CS21si: AI for Social Good

Lecture 2: Basics of Neural Networks

Intro to Classification

- We want to generate a prediction from some input
- Output should give some probability of all our expected outputs
- Network should give highest probability to correct class most of the time

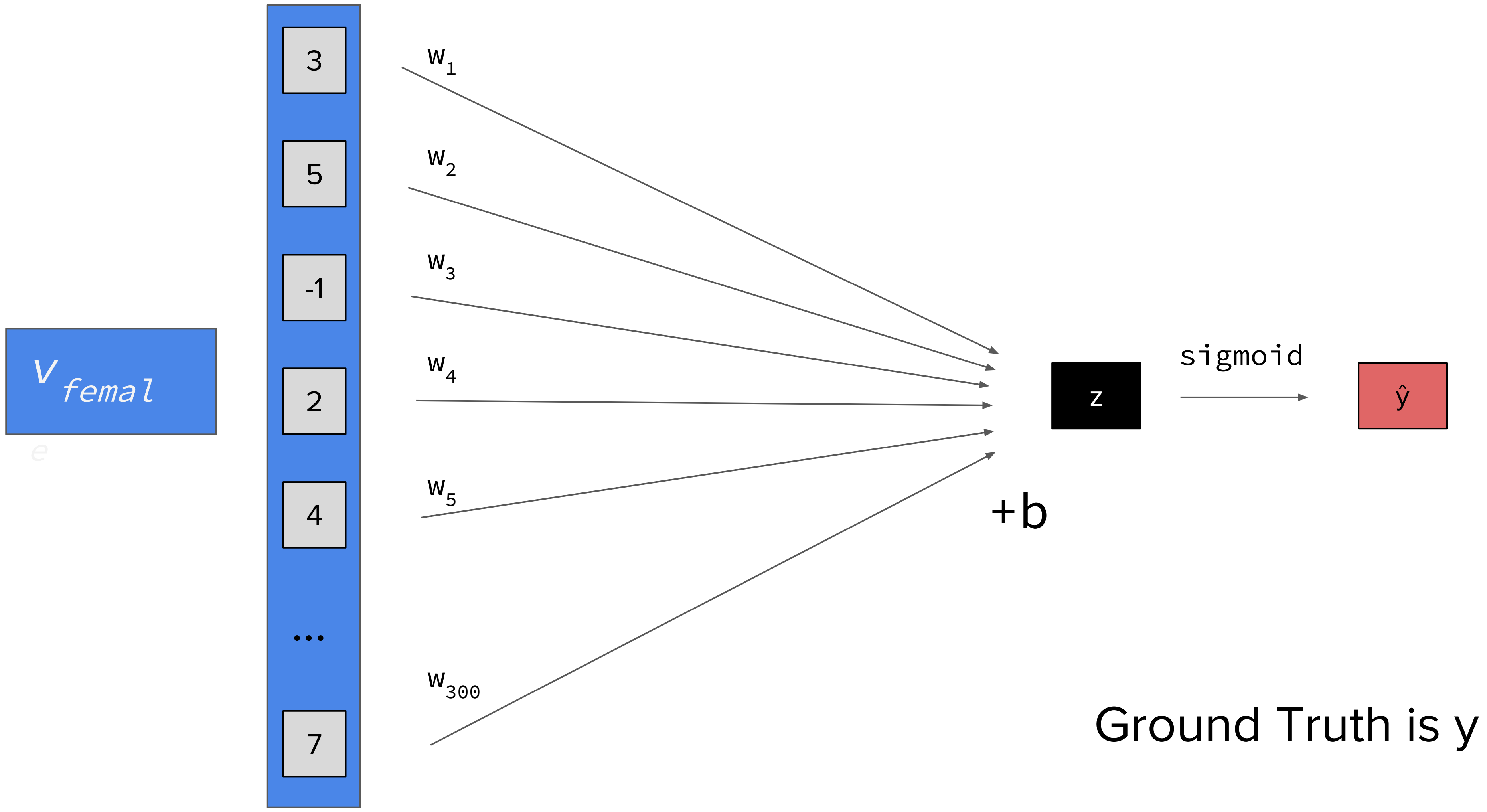
Last time we started **training a model** and ended on the idea of **loss**...

1. How do we calculate loss?

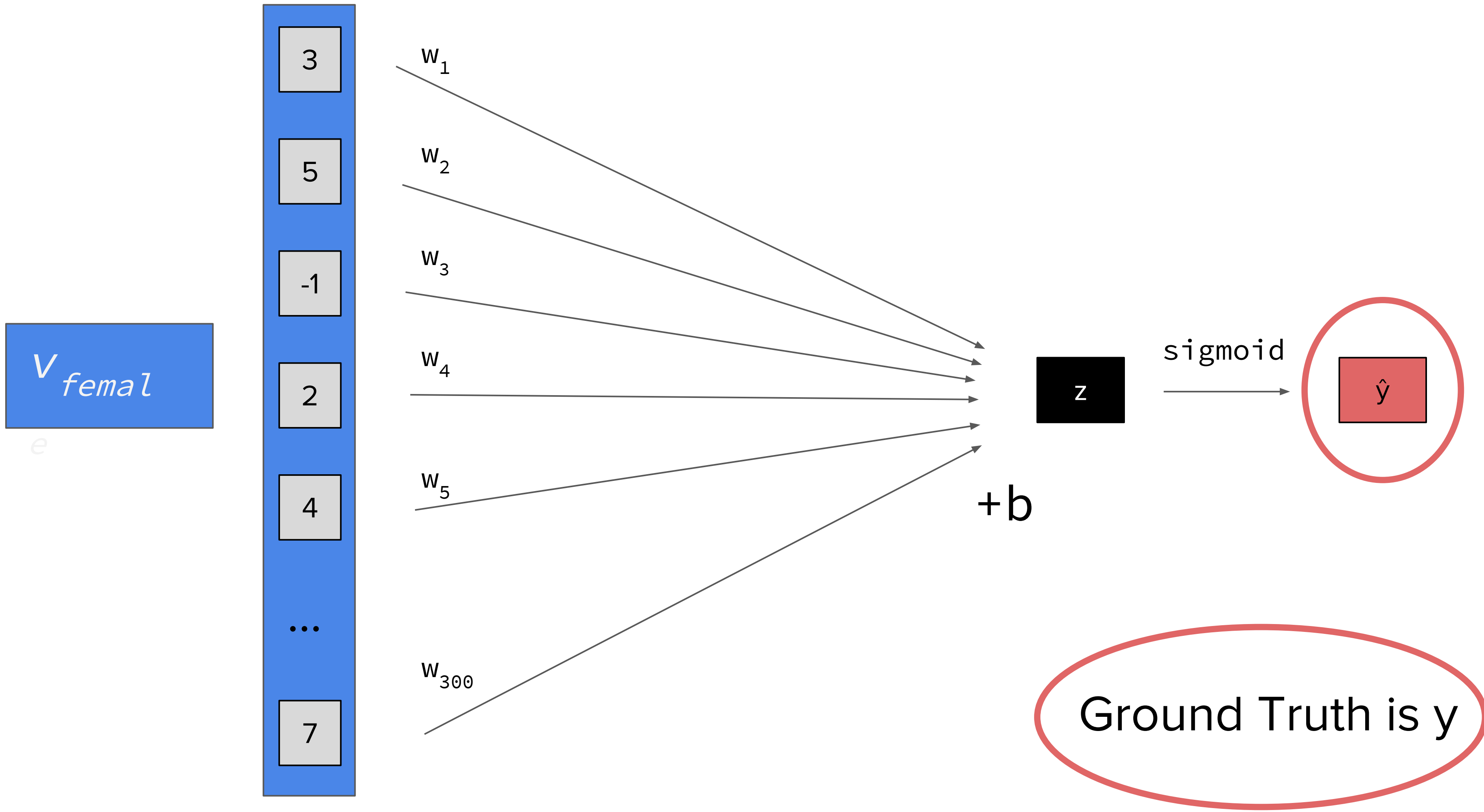
2. How do we use loss to get the right weights, bias?

1. How do we calculate loss?

2. How do we use loss to get the right weights, bias?



V female
3
5
-1
2
4
...
7
w1
w2
w3
w4
w5
w300
+b
z
sigmoid
ŷ
Ground Truth is y



V_{female}
3
5
-1
2
4
...
7
w_1
w_2
w_3
w_4
w_5
w_{300}
z
+b
sigmoid
ŷ
Ground Truth is y

$$Loss(y, \hat{y})$$

Ground truth

Predicted label

Idea: maximize this!

$$\hat{y}^{y} * (1 - \hat{y})^{1-y}$$

$$y * \log \hat{y} + (1 - y) * \log (1 - \hat{y})$$

$$Loss = -y * \log \hat{y} - (1 - y) * \log (1 - \hat{y})$$

$$Cost = \sum_{i=1}^{m} Loss(y_i, \hat{y}_i)$$

Questions?

Jupyter Notebook Exercises: Part 1

You'll need:

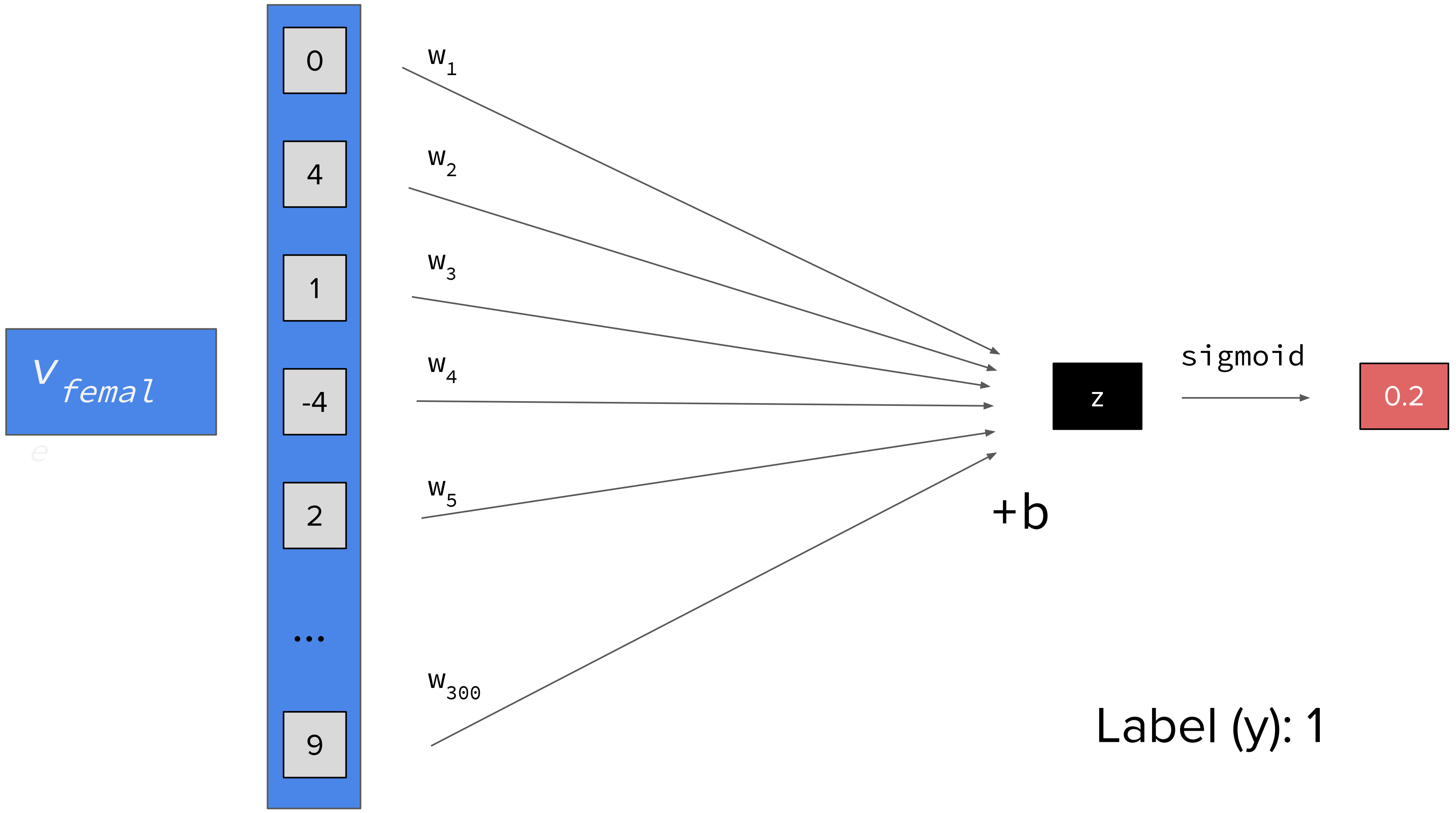
```
np.log(x)
```

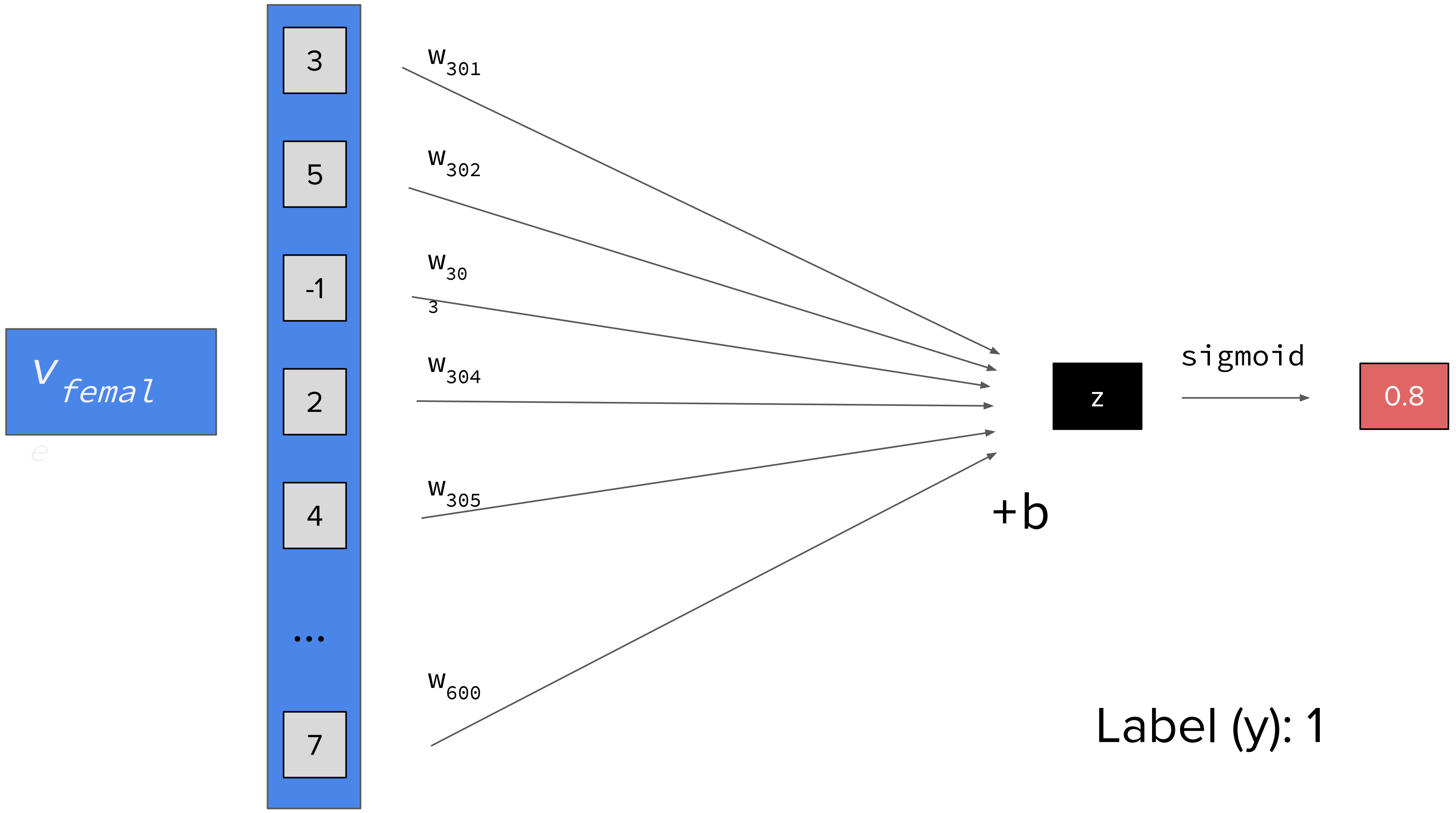
$$Loss = -y * \log \hat{y} - (1 - y) * \log (1 - \hat{y})$$

1. How do we calculate loss?

2. How do we use loss to get the right weights, bias?

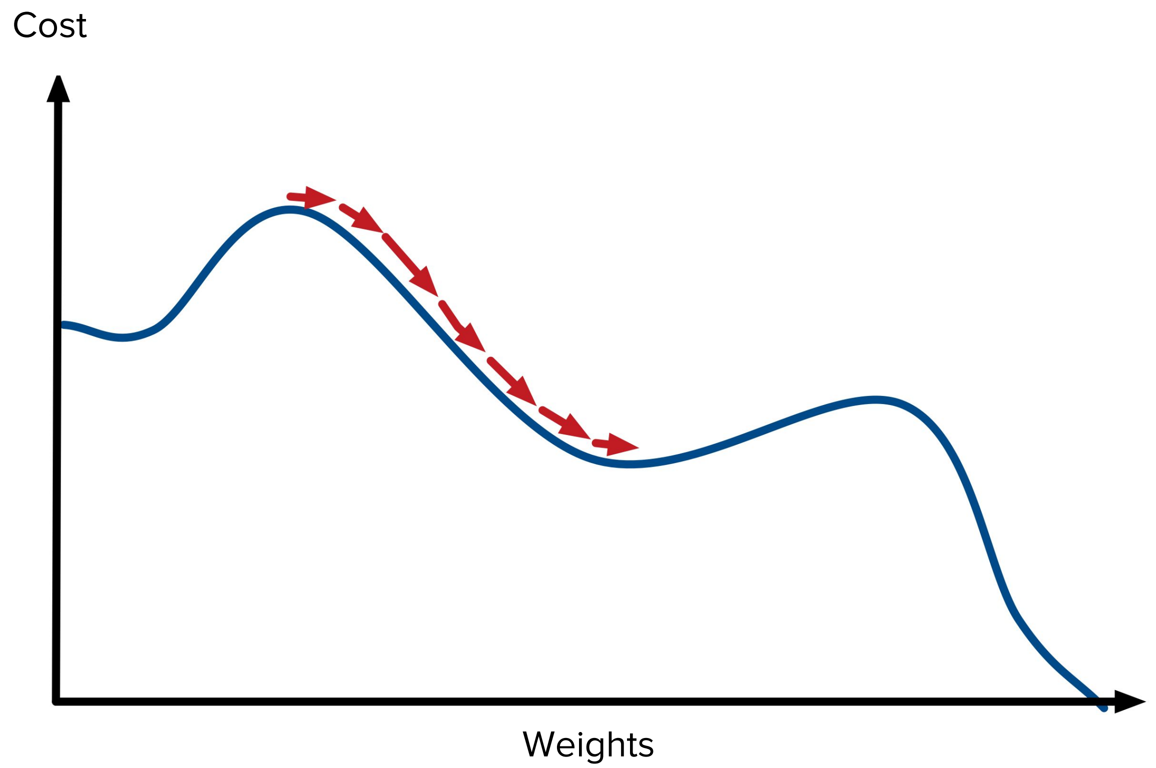
Can we randomly guess?


V_{female}
0
4
1
-4
2
...
9
w_1
w_2
w_3
w_4
w_5
w_{300}
+b
z
sigmoid
0.2
Label (y): 1



V_{female}
3
5
-1
2
4
...
7
w_{301}
w_{302}
w_{303}
w_{304}
w_{305}
w_{600}
z
+b
sigmoid
0.8
Label (y): 1

Gradient Descent



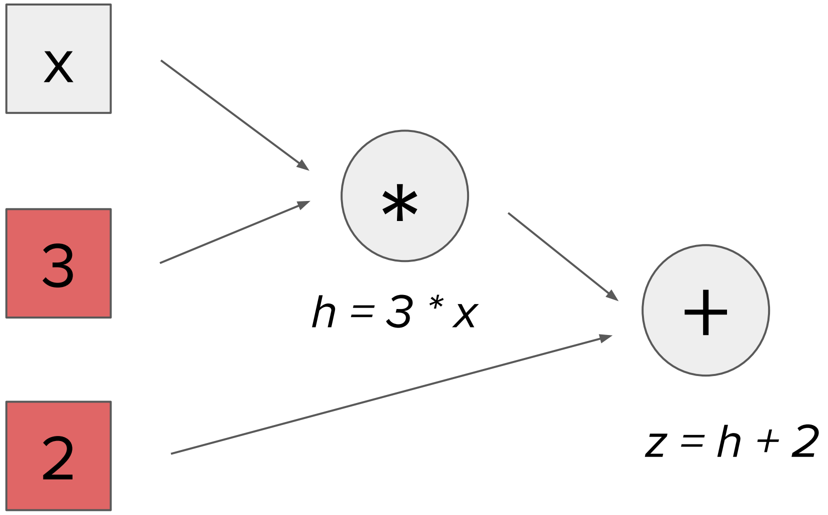
How do we calculate the gradients?

Backpropagation

- Each computation is represented as a node in a computational graph
- Gradients are computed at each node of the graph
- Chain rule is applied recursively to get total gradients on each weight

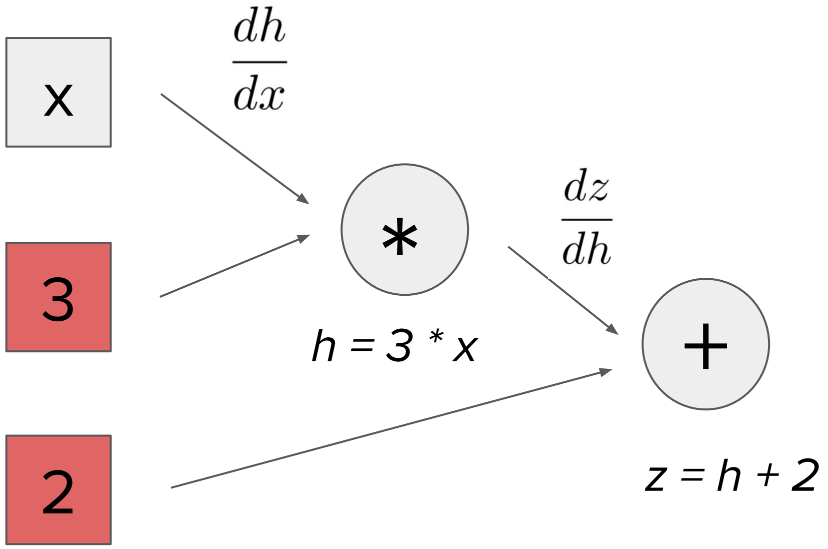
Backpropagation

3x + 2



Backpropagation

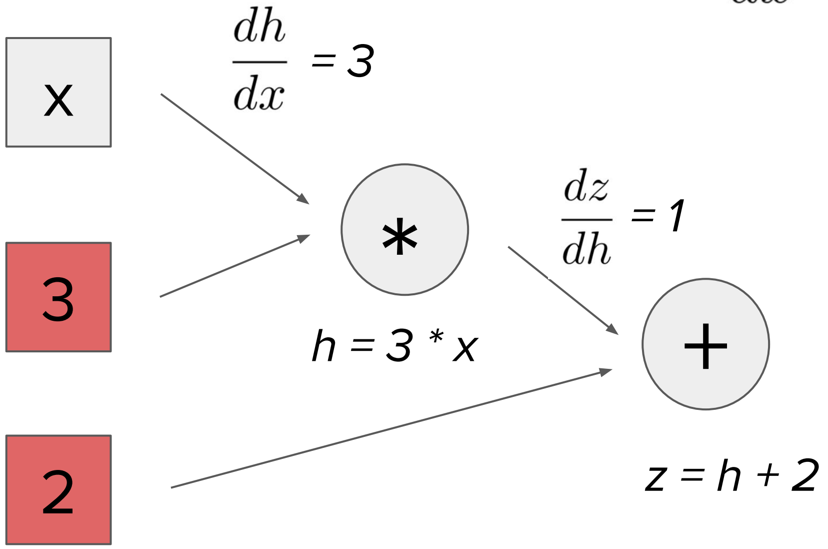
3x + 2



Backpropagation

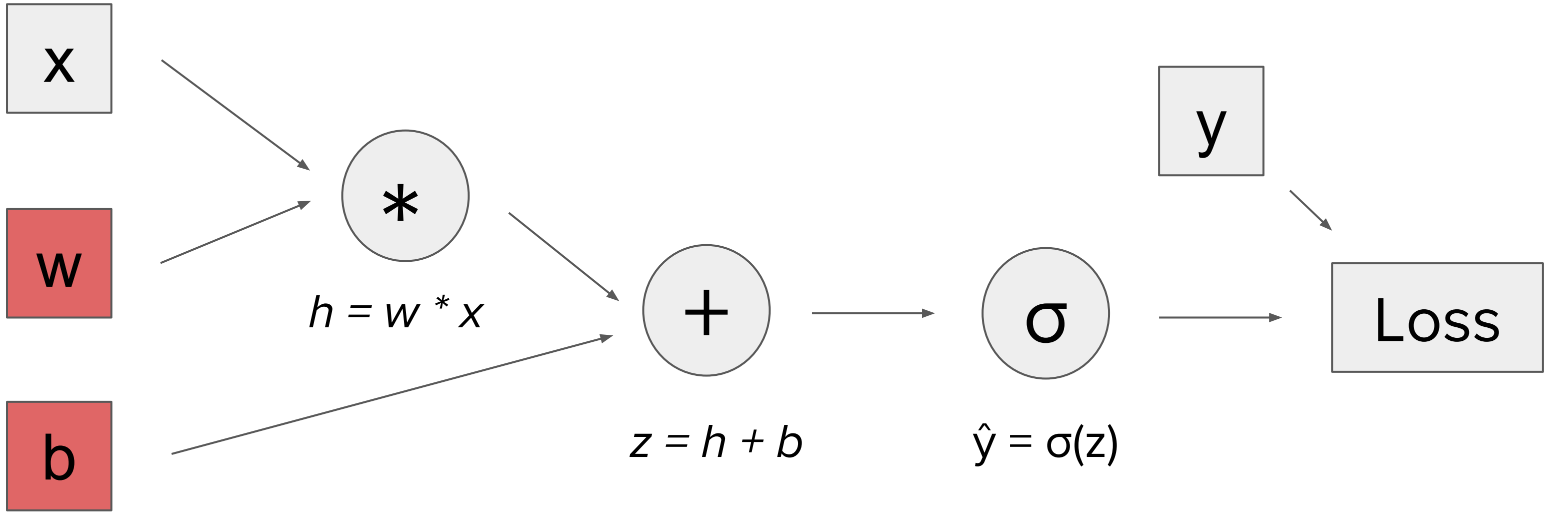
3x + 2

$$\frac{dz}{dx} = \frac{dz}{dh}\frac{dh}{dx} = 3$$

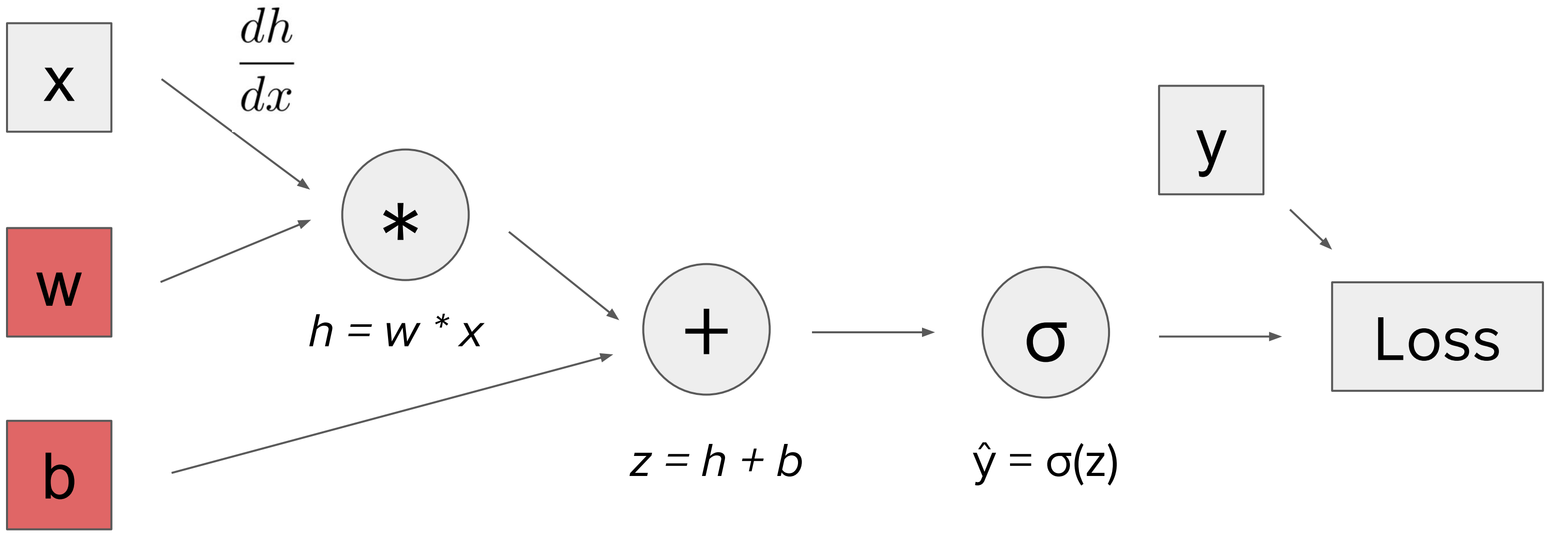


Questions?

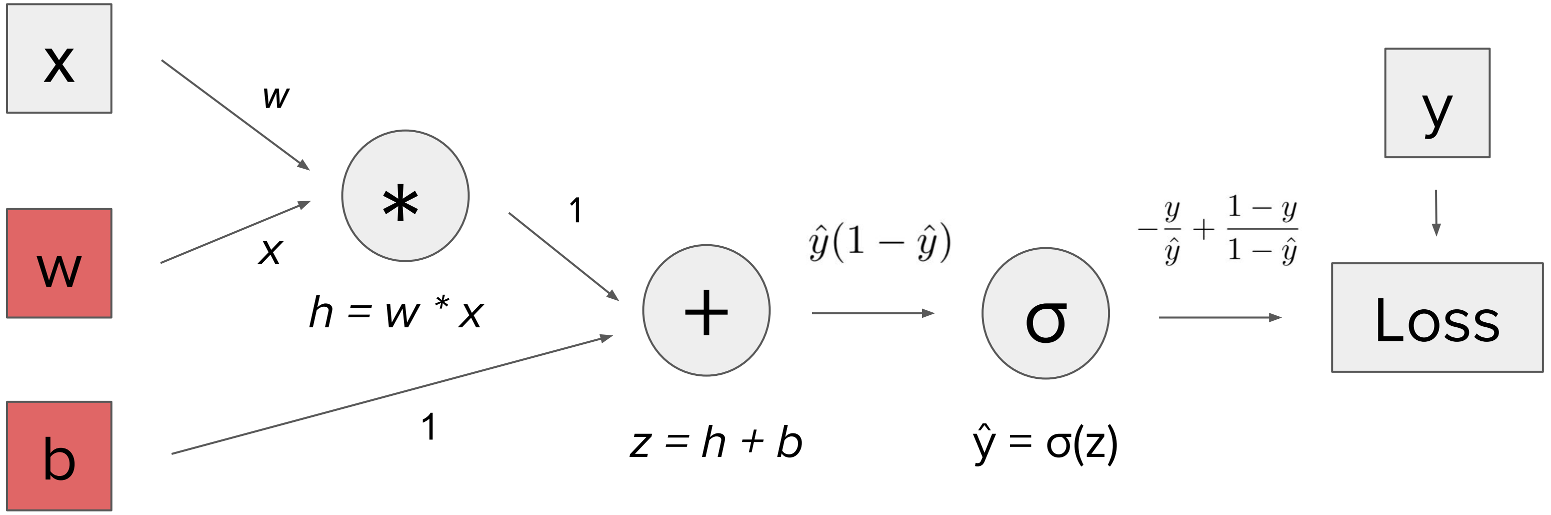
Backpropagation



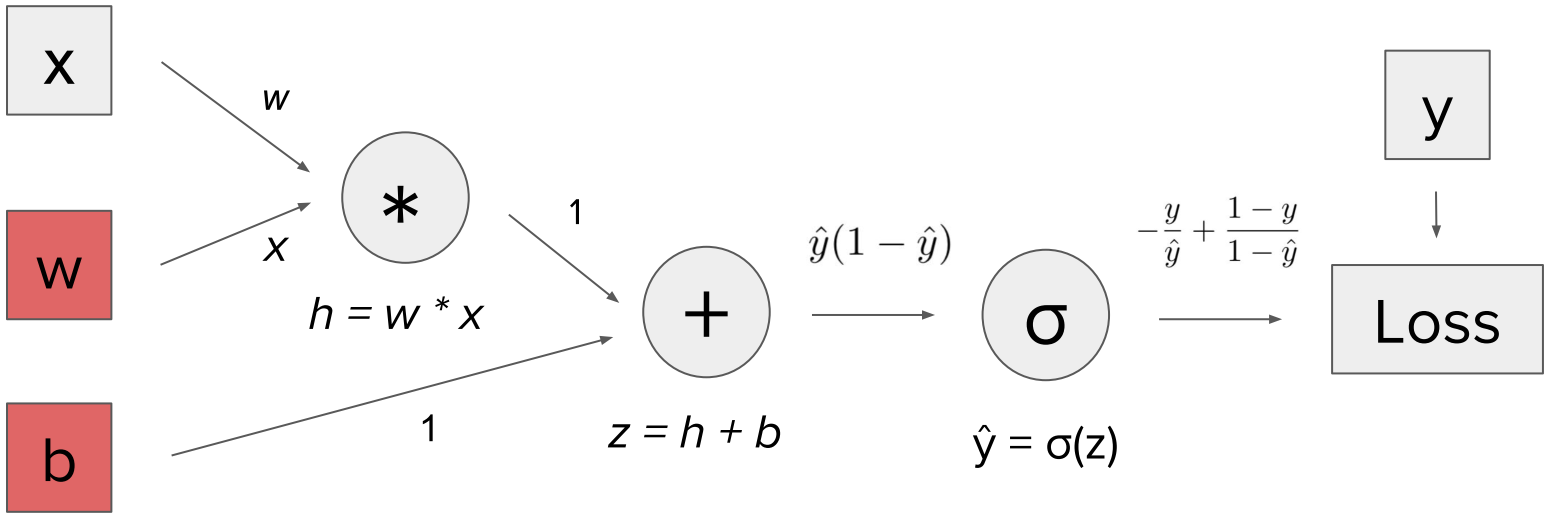
Backpropagation



Backpropagation



Backpropagation



$$\frac{\partial Loss}{\partial w} = (\hat{y} - y) * x \qquad \frac{\partial Loss}{\partial b} = \hat{y} - y$$

Jupyter Notebook Exercises: Part 2

You'll need:

$$\frac{\partial Loss}{\partial w} = (\hat{y} - y) * x \qquad \frac{\partial Loss}{\partial b} = \hat{y} - y$$

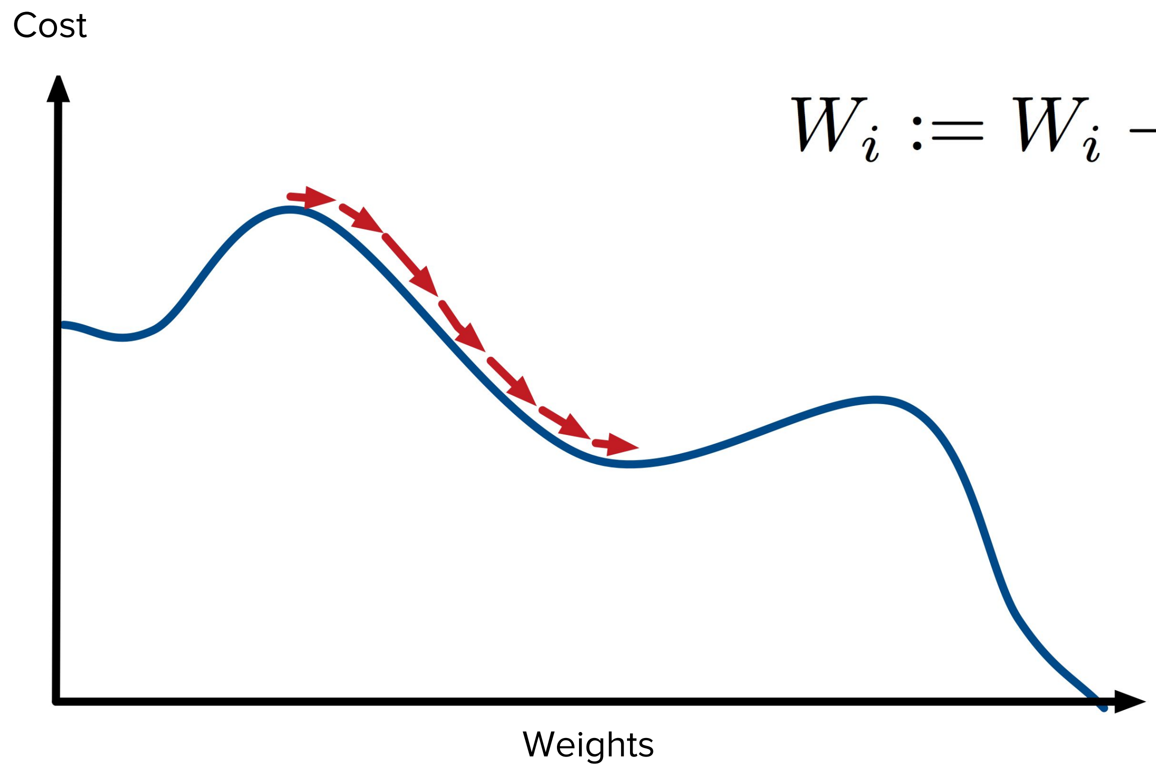
Training

Weight Update

- Once we find gradients, we must change each weight in the opposite direction
- We can use an update rule to accomplish this; the simplest is SGD:

$$W_i := W_i - \alpha \frac{\partial L}{\partial W_i}$$

Stochastic Gradient Descent



$$W_i := W_i - \alpha \frac{\partial L}{\partial W_i}$$

Jupyter Notebook Exercises: Part 3

You'll need:

```
np.random.randn(size)
```

Neural networks

- Logistic regression is a special case of a neural network
- By adding more “layers” of computation to a logistic regression model, you can turn it into a “deep” NN

What we’ve learned...

- You can train logistic regression using loss computation against ground truth to measure how badly you are doing
- How to minimize loss using SGD in a neural network
- By adding more “layers” of computation to logistic regression, you get a deep neural network