Training a Neural Network

Michael Shao

Machine Learning

- Looking for the function

$$f(x_0, x_1, \ldots) = \hat{y}$$

We’ve defined the structure

- but what about the parameters?

$$w_0,\ w_1, w_2, \dots$$

$$f_\theta$$

$$\theta$$

$$b_0,\ b_1, b_2, \dots$$

What's the best θ?

- We can evaluate the performance of the model based on how closely the output follows the distribution in the **labeled** training set.

Comparing distributions?

- Defining a loss function

$L(\theta) =$ Difference

be

$f_\theta(X)$ and y

Optimization

Gradient Descent

Minimizing the loss function

$$\theta^\star = \arg\min_\theta L(\theta)$$

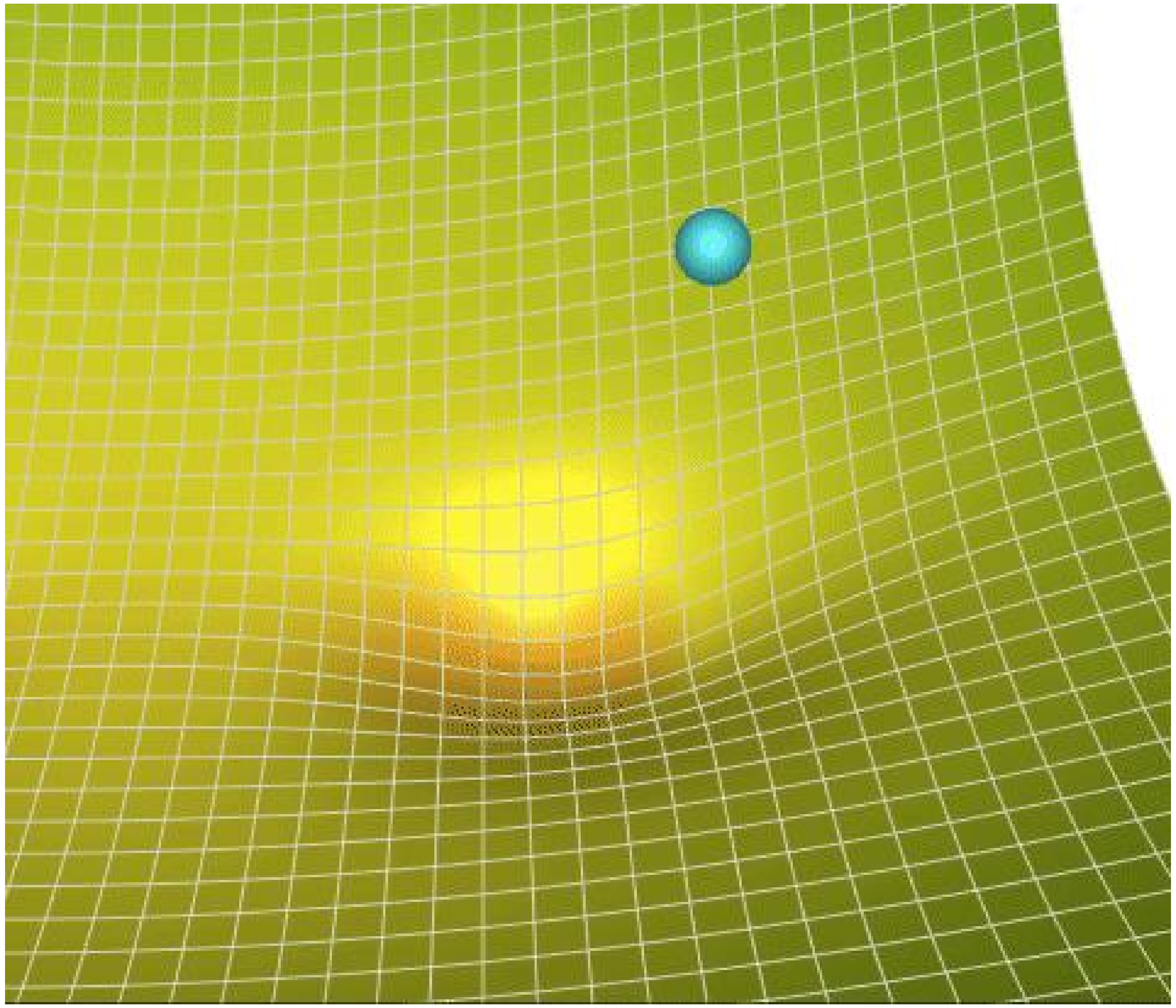
How do we do this?

Gradient Descent

- Descending a slope:

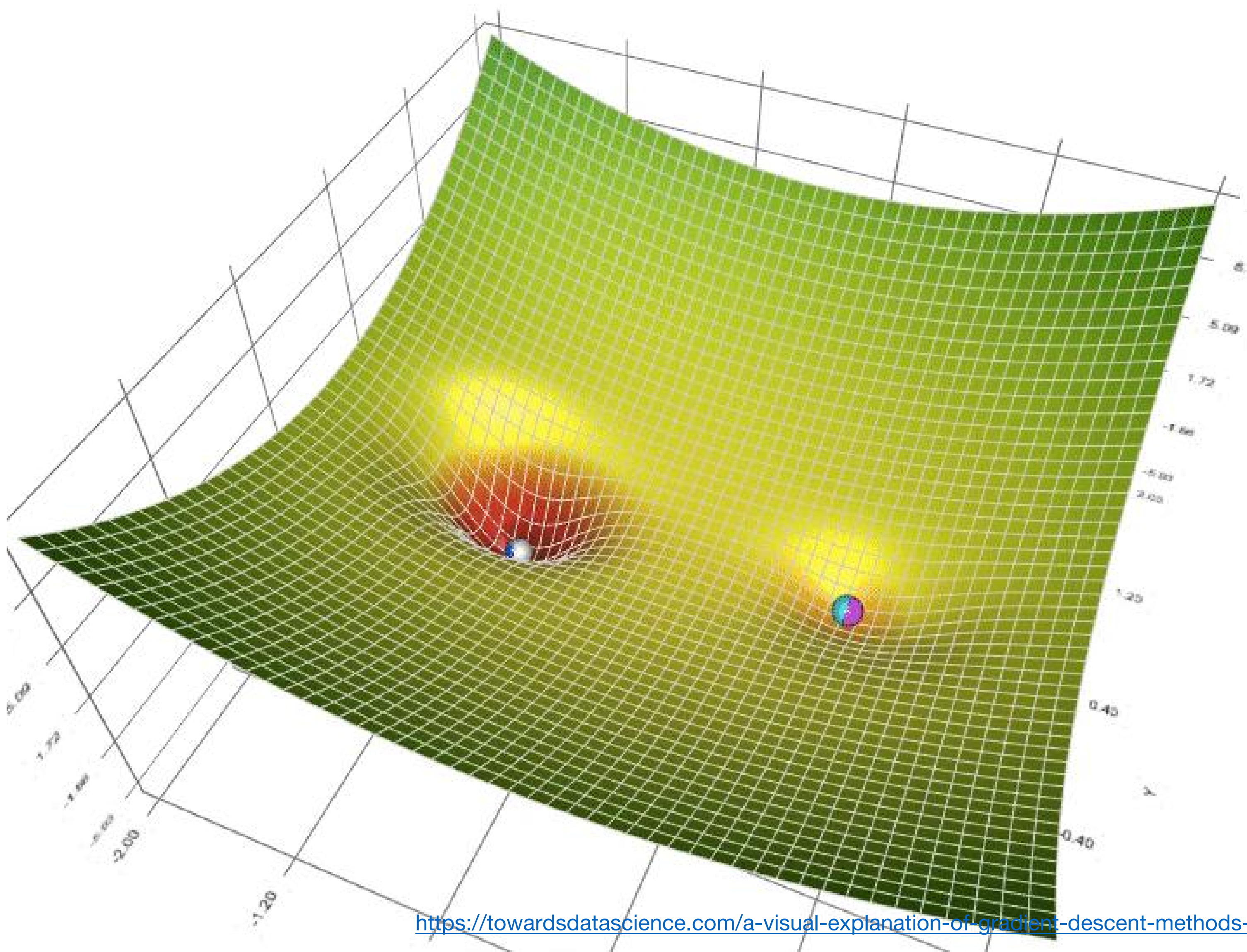
$$\theta \leftarrow \theta - \eta \nabla_{\theta} L(\theta)$$

https://uclaacm.github.io/gradient-descent-visualiser/#playground



https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898

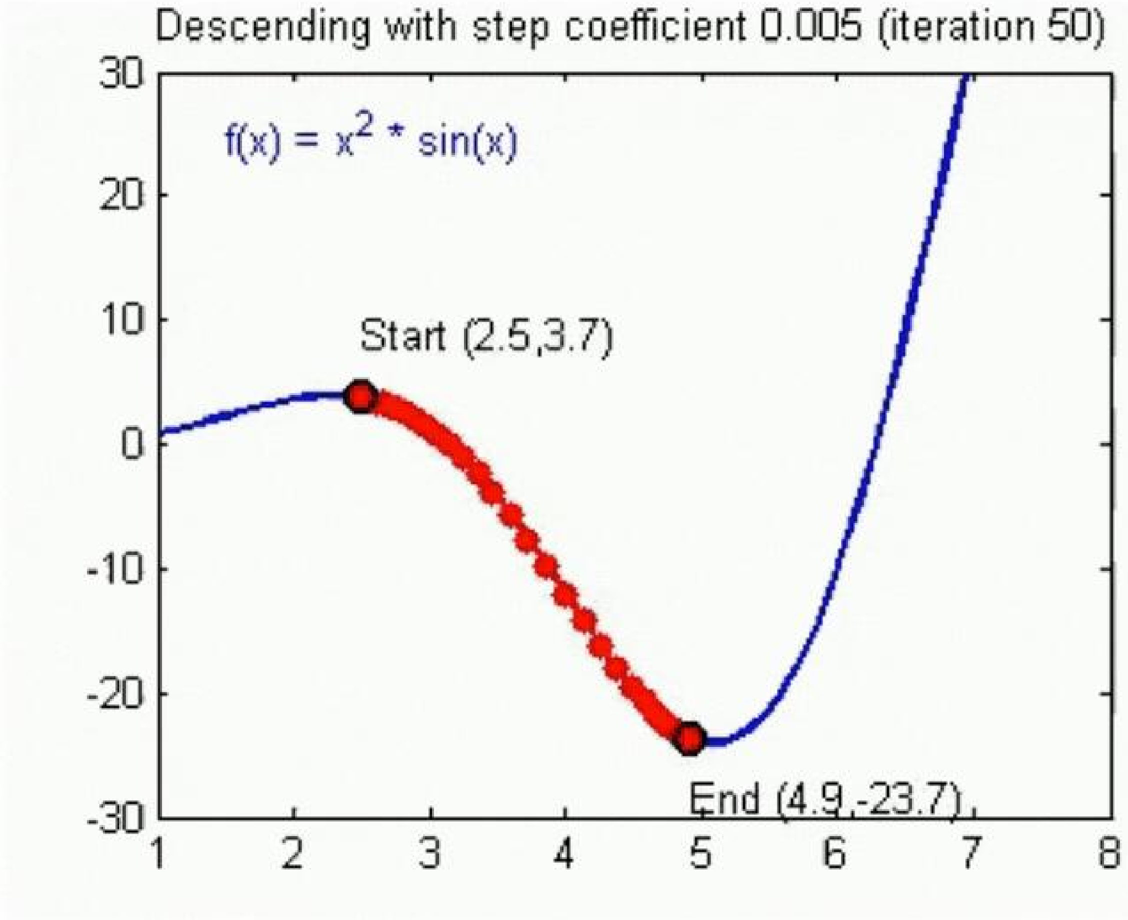
Local Minima

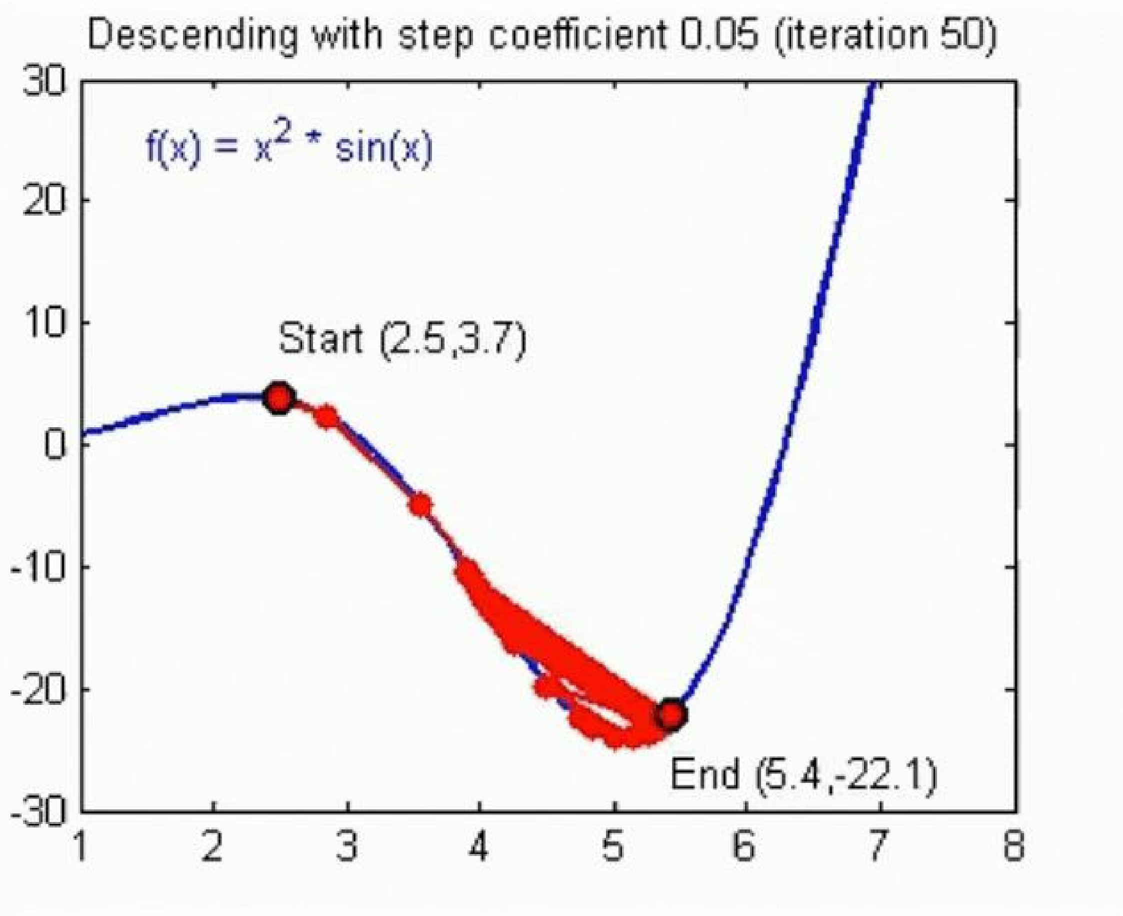


https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898

Convergence

Divergence





https://towardsai.net/p/machine-learning/analysis-of-learning-rate-in-gradient-descent-algorithm-using-python

Step, Batch, Epoch

- Split training data randomly into batches based on batch size.
- Calculate loss on batch, perform gradient descent = 1 step
- Iterate over all batches = 1 epoch

Loss Functions

The Training Criteria

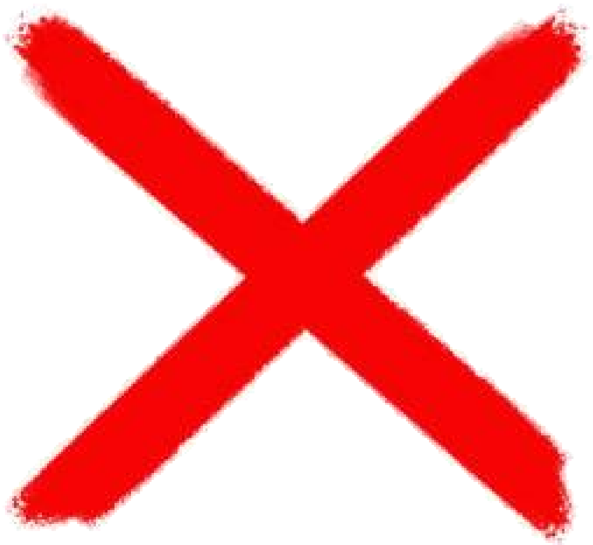
Mean Squared Error Loss

criterion = torch.nn.MSELoss()

$$\frac{1}{N}\sum_{i=1}^{N}(f_{\theta}(x_i) - y_i)^2$$

Loss for Classification Tasks?

- Number of misclassifications?
- Inaccuracy?



What does a classification model output?

0	0.2
1	1.3
2	(25.7)
3	−11
4	102

But each output has no “meaning”. Their values are only relative to the oth

Introducing… Softmax Activation

- Map values to (0, 1) with a sum of 1.

$$\sigma\begin{pmatrix} 0.2 \\ 1.3 \\ 25.7 \\ -11 \\ 102 \end{pmatrix} = \begin{pmatrix} 6.15 \times 10^{-45} \\ 1.85 \times 10^{-44} \\ 7.3 \times 10^{-34} \\ 8.41 \times 10^{-50} \\ 1 \end{pmatrix}$$

Now, these values can be interpreted as probabilities: $\begin{pmatrix} p_0 \\ p_1 \\ \dots \\ p_C \end{pmatrix}$.

Cross Entropy Loss

- We want the corresponding probability to be close to 1, so…
- We want the log probability to be close to 0. In other words,
- We want to minimize $-\log(p_y)$
- $L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\log(p_{i,y_i})$

Cross Entropy Loss

- criterion = torch.nn.CrossEntropyLoss()

- criterion = torch.nn.BCELoss() # For Binary CE

Calculating The Gradient

Back Propagation… In very very very simple words

Back Propagation

- The gradient is propagated from the last layer all the way back to the first layer.
- Keywords: partial derivatives, chain rule
- https://www.youtube.com/watch?v=Ilg3gGewQ5U