}} + \cdots$$

ICE #4

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. Gradient Descent
4. Backpropagation in Neural Networks
5. Overfitting in Neural Networks