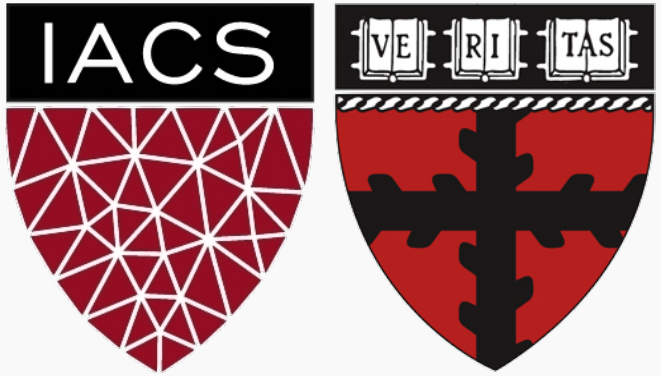
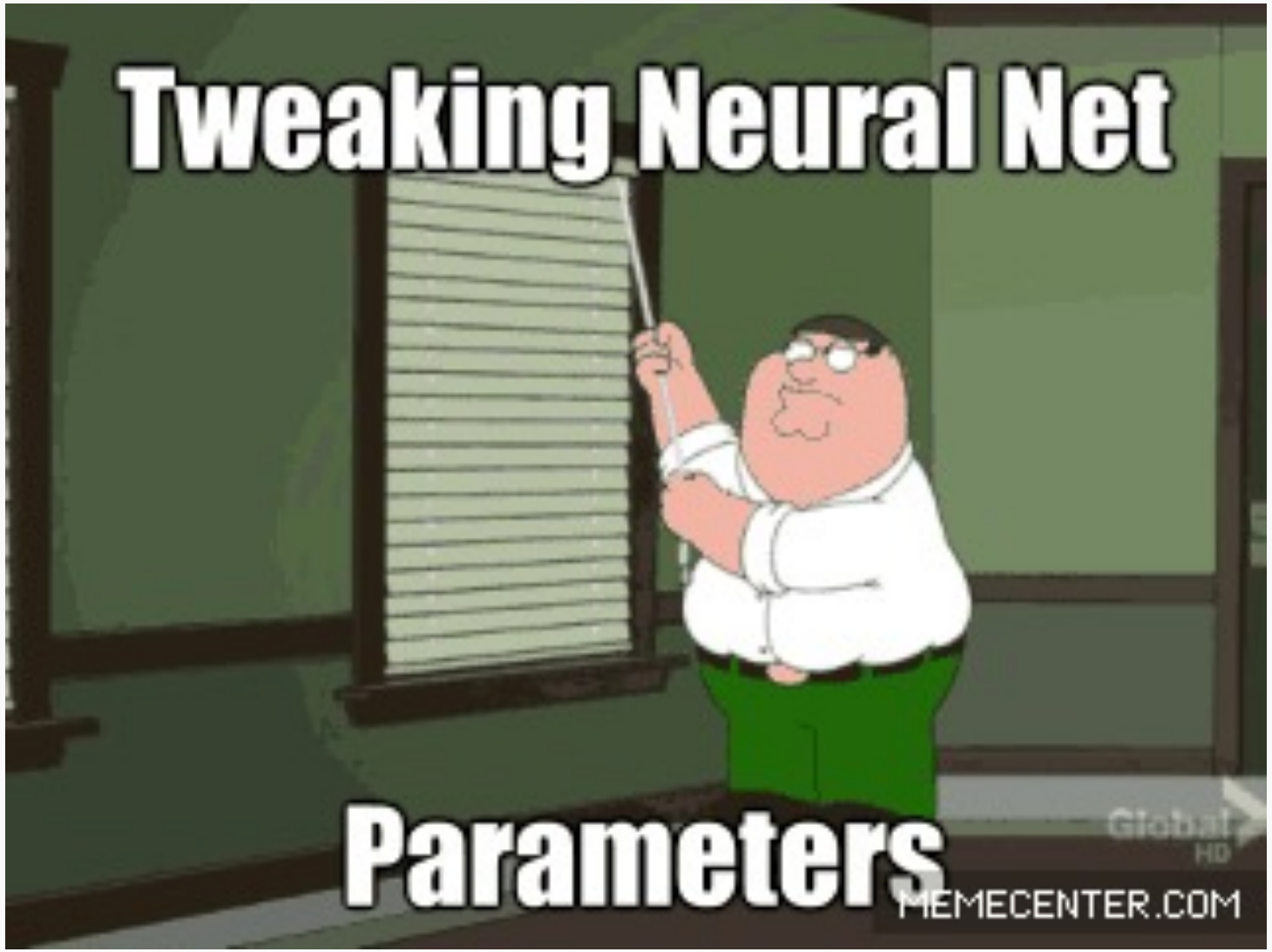
Fitting Neural Networks
Stochastic Gradient Descent

CS109B Data Science 2

Pavlos Protopapas, Mark Glickman




Tweaking Neural Net
Parameters
MEMECENTER.COM

Gradient Descent Considerations

- We still need to calculate the derivatives.
- We need to set the learning rate.
- Local vs global minima.
- **The full likelihood function includes summing up all individual '*errors*'. Sometimes this includes hundreds of thousands of examples.**

Stochastic Gradient Descent

Deep learning models crave data, and therefore datasets could be huge. The more the data, the more chances of a model to be good.

In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step. We do the following steps in **one epoch** for SGD:

1. Take one example
2. Feed it to Neural Network
3. Calculate its gradient
4. Use the gradient we calculated in step 3 to update the weights
5. Repeat steps 1–4 for all the examples in training dataset

Stochastic Gradient Descent

Epoch:

one forward pass and one backward pass of *all* the training examples

Why is this stochastic?

Order the algorithm sees data is random

Mini-Batch Stochastic Gradient Descent

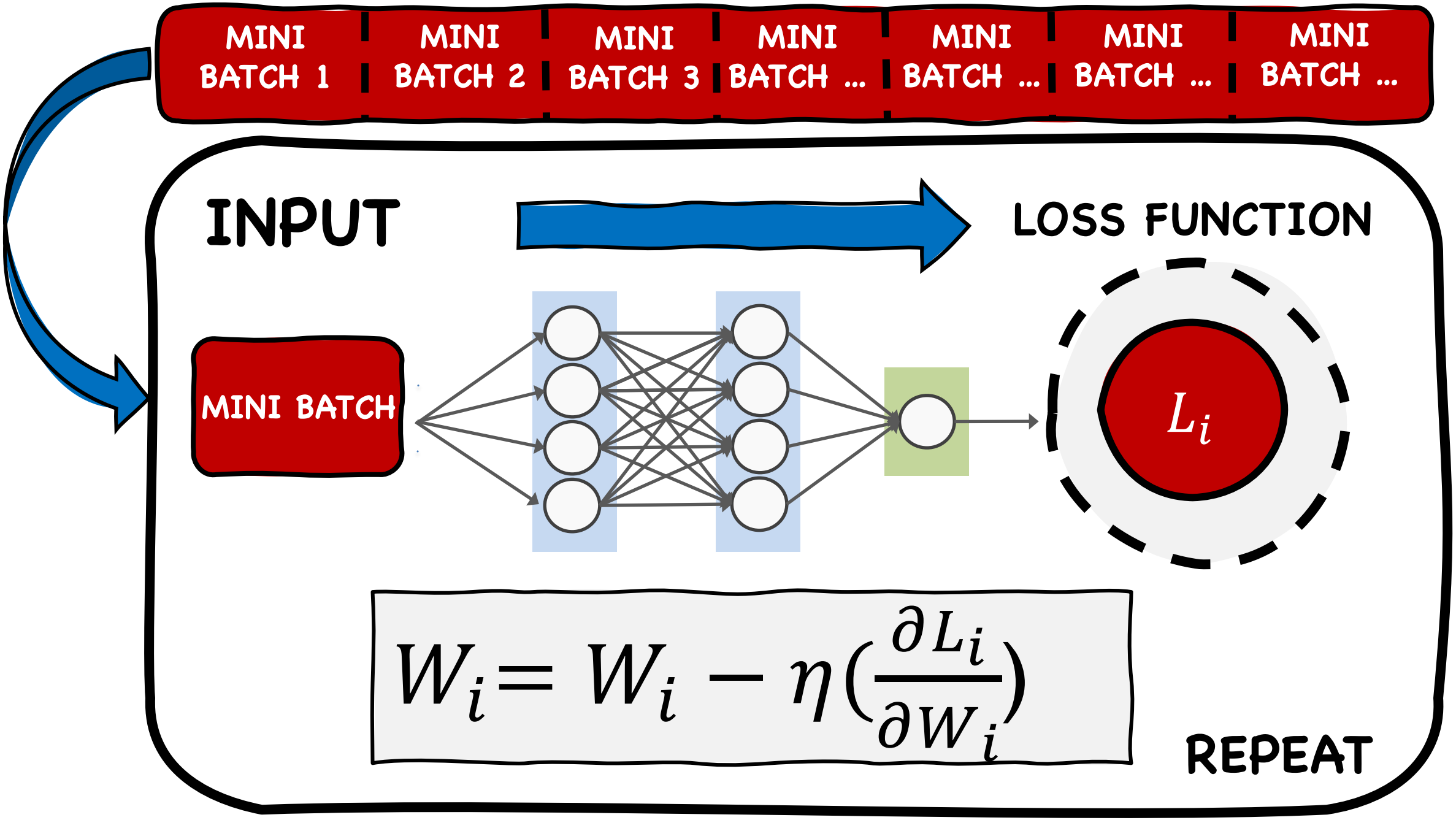
$$\mathcal{L}_i = -y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$

Instead of using one examples for every step, use a subset of them which we call mini batch.

We use only the points in the mini batch to calculate the loss function:

$$\mathcal{L}^k = -\sum_{i \in b^k} [y_i \log p_i + (1 - y_i) \log(1 - p_i)]$$

which approximates the full loss function.


MINI BATCH 1
MINI BATCH 2
MINI BATCH 3
MINI BATCH ...
MINI BATCH ...
MINI BATCH ...
MINI BATCH ...
INPUT
LOSS FUNCTION
MINI BATCH
L_i
$W_i = W_i - \eta\left(\frac{\partial L_i}{\partial W_i}\right)$
REPEAT

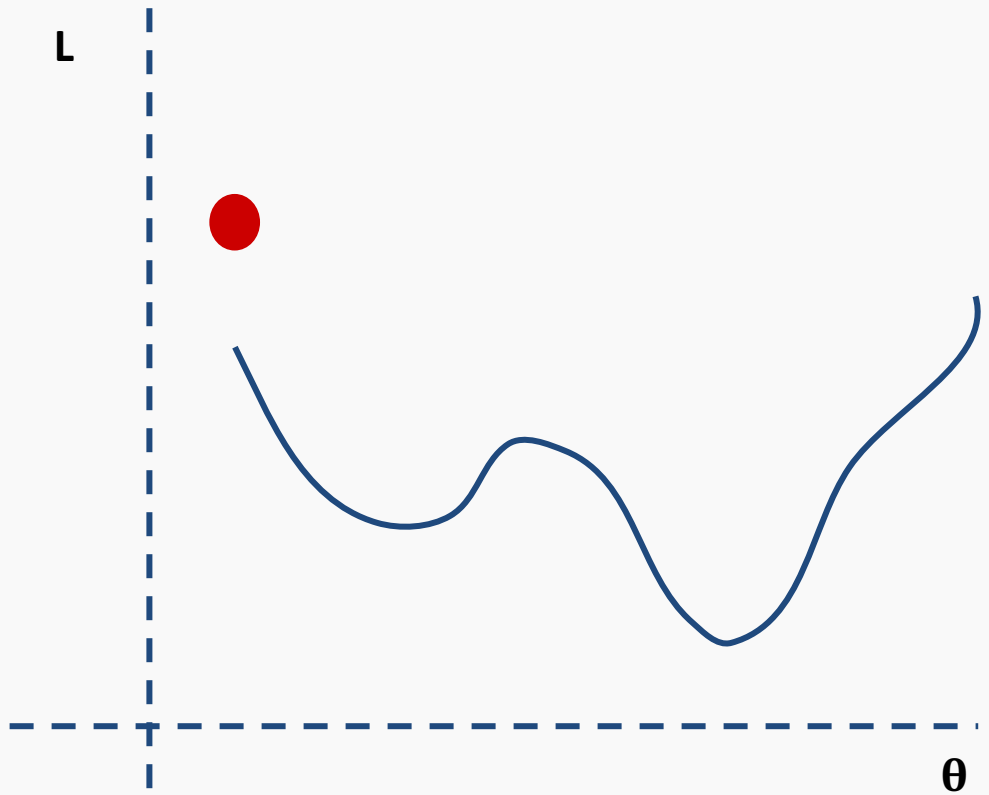
Mini-Batch Stochastic Gradient Descent

PSEUDO-CODE

1. Divide data into mini-batches
2. Pick a mini-batch
3. Feed it to Neural Network
4. Calculate the mean gradient of the mini-batch
5. Use the mean gradient we calculated in step 4 to update the weights
6. Repeat steps 2–5 for the mini-batches we created

One epoch

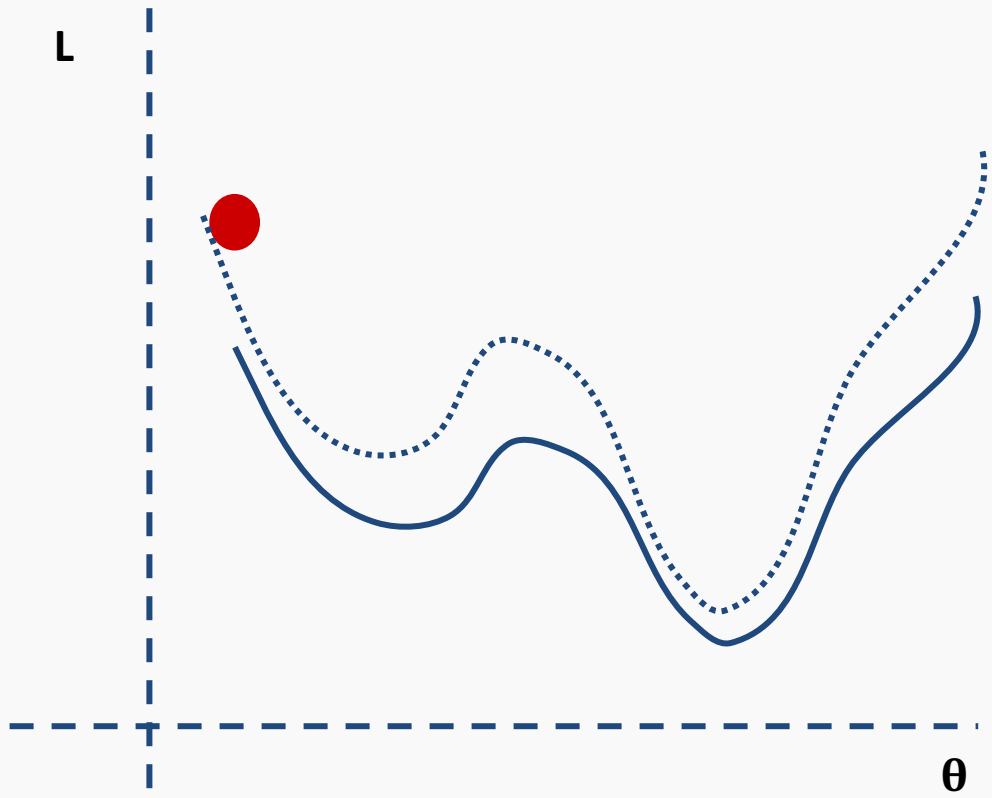
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

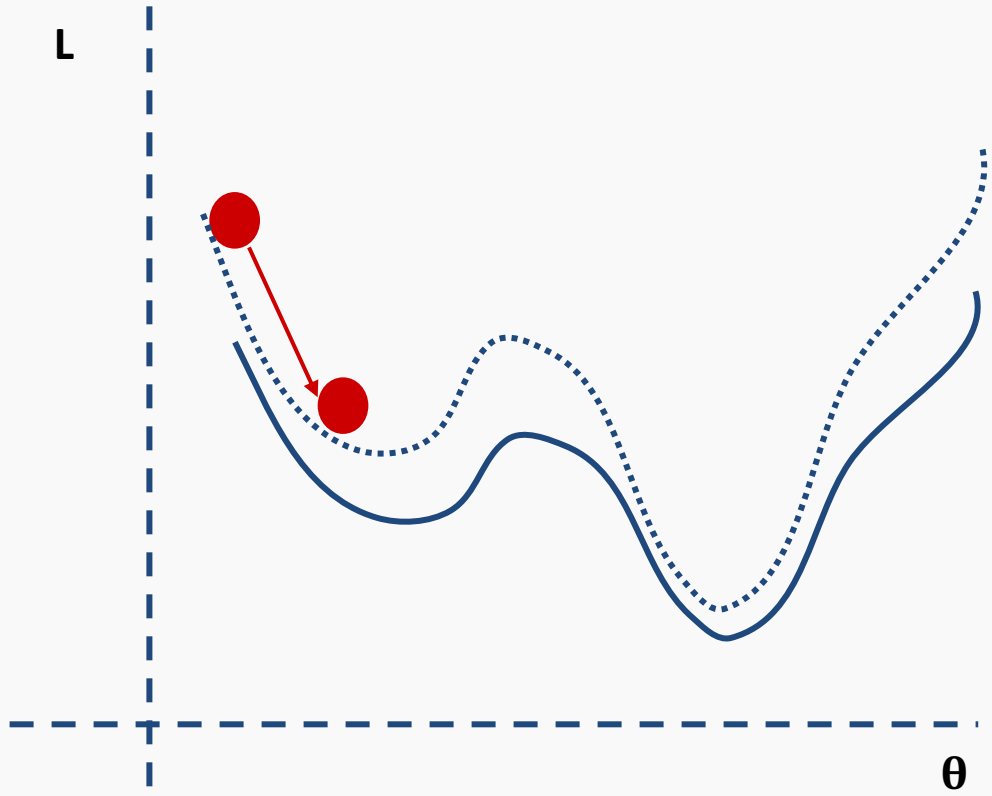
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

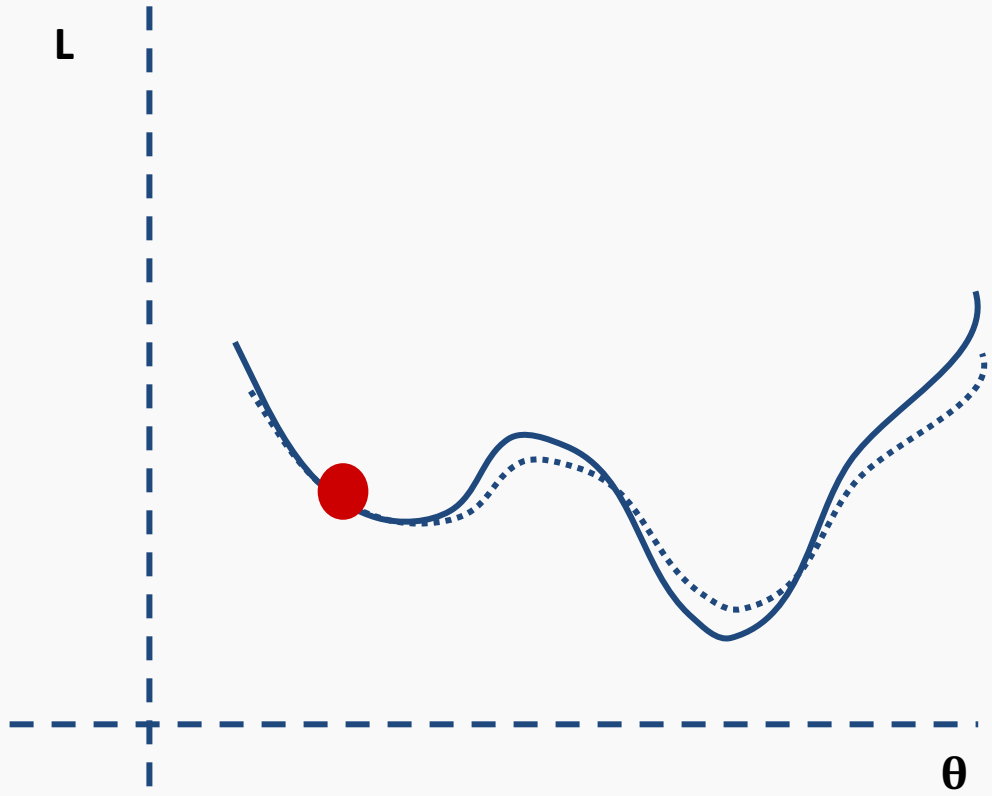
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

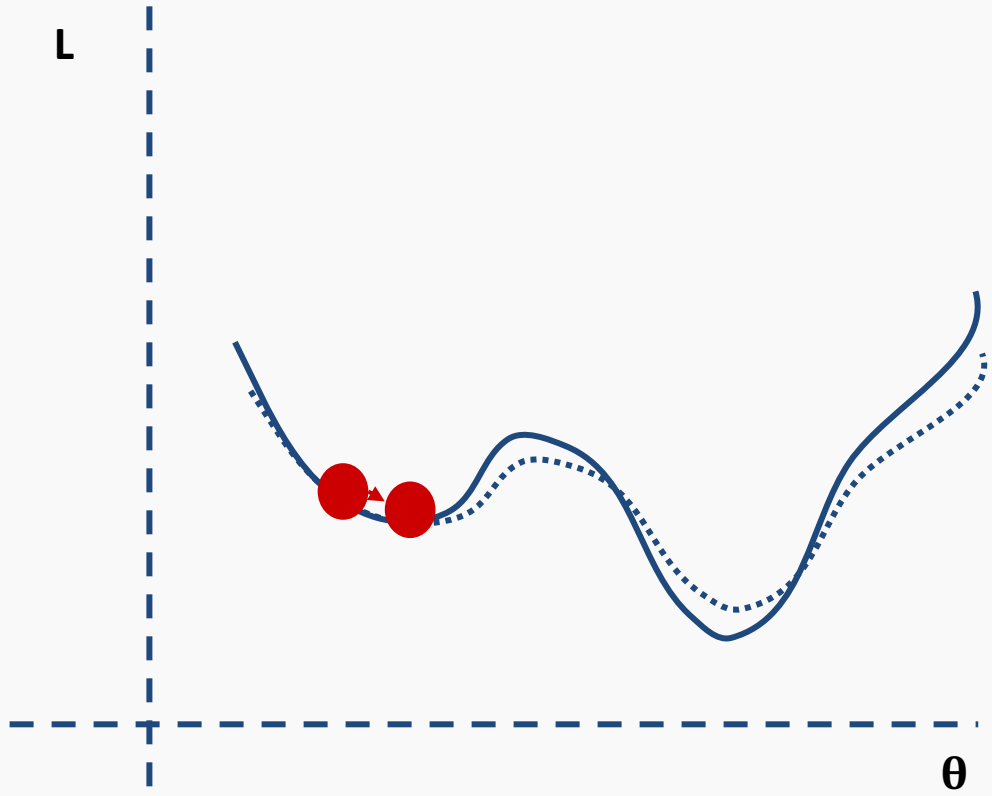
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

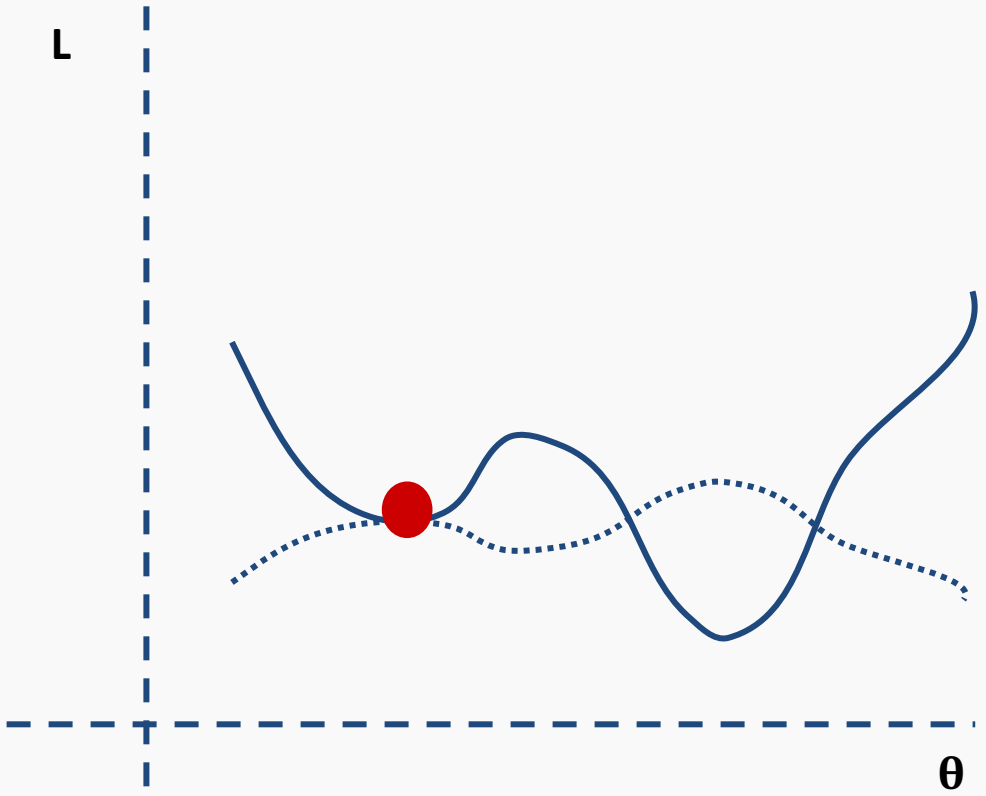
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

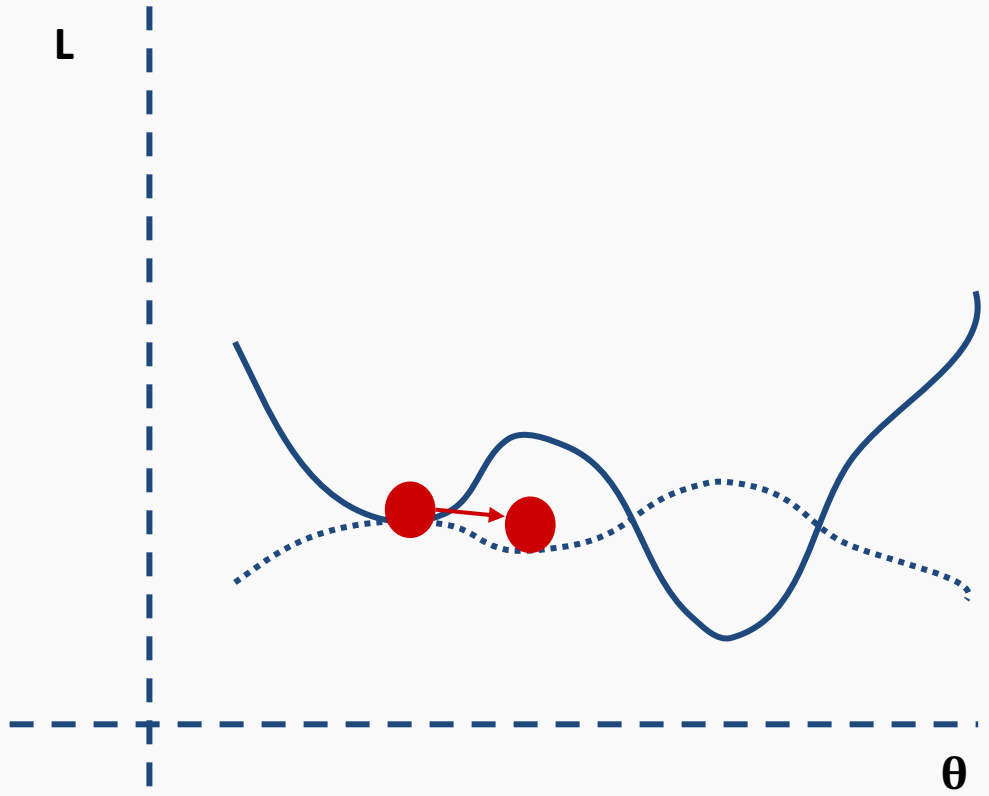
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

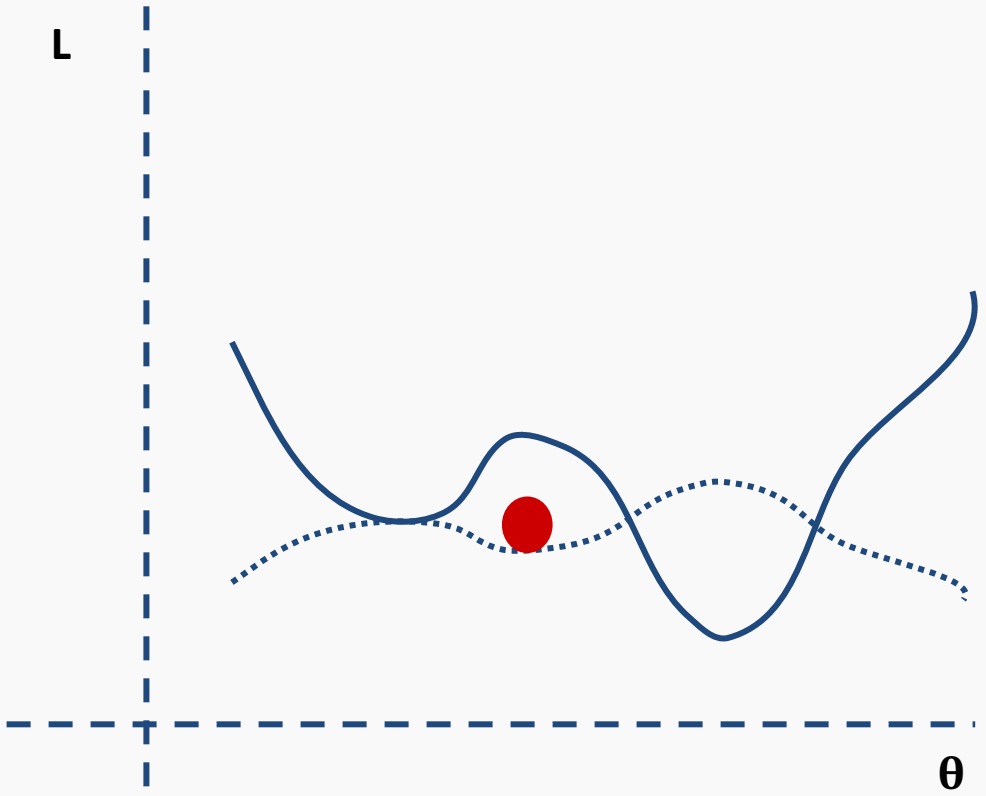
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

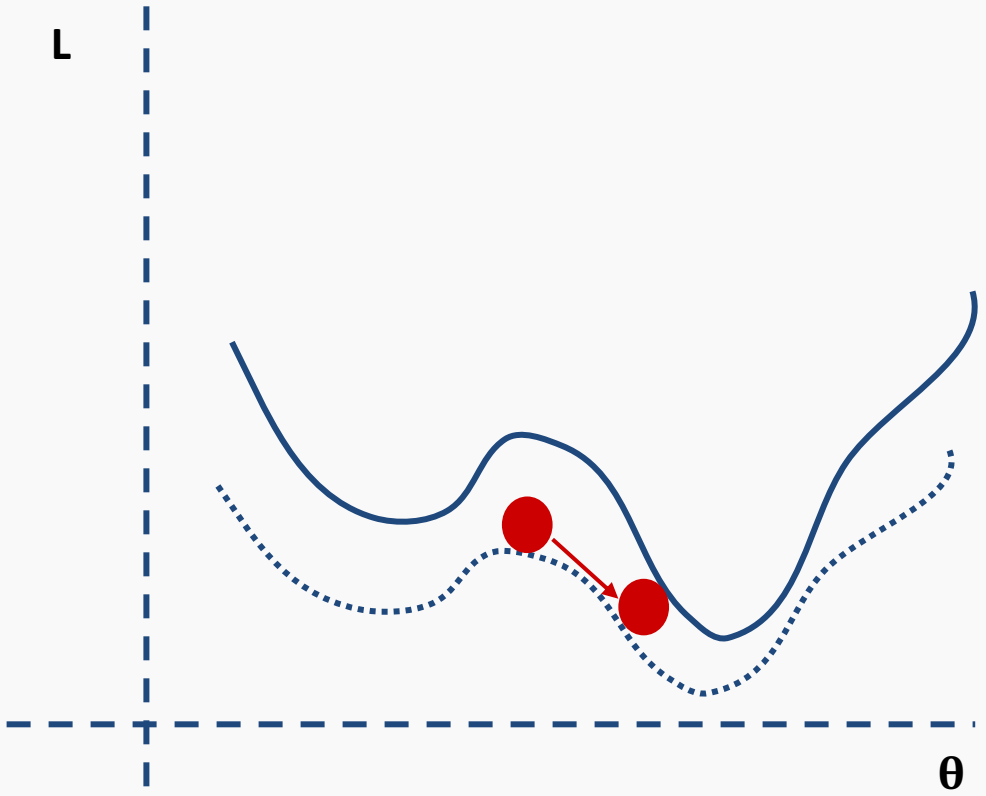
Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

Batch and Stochastic Gradient Descent



Full Likelihood:

Batch Likelihood:

Mini-Batch Stochastic Gradient Descent

Mini-batch size:

Often called batch size. In general, we choose powers of 2 and that is because of memory considerations, especially if we use GPUs

Large batch size it is equivalent to gradient descent (aka as batch gradient descent).

Very small batch size is like the vanilla SGD though is very jumpy but can avoid local minima.