EEP 596: LLMs: From Transformers to GPT || Lecture 4

Dr. Karthik Mohan

Univ. of Washington, Seattle

January 14, 2026

Logistics

- Paper Presentation Slots Assignment
- Assignment 2
- Review Session Friday
- Zoom Lecture on Monday (Covering for a short-quarter)
- 1:1 connect for pet project or ML topic (limited calendly spots)
- Anything else?

Paper Presentation Pointers

- Main idea is for you to get in touch with cutting edge papers + ability to go deeper into understanding a paper!
- Form teams of 2 - We unfortunately don't have bandwidth for solo presentations!
- No more than 3 presentations per date/class
- Pre-Record a 3 minute video before your class with inputs from both the team members - No live presentations. We will play the pre-recorded video in class when its your turn
- Format for each team is 3 minute pre-recorded video + 2 minute questions
- Template for slide deck will be shared by Friday, this week
- Please make sure to understand the paper really well - As you will get couple of questions post your talk - This will count towards your grade!

Coding Assignment 2 - Yelp Review Rating Prediction

- Multi-class classification problem
- Compare GPT4o-Nano with fine-tuned BERT
- Interpret your results and seek insights
- Kaggle evaluation for a friendly match-up with the class
- Assigned today and due next Friday

Last Lecture

- Deep Learning Fundamentals

Today's Lecture

- Deep Learning Fundamentals
- Semantic Search

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

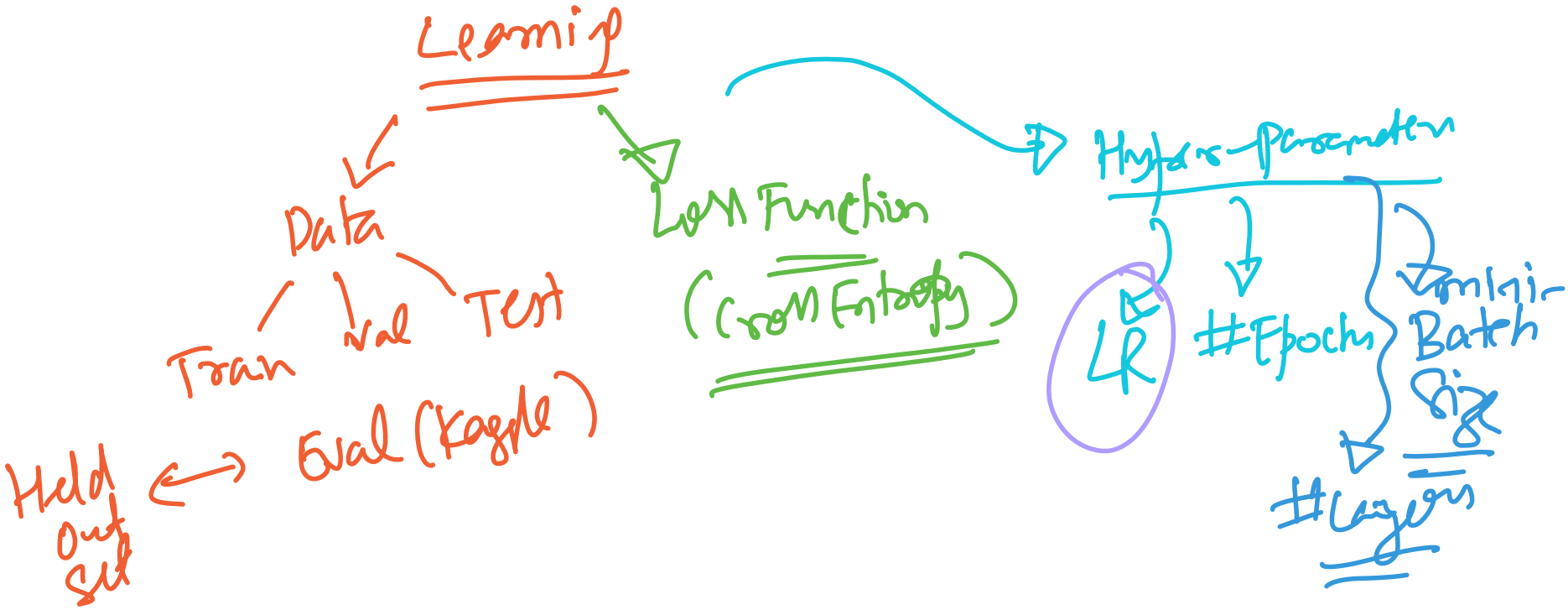
SGD with mini-batch

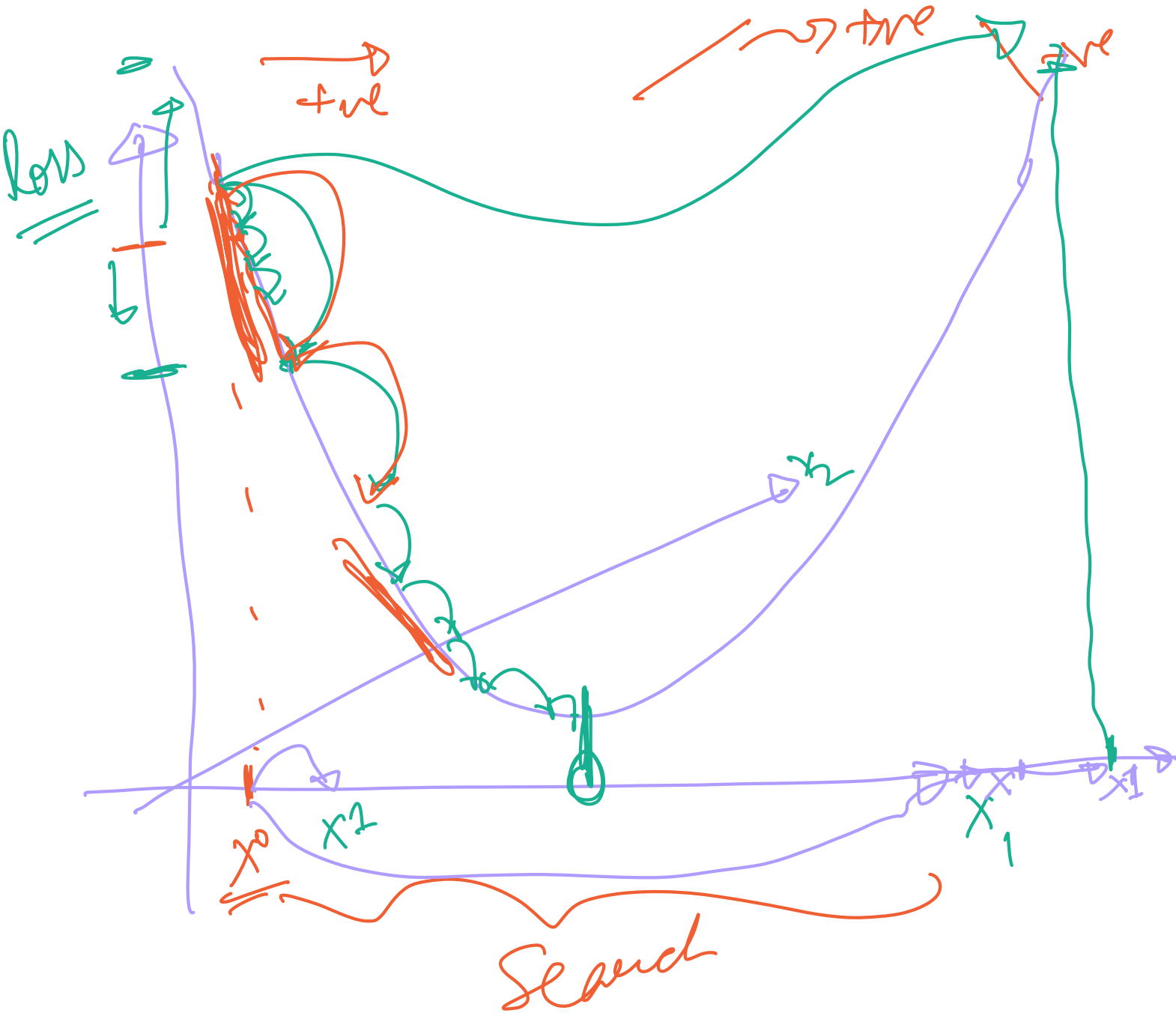
SGD mini-batch is the staple diet. However there are some **learning rate schedulers** that are known to work better for DNNs - Such as Adagrad and more recently, ADAM. ADAM adapts the learning rate to each individual parameter instead of having a global learning rate.

How do we compute gradient in a DNN?

Back-propagation!

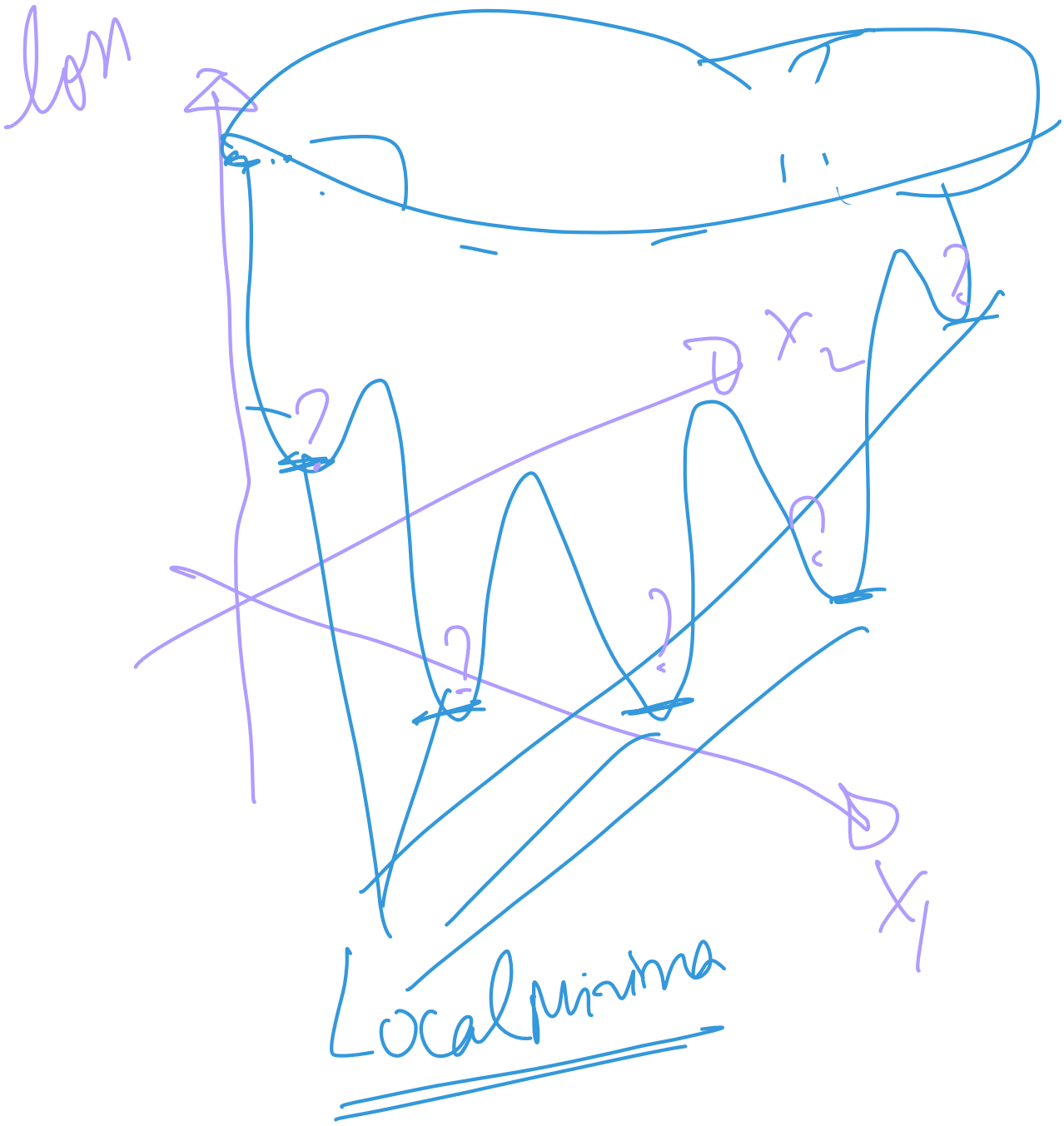
Impact of Learning Rate - Example



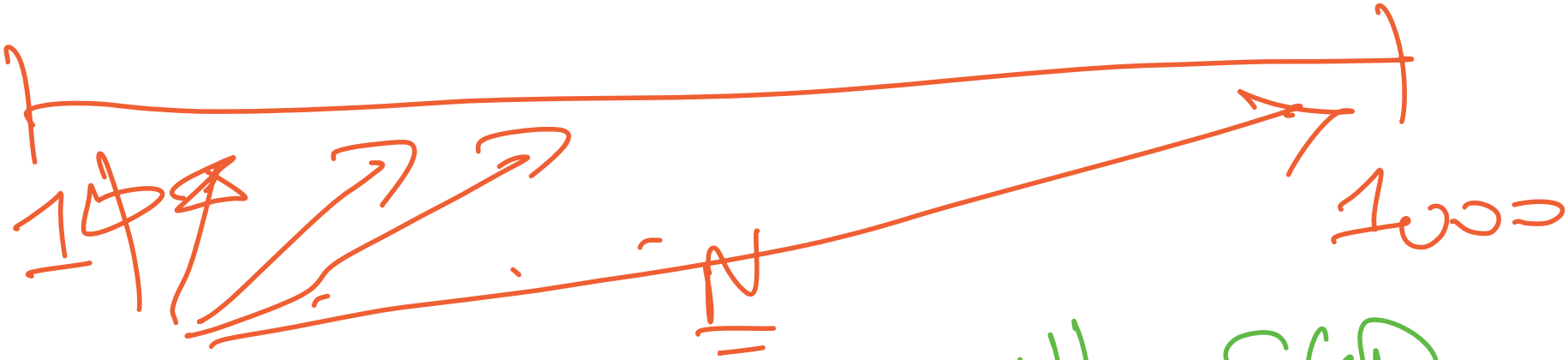


Window and Learning rate

1e-4 - 1e-2

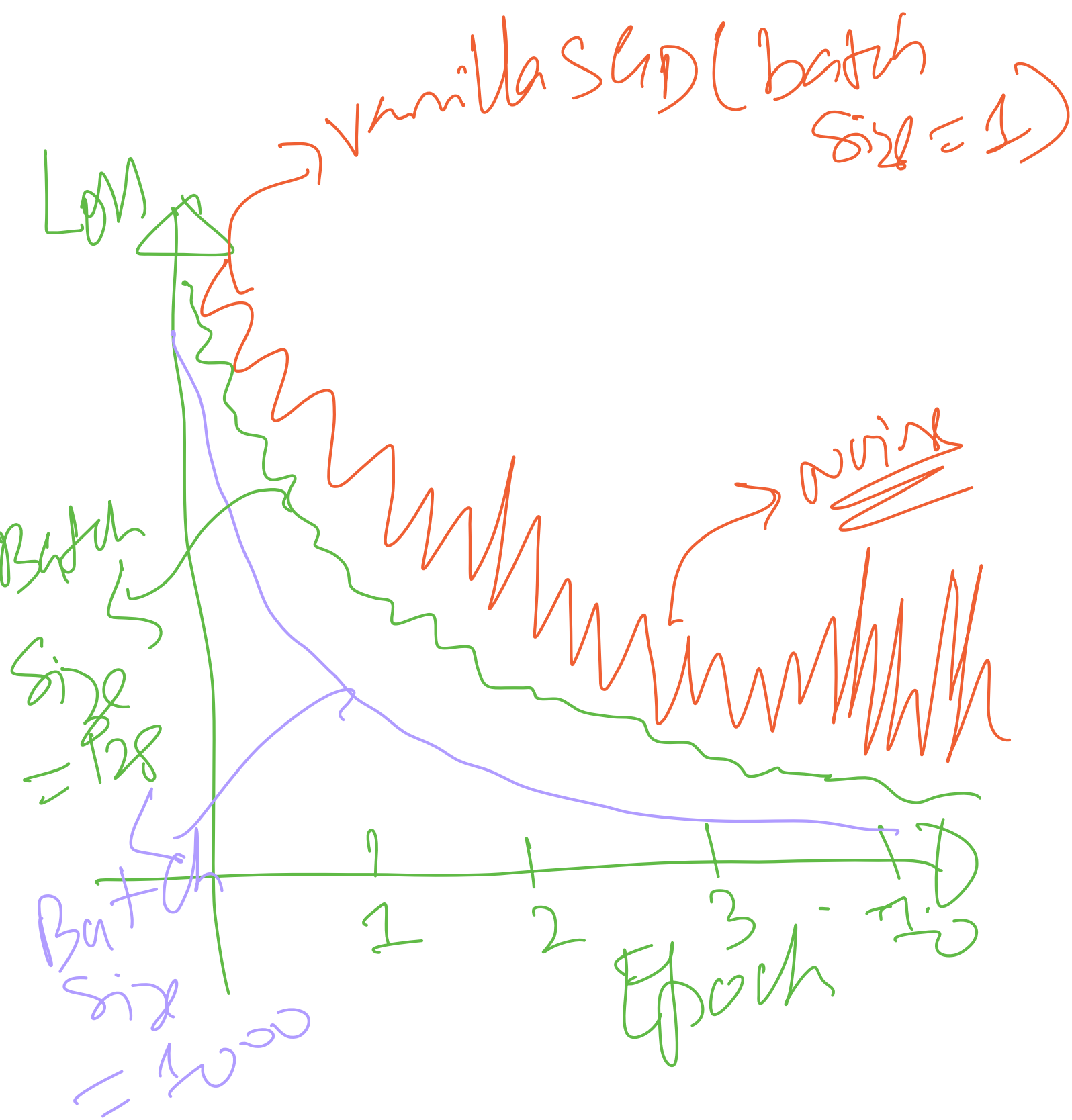

Local minima

Understanding Batch Size



batch size = 1 → Vanilla SGD

batch size = N → Regular Gradient Descent


vanilla SGD (batch size = 1)
Loss
noisy
Batch size = 128
Batch size = 1000
1
2
3
10
Epoch

ICE #1

Assume you are minimizing a function as follows:

$$f(\theta) = \sum_{i=1}^{d} (a_i \theta_i^2 + b_i \theta_i + c_i)$$

What is the dimension of the gradient with a full-batch size and a batch-size of 8 (assume $d > 8$)

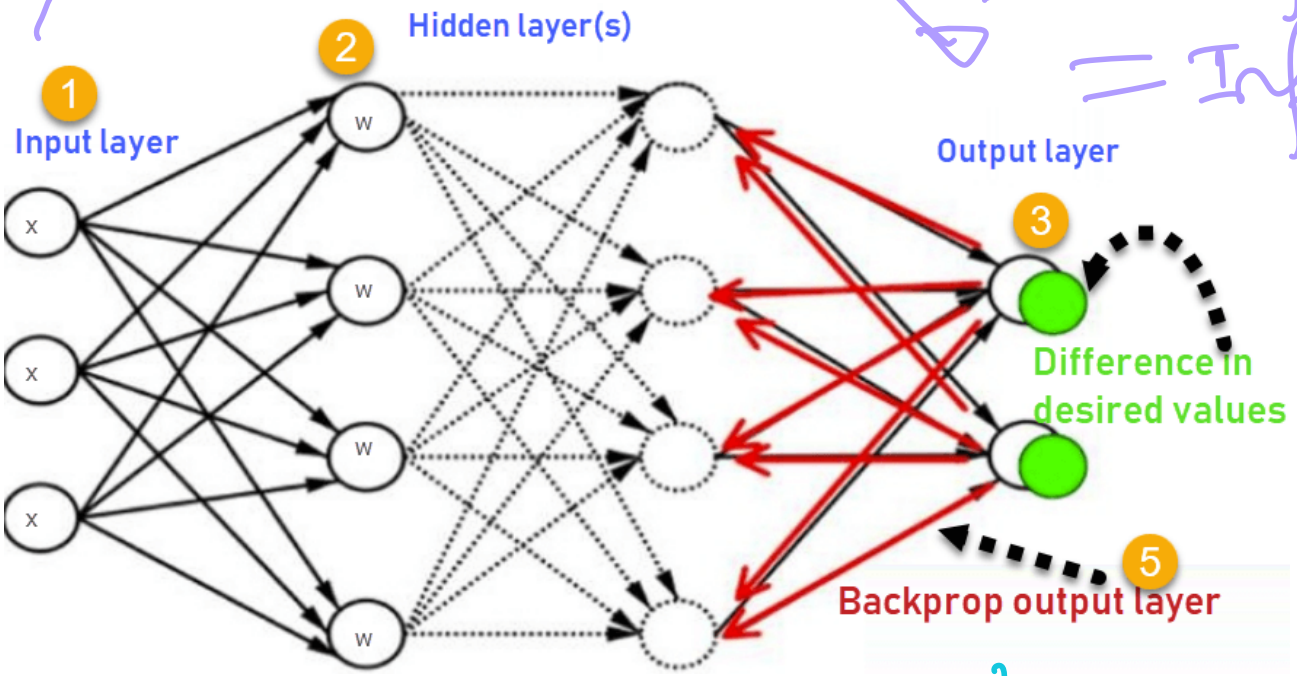
1. 8,8
2. d,8
3. 1,8
4. d,d

ICE #2

What is the main advantage of SGD as compared to full-batch GD?

1. Lower compute requirement
2. Lower memory requirement
3. Better accuracy
4. Safer model

Forward Propagation vs Back-propagation

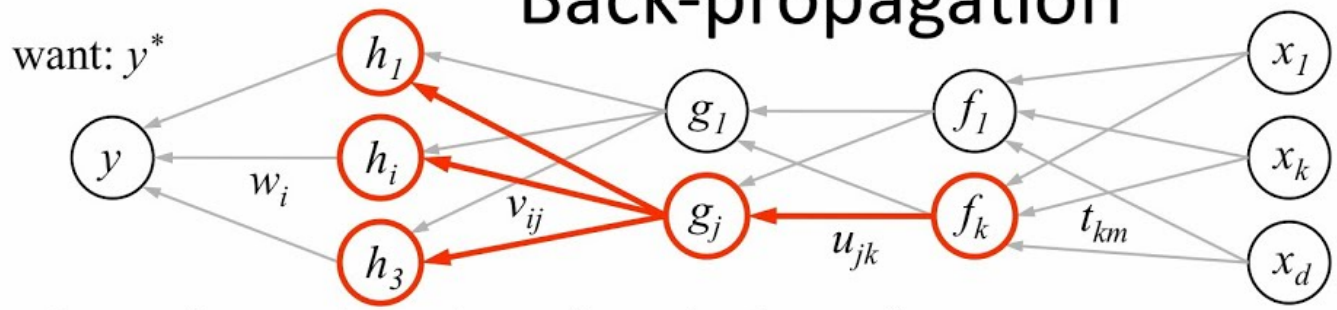


Forward Propagation = Getting Prediction = Inference

Backprop = Computing Gradient

Back Propagation explained

Back-propagation



1. receive new observation $\boldsymbol{x} = [x_1 \ldots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer 1…L
   compute g_j based on units f_k from previous layer: $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction y and error $(y\text{-}y^*)$
4. **back-propagate error:** for each unit g_j in each layer L…1

(a) compute error on g_j

$$\frac{\partial E}{\partial g_j} = \sum_i \sigma'(h_i) v_{ij} \frac{\partial E}{\partial h_i}$$

should g_j be higher or lower? | how h_i will change as g_j changes | was h_i too high or too low?

(b) for each u_{jk} that affects g_j

(i) compute error on u_{jk}

$$\frac{\partial E}{\partial u_{jk}} = \frac{\partial E}{\partial g_j} \sigma'(g_j) f_k$$

do we want g_j to be higher/lower | how g_j will change if u_{jk} is higher/lower

(ii) update the weight

$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$$

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

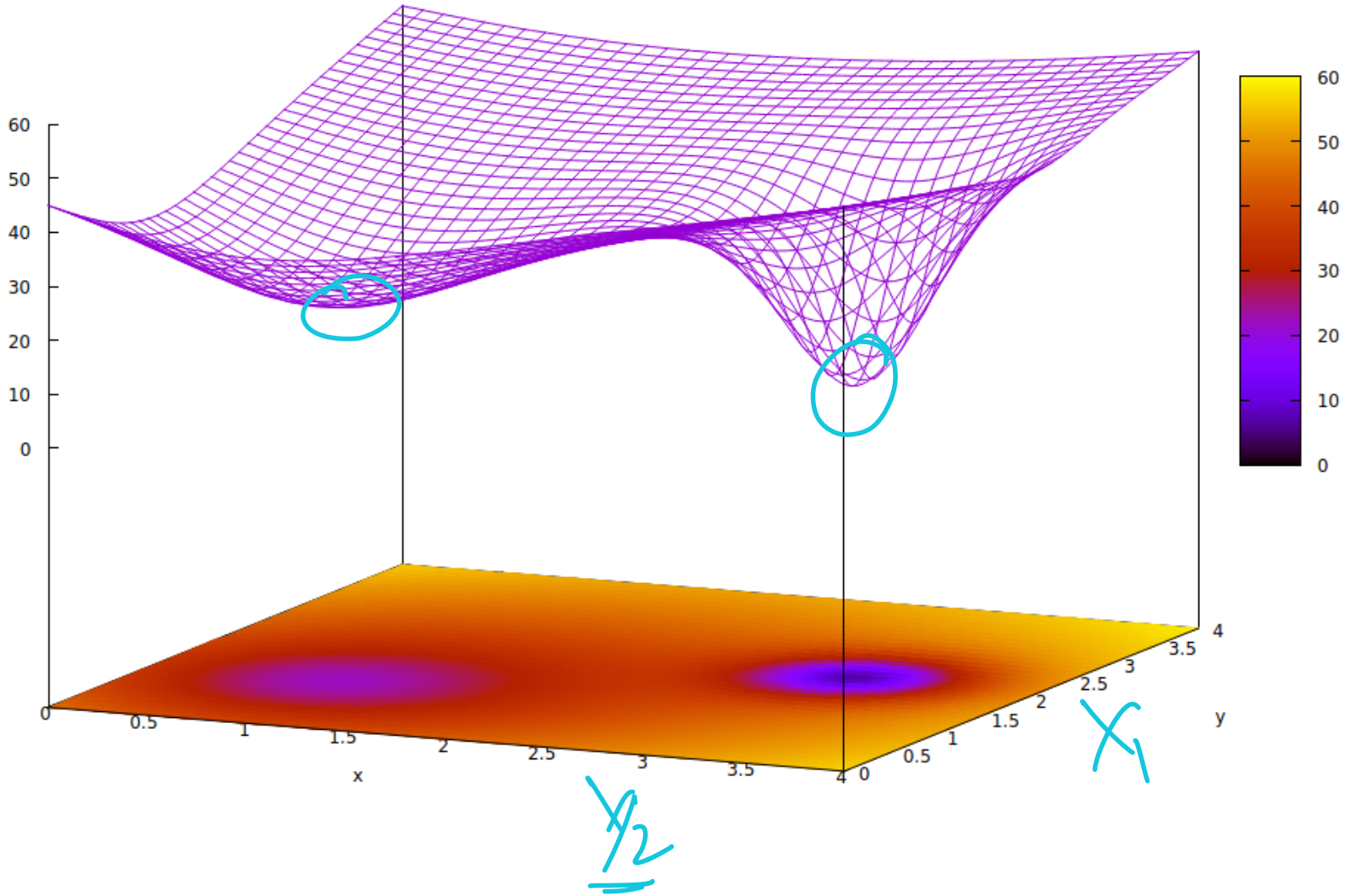
It can also be thought of as flow information from the error in the output (the loss function) down to the weights. Update the weights so we don't make **this error** next time around. Back prop is a way to do **gradient descent in neural networks!**

ICE #3

What powers the learning component when fine-tuning the parameters of an LLM in post-training? Select all that apply

1. High-quality data for fine-tuning
2. SGD
3. GPUs
4. Reinforcement learning

Good vs Bad Local minima



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. Batch-Size

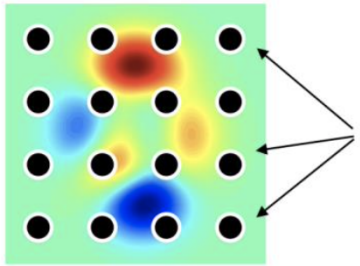
Hyper-parameters in Deep Learning

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. Batch-Size
6. Anything else?

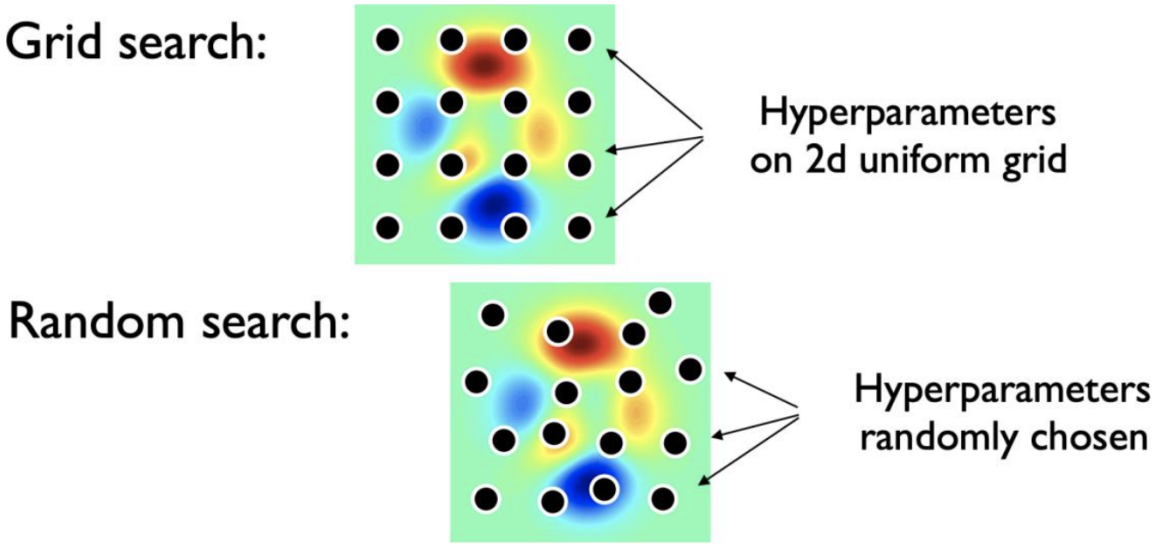
Hyper-parameter tuning methods

Grid search:



Hyperparameters
on 2d uniform grid

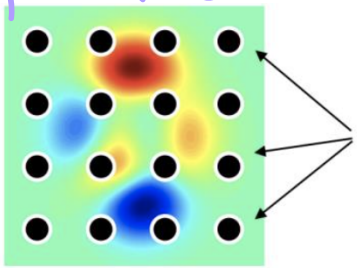
Hyper-parameter tuning methods



Hyper-parameter tuning methods

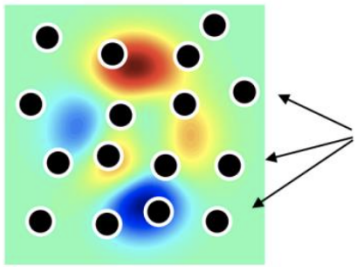
Hyper-params fine-tuned on validation
Parameter fine-tuned on Train
Results reported on Test / Eval / Held-out Set

Grid search:



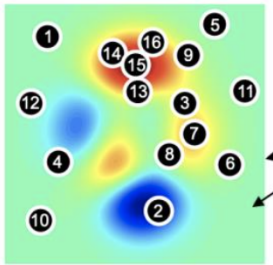
Hyperparameters on 2d uniform grid

Random search:



Hyperparameters randomly chosen

Bayesian Optimization:

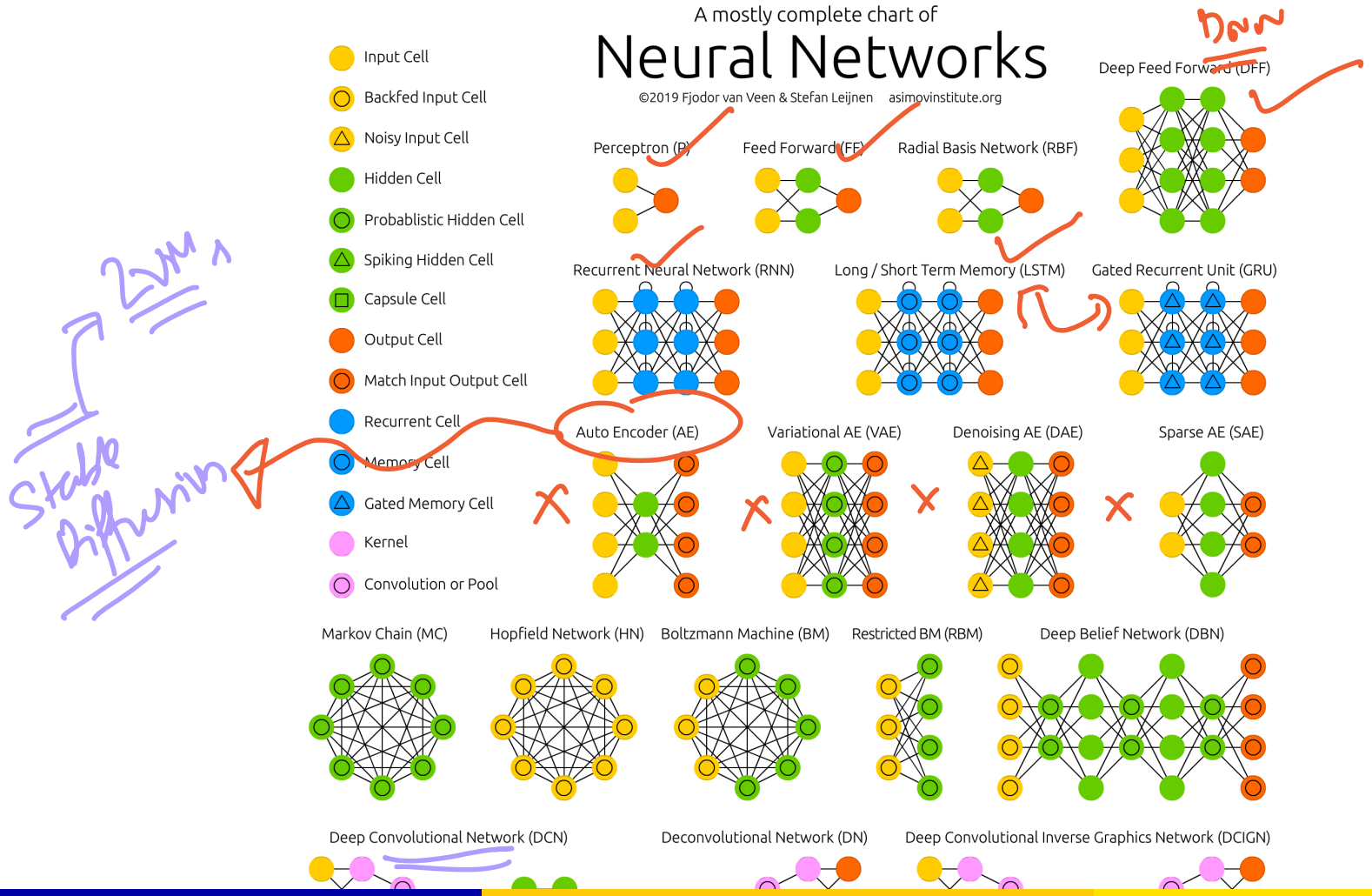


Hyperparameters ***adaptively*** chosen

More DL Architectures

Neural Networks Zoo

Zoo Reference



More DL Architectures

Neural Networks Zoo