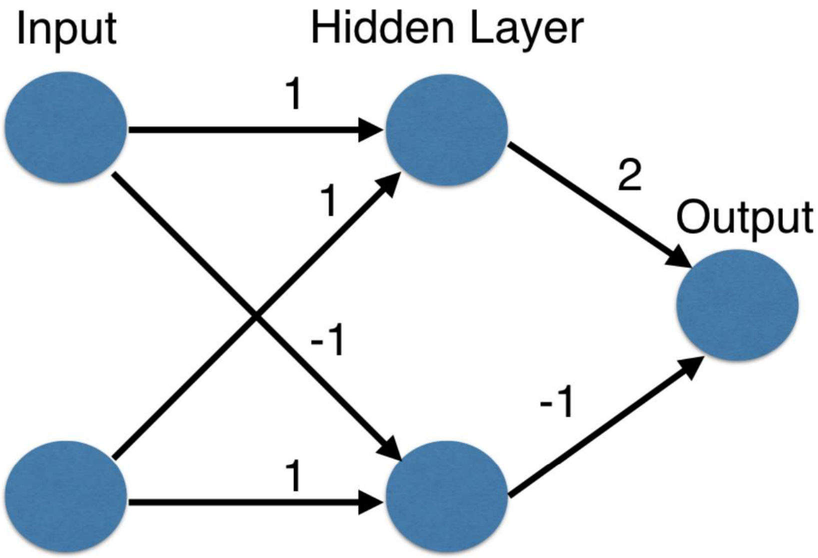
The need for optimization

INTRODUCTION TO DEEP LEARNING IN PYTHON

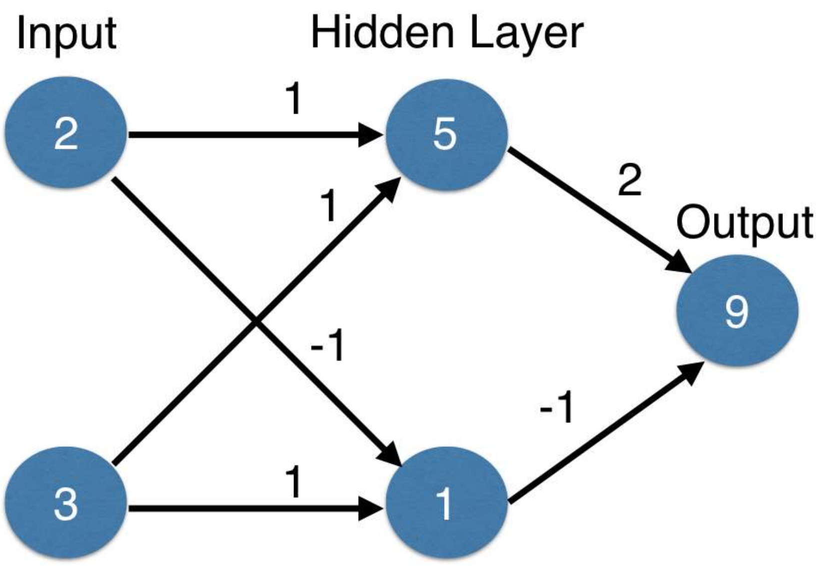
Dan Becker

Data Scientist and contributor to Keras and TensorFlow libraries

A baseline neural network

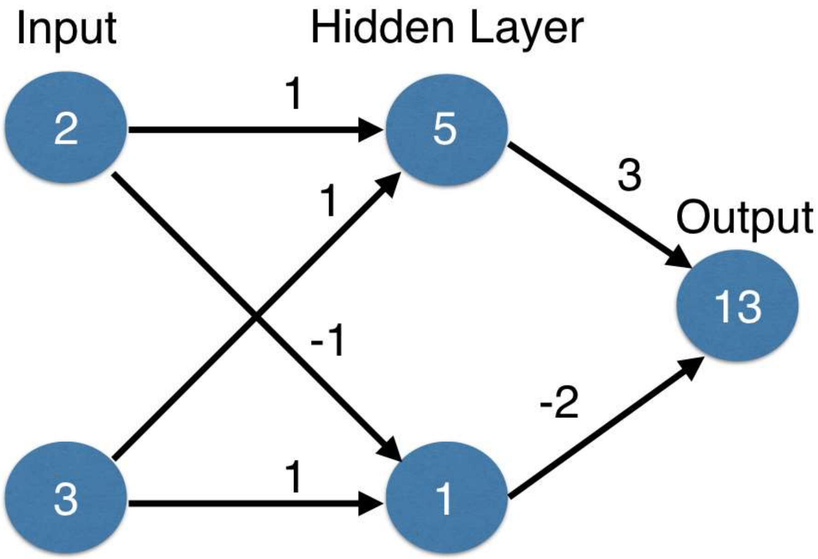


A baseline neural network



- Actual Value of Target: 13
- Error: Predicted - Actual = -4

A baseline neural network



- Actual Value of Target: 13
- Error: Predicted - Actual = 0

Predictions with multiple points

- Making accurate predictions gets harder with more points
- At any set of weights, there are many values of the error
- ... corresponding to the many points we make predictions for

Loss function

- Aggregates errors in predictions from many data points into single number
- Measure of model's predictive performance

Squared error loss function

Prediction	Actual	Error	Squared Error
10	20	-10	100
8	3	5	25
6	1	5	25

Squared error loss function

Prediction	Actual	Error	Squared Error
10	20	-10	100
8	3	5	25
6	1	5	25

- Total Squared Error: 150
- Mean Squared Error: 50

Loss function

- Lower loss function value means a better model
- Goal: Find the weights that give the lowest value for the loss function
- Gradient descent

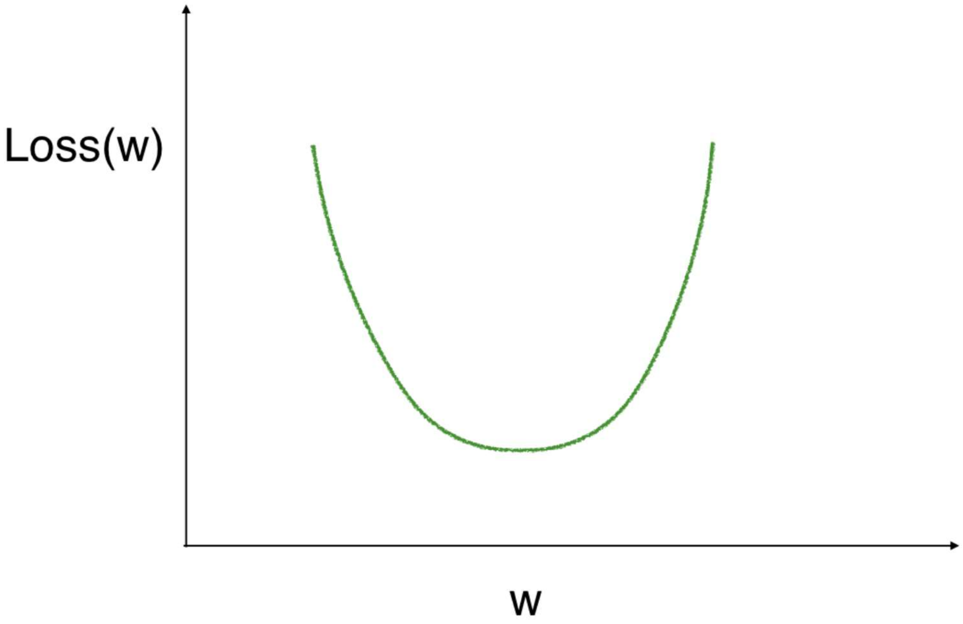
Gradient descent

- Imagine you are in a pitch dark field
- Want to find the lowest point
- Feel the ground to see how it slopes
- Take a small step downhill
- Repeat until it is uphill in every direction

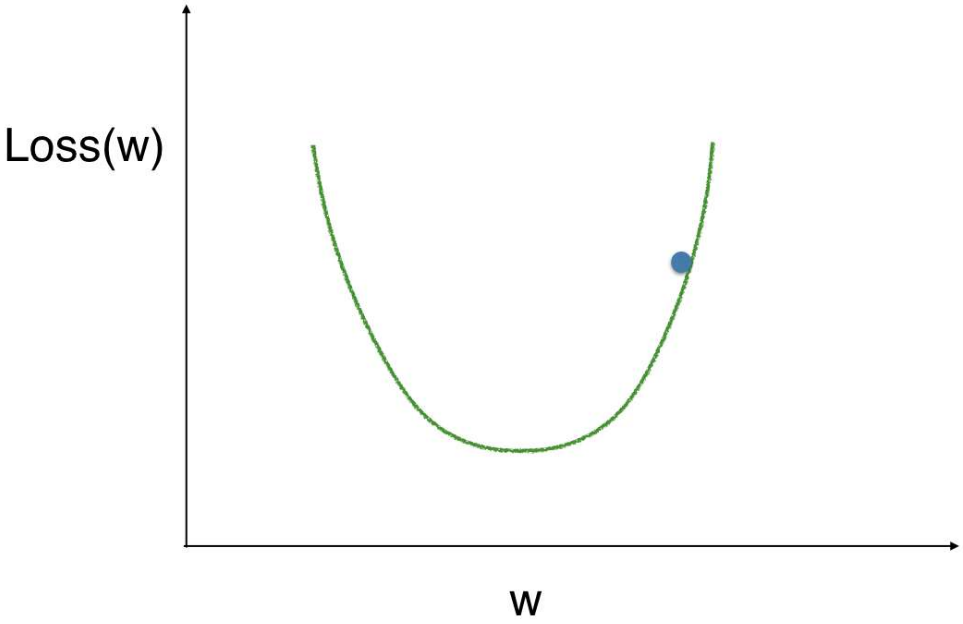
Gradient descent steps

- Start at random point
- Until you are somewhere flat:
  - Find the slope
  - Take a step downhill

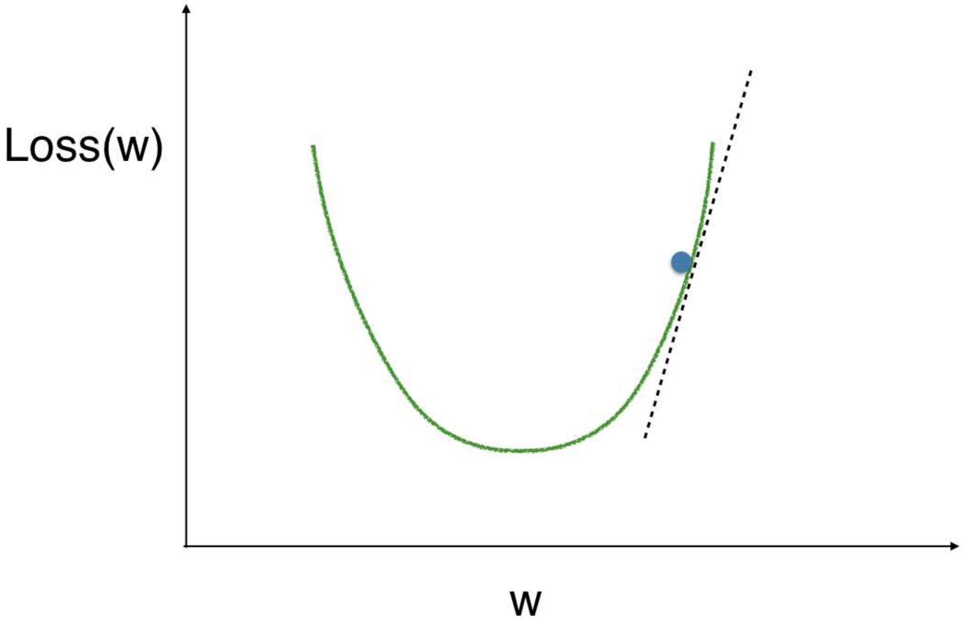
Optimizing a model with a single weight



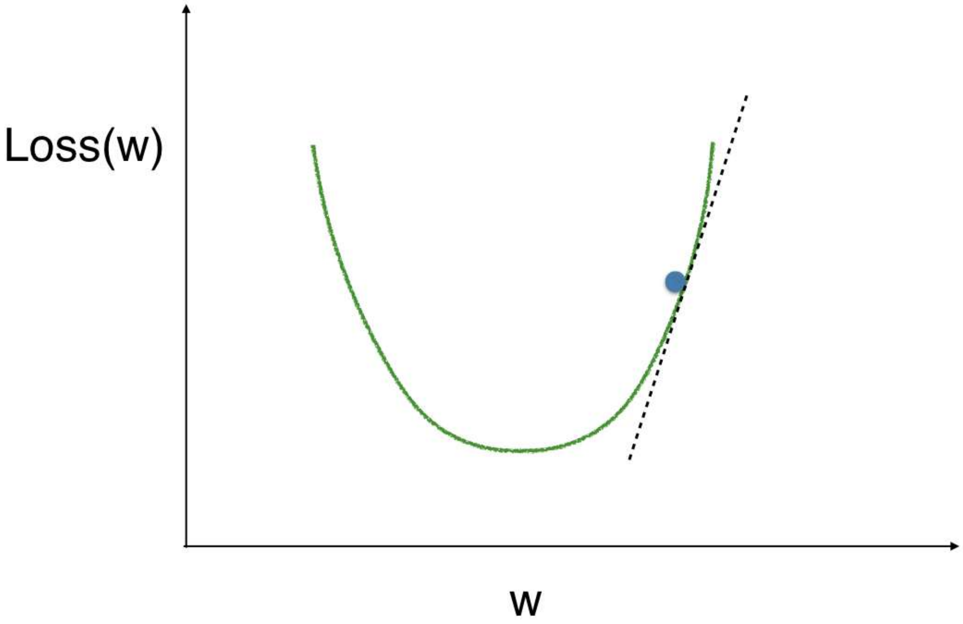
Optimizing a model with a single weight



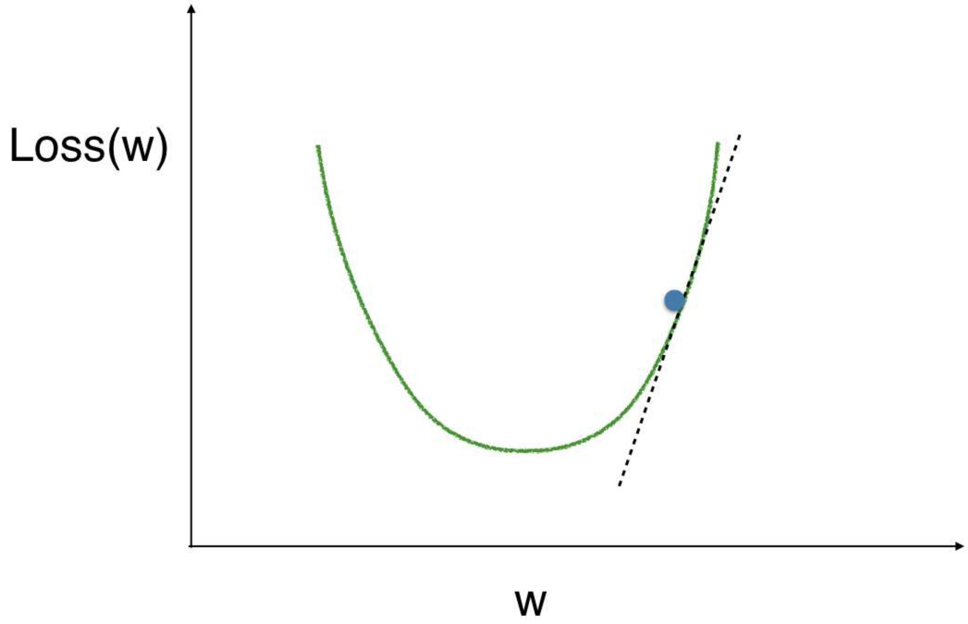
Optimizing a model with a single weight



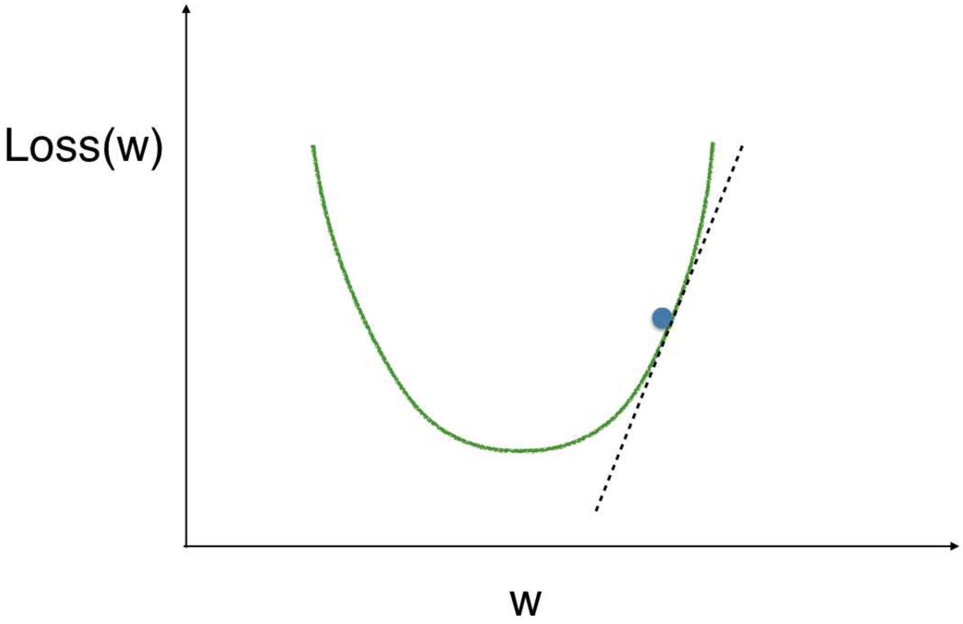
Optimizing a model with a single weight



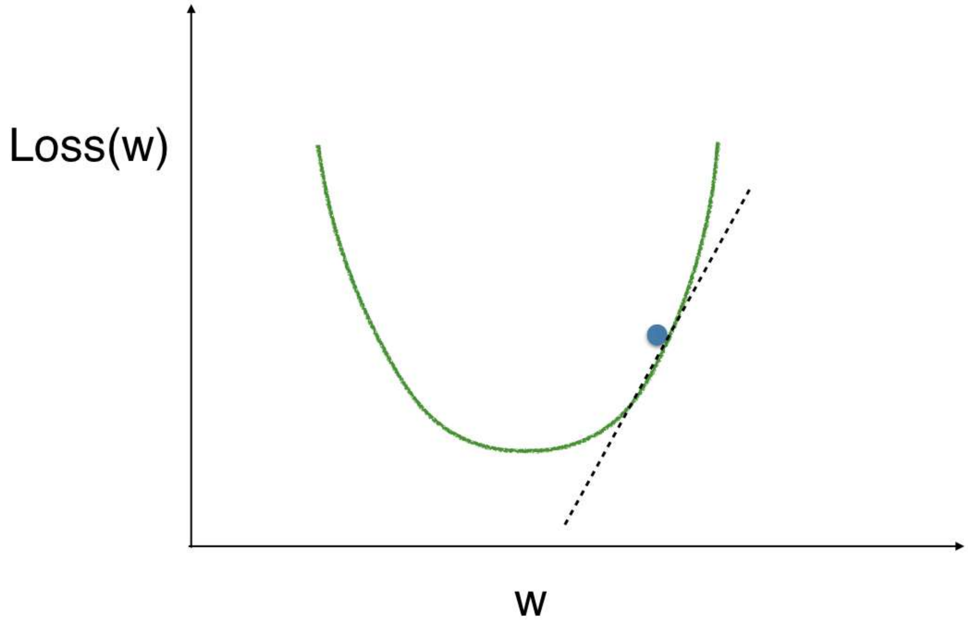
Optimizing a model with a single weight



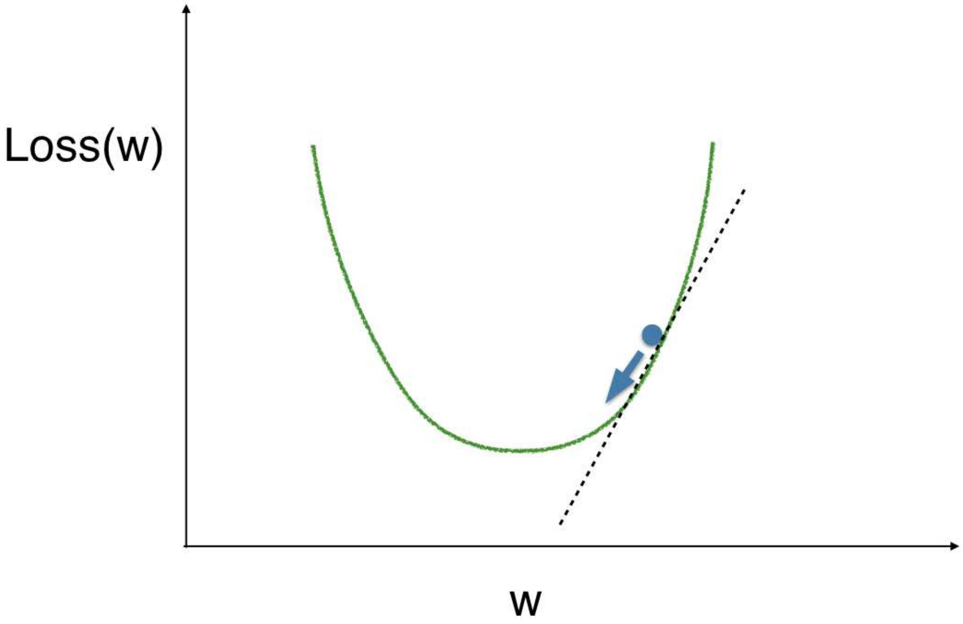
Optimizing a model with a single weight



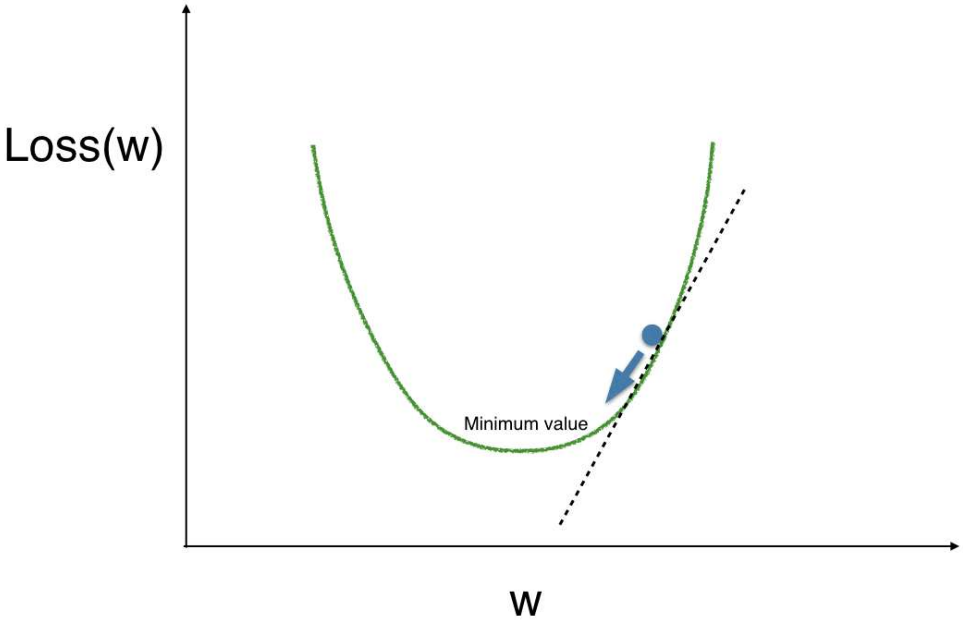
Optimizing a model with a single weight



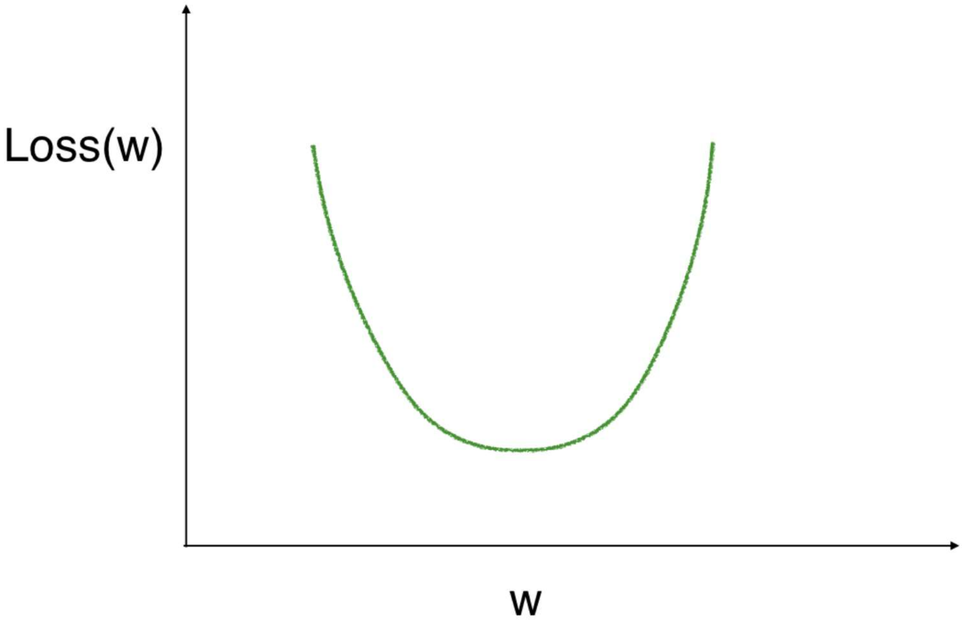
Optimizing a model with a single weight



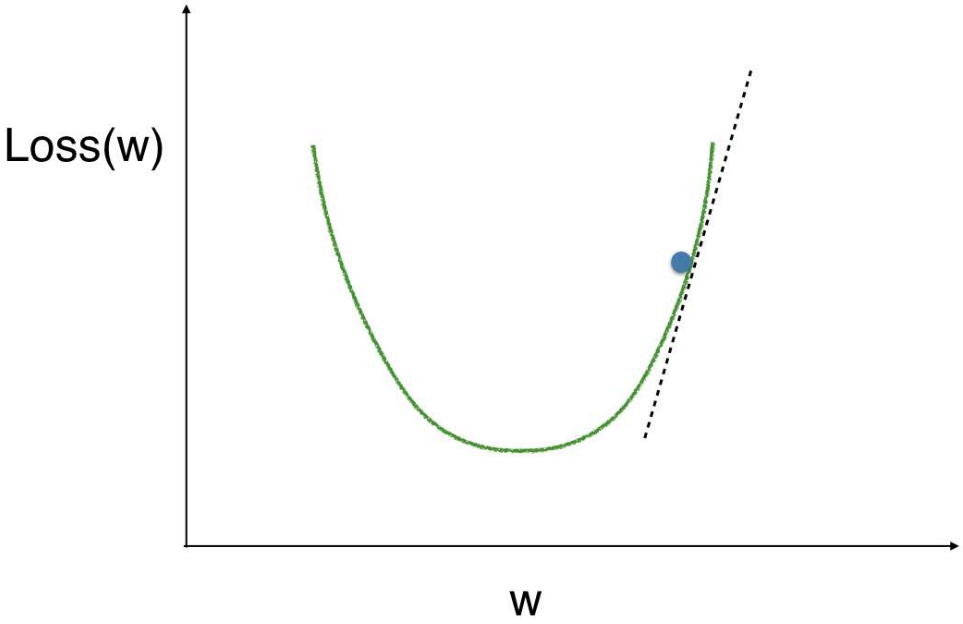
Optimizing a model with a single weight



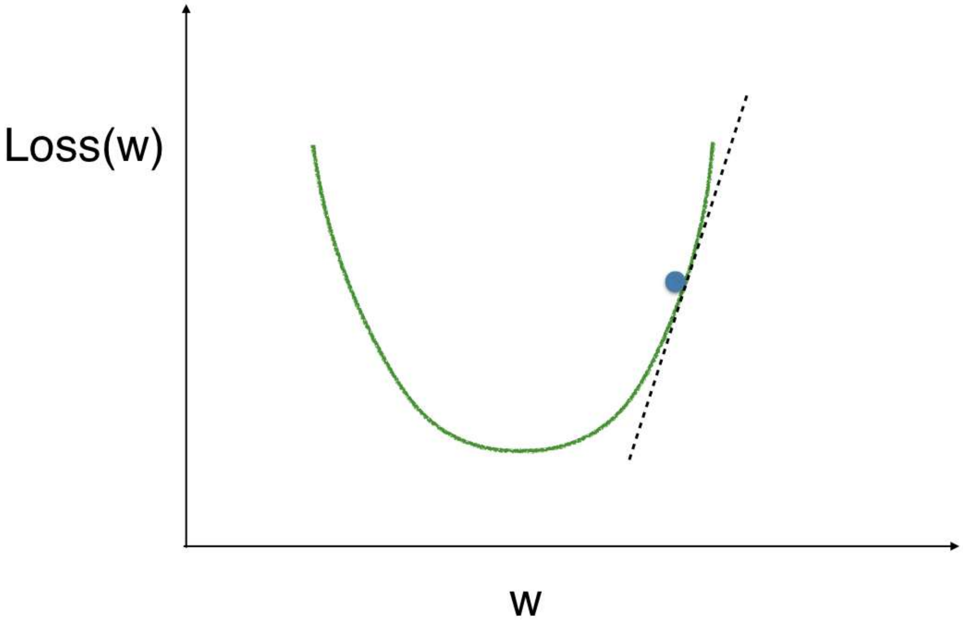
Gradient descent



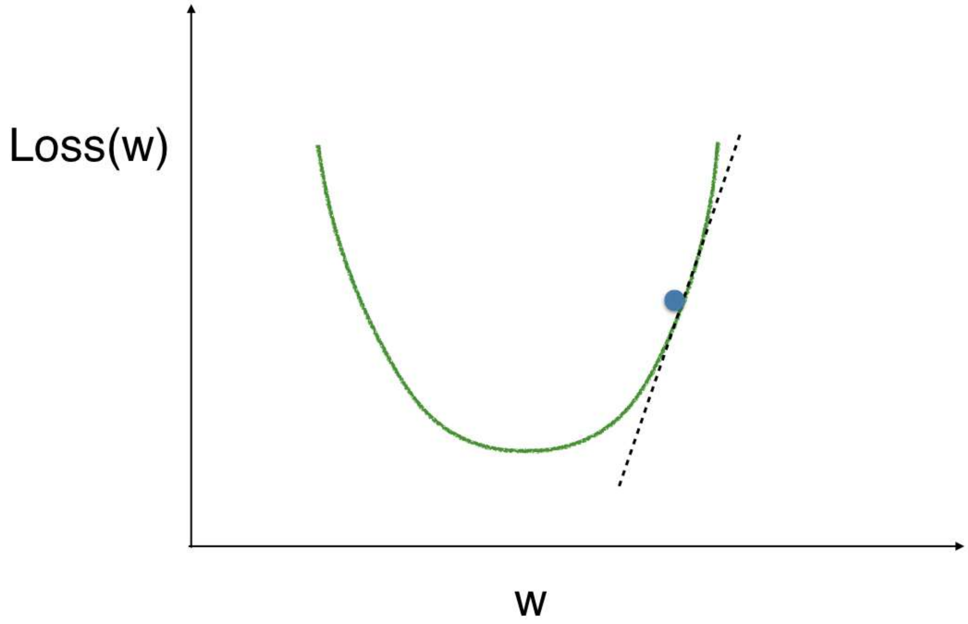
Gradient descent



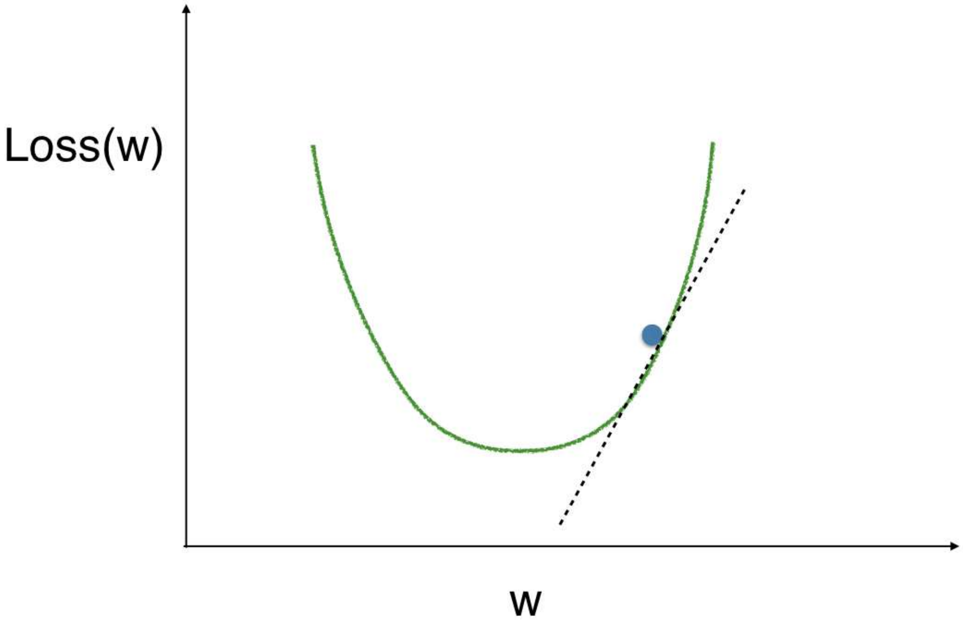
Gradient descent



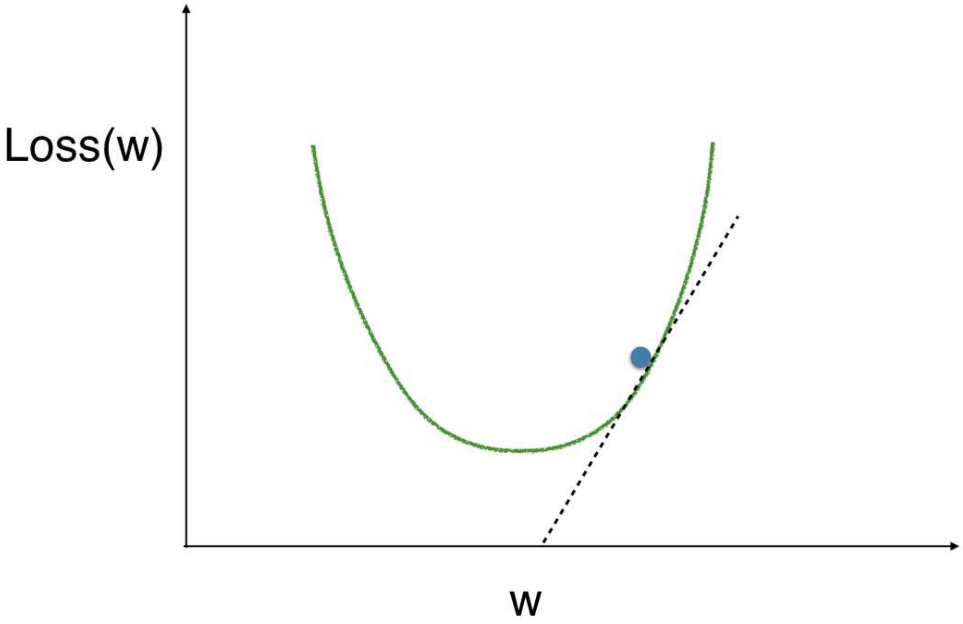
Gradient descent



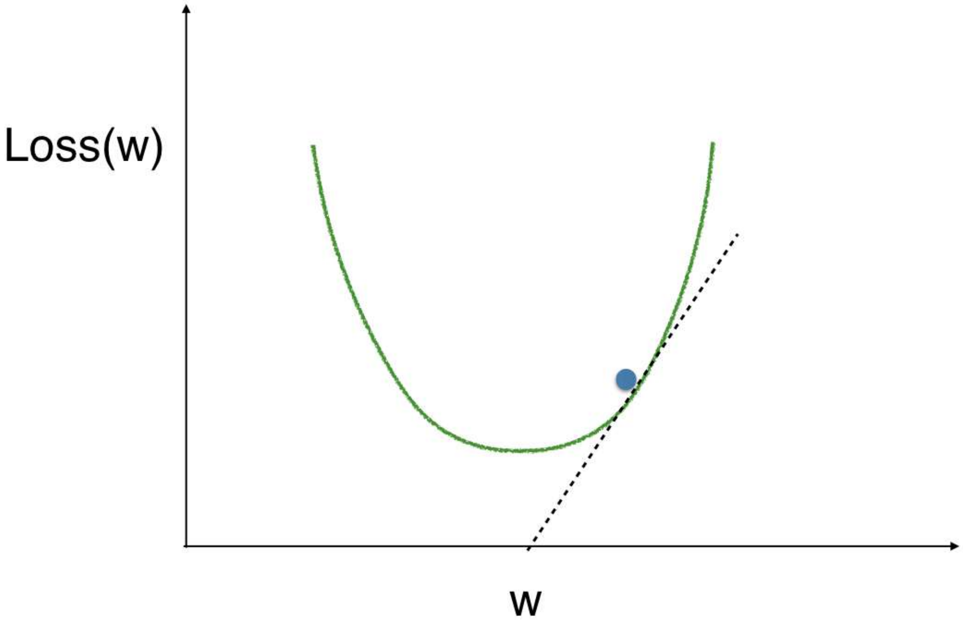
Gradient descent



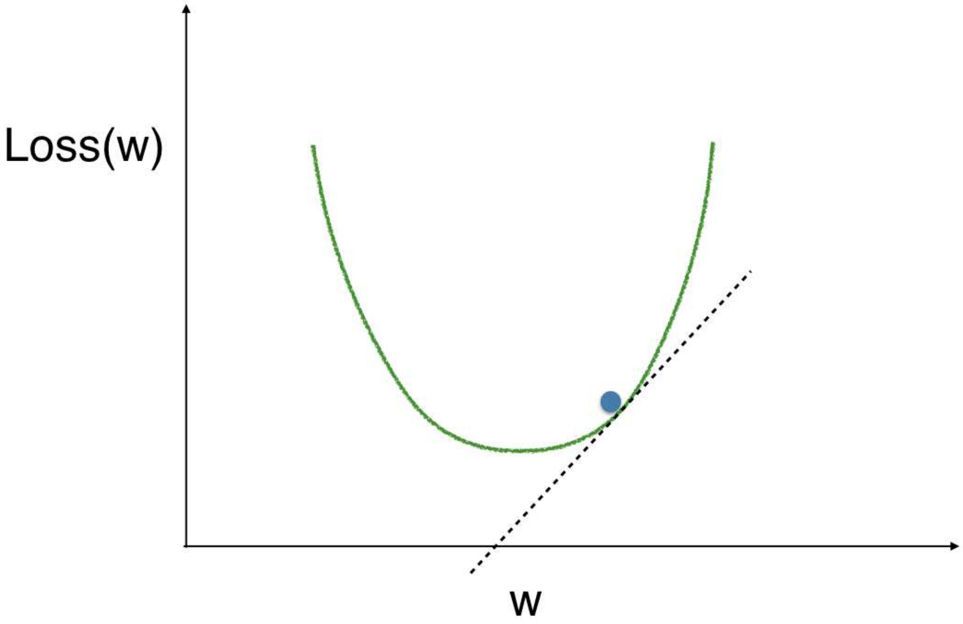
Gradient descent



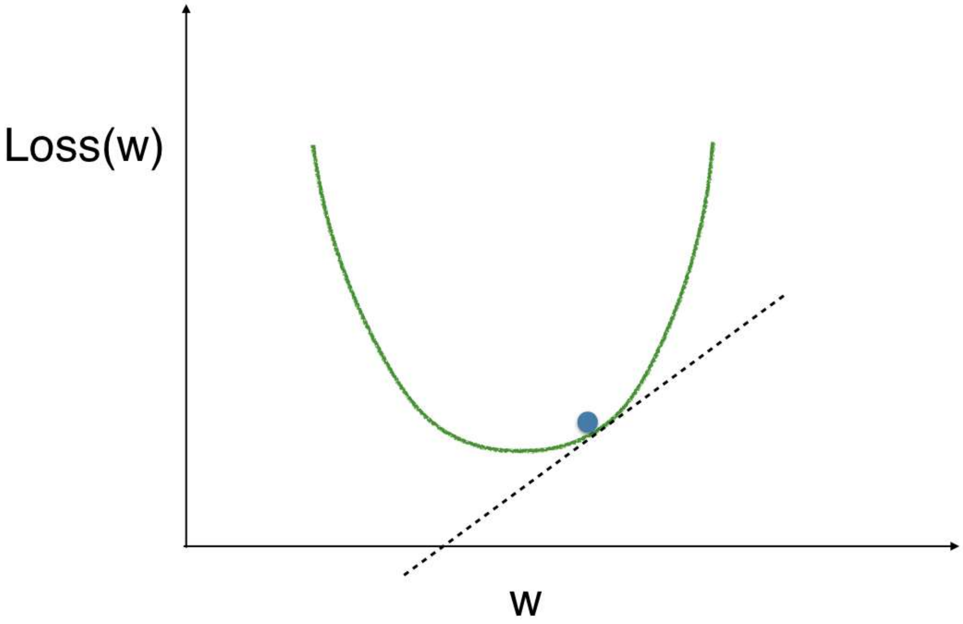
Gradient descent



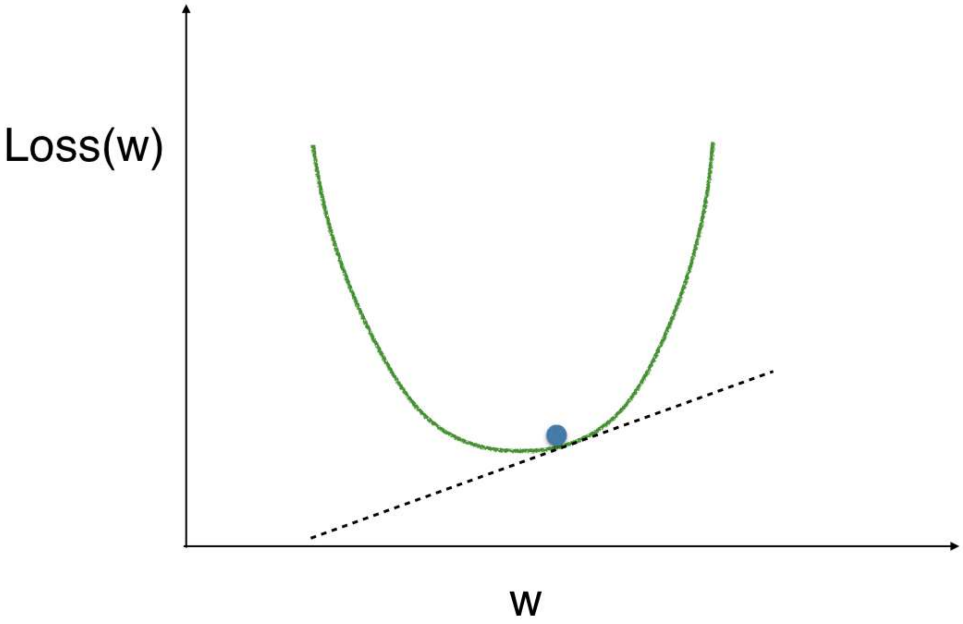
Gradient descent



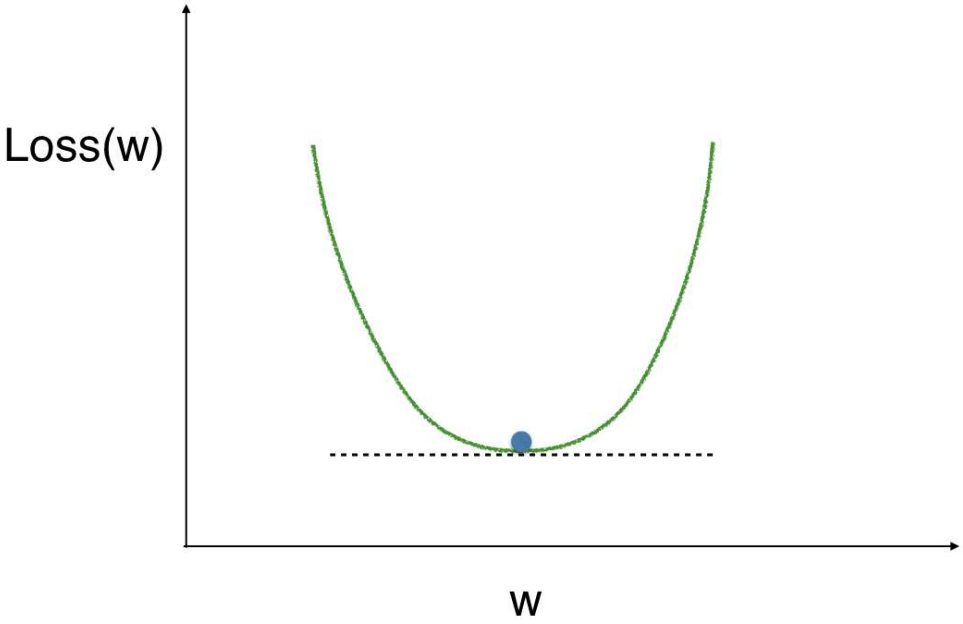
Gradient descent



Gradient descent



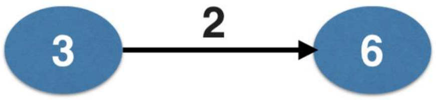
Gradient descent



Gradient descent

- If the slope is positive:
    - Going opposite the slope means moving to lower numbers
    - Subtract the slope from the current value
    - Too big a step might lead us astray
- Solution: learning rate
    - Update each weight by subtracting learning rate * slope

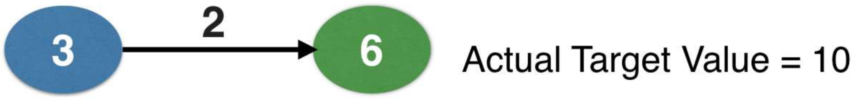
Slope calculation example



Actual Target Value = 10

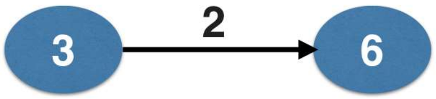
- To calculate the slope for a weight, need to multiply:
  - Slope of the loss function w.r.t value at the node we feed into
  - The value of the node that feeds into our weight
  - Slope of the activation function w.r.t value we feed into

Slope calculation example



- To calculate the slope for a weight, need to multiply:
  - Slope of the loss function w.r.t value at the node we feed into
  - The value of the node that feeds into our weight
  - Slope of the activation function w.r.t value we feed into

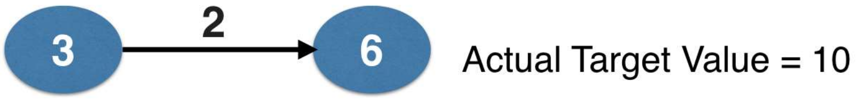
Slope calculation example



Actual Target Value = 10

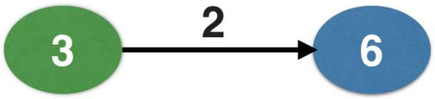
- Slope of mean-squared loss function w.r.t prediction:
    - 2 *(Predicted Value - Actual Value) = 2* Error
    - 2 * -4

Slope calculation example



- To calculate the slope for a weight, need to multiply:
  - Slope of the loss function w.r.t value at the node we feed into
  - **The value of the node that feeds into our weight**
  - Slope of the activation function w.r.t value we feed into

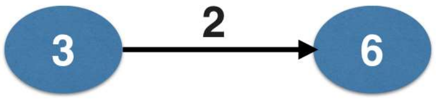
Slope calculation example



Actual Target Value = 10

- To calculate the slope for a weight, need to multiply:
  - Slope of the loss function w.r.t value at the node we feed into
  - The value of the node that feeds into our weight
  - **Slope of the activation function w.r.t value we feed into**

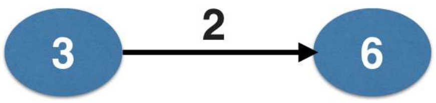
Slope calculation example



Actual Target Value = 10

- To calculate the slope for a weight, need to multiply:
  - Slope of the loss function w.r.t value at the node we feed into
  - The value of the node that feeds into our weight
  - Slope of the activation function w.r.t value we feed into

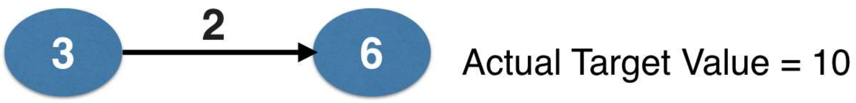
Slope calculation example



Actual Target Value = 10

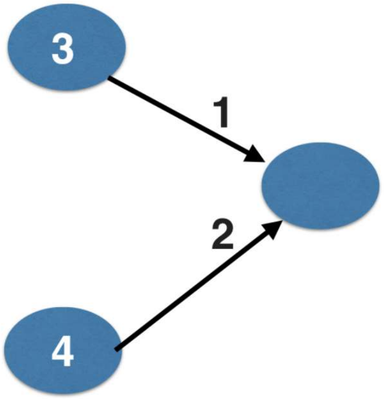
- To calculate the slope for a weight, need to multiply:
    - Slope of the loss function w.r.t value at the node we feed into
    - The value of the node that feeds into our weight
    - ~~Slope of the activation function w.r.t value we feed into~~

Slope calculation example



- `2 * -4 * 3`
- `-24`
- If learning rate is `0.01` , the new weight would be
- `2 - 0.01(-24) = 2.24`

Network with two inputs affecting prediction



Code to calculate slopes and update weights

```python
import numpy as np
weights = np.array([1, 2])
input_data = np.array([3, 4])
target = 6
learning_rate = 0.01
preds = (weights * input_data).sum()
error = preds - target
print(error)
```

```
5
```

Code to calculate slopes and update weights

```
gradient = 2 * input_data * error
gradient
```

```
array([30, 40])
```

```
weights_updated = weights - learning_rate * gradient
preds_updated = (weights_updated * input_data).sum()
error_updated = preds_updated - target
print(error_updated)
```

```
2.5
```