ML Lab Supplement

Ruvan Weerasinghe

(With thanks for material heavily borrowed from Fahim Dalvi of QCRI)

Overview

- Backpropagation
- Bias
- Parameter Initialization
- Regularization

Backpropagation

Backpropagation is a technique to compute gradients of any function with respect to a variable using the concept of a *computation graph*

Backpropagation

Computation graph: Graphical way of describing any function:

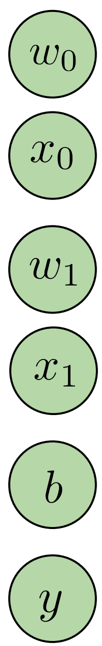
$$\mathcal{L} = (f(x, W, b) - y)^2$$

Backpropagation

$f(x, W, b) = w_0 \cdot x_0 + w_1 \cdot x_1 + b$

Computation graph: Graphical way of describing any function:

$$\mathcal{L} = (f(x, W, b) - y)^2$$



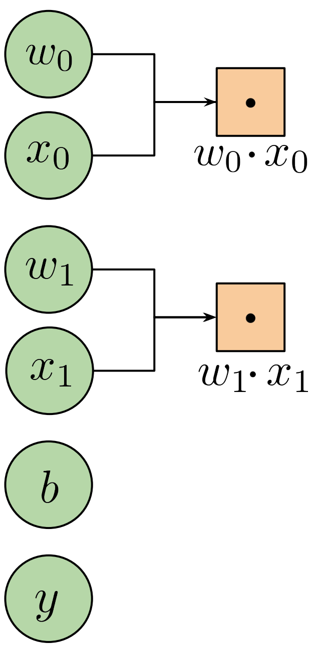
Each node in the graph is either an input, an operation or an output

Backpropagation

$f(x, W, b) = w_0 \cdot x_0 + w_1 \cdot x_1 + b$

Computation graph: Graphical way of describing any function:

$$\mathcal{L} = (f(x, W, b) - y)^2$$



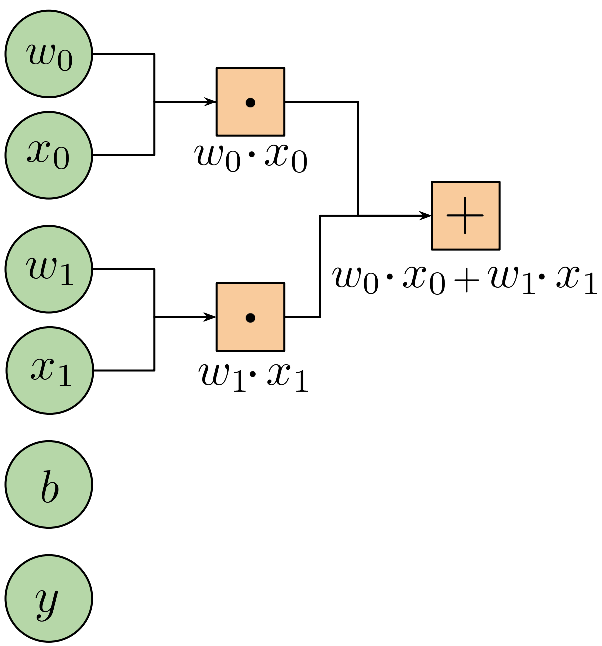
Each node in the graph is either an input, an operation or an output

Backpropagation

$f(x, W, b) = w_0 \cdot x_0 + w_1 \cdot x_1 + b$

Computation graph: Graphical way of describing any function:

$$\mathcal{L} = (f(x, W, b) - y)^2$$

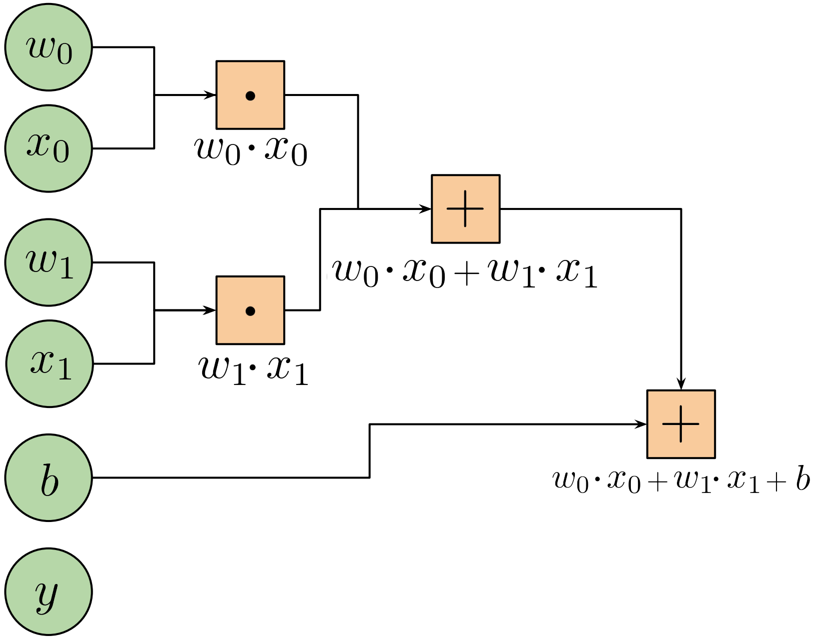


Each node in the graph is either an input, an operation or an output

Backpropagation

Computation graph: Graphical way of describing any function:

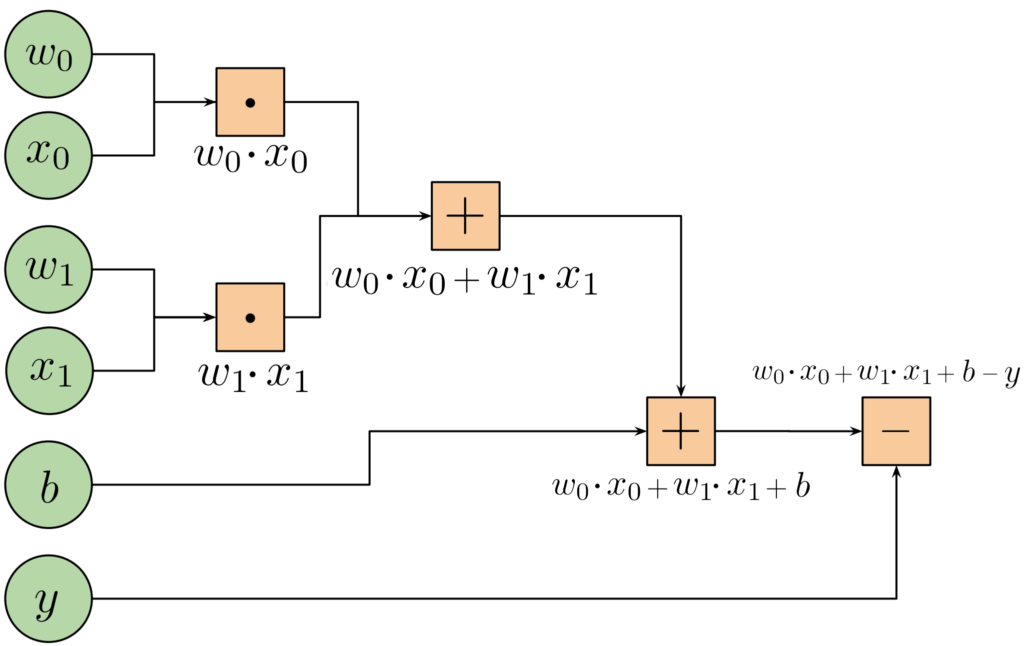
$$\mathcal{L} = (f(x, W, b) - y)^2$$



Backpropagation

Computation graph: Graphical way of describing any function:

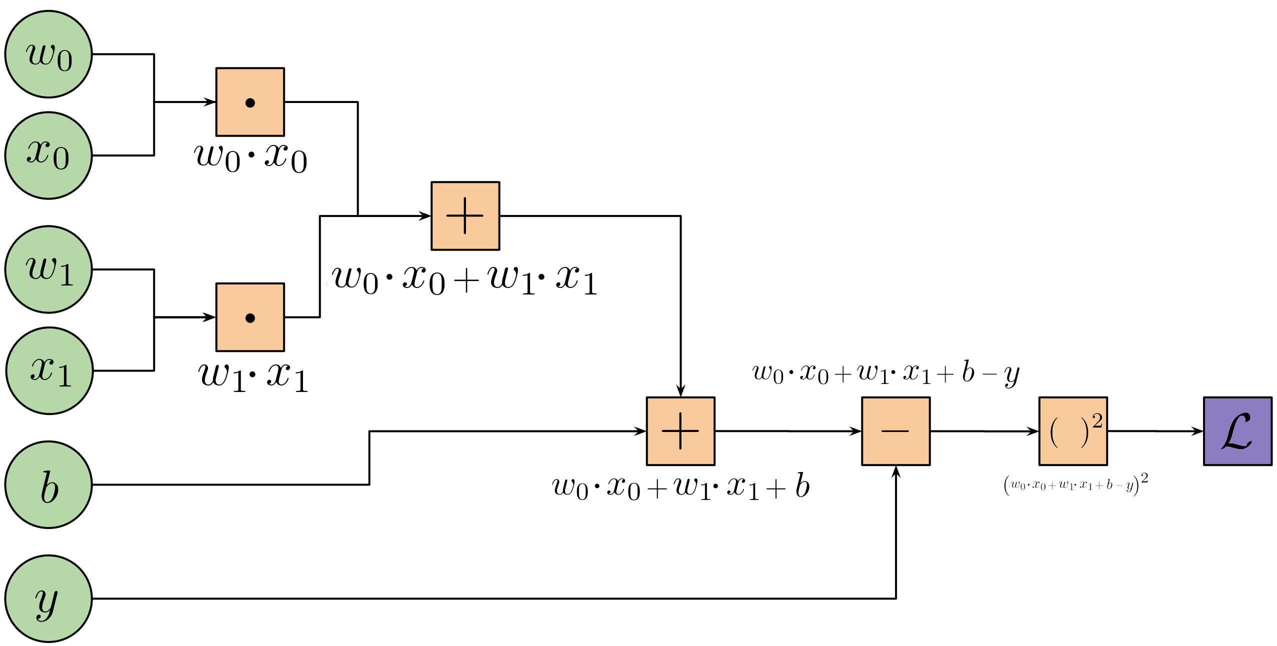
$$\mathcal{L} = (f(x, W, b) - y)^2$$



Backpropagation

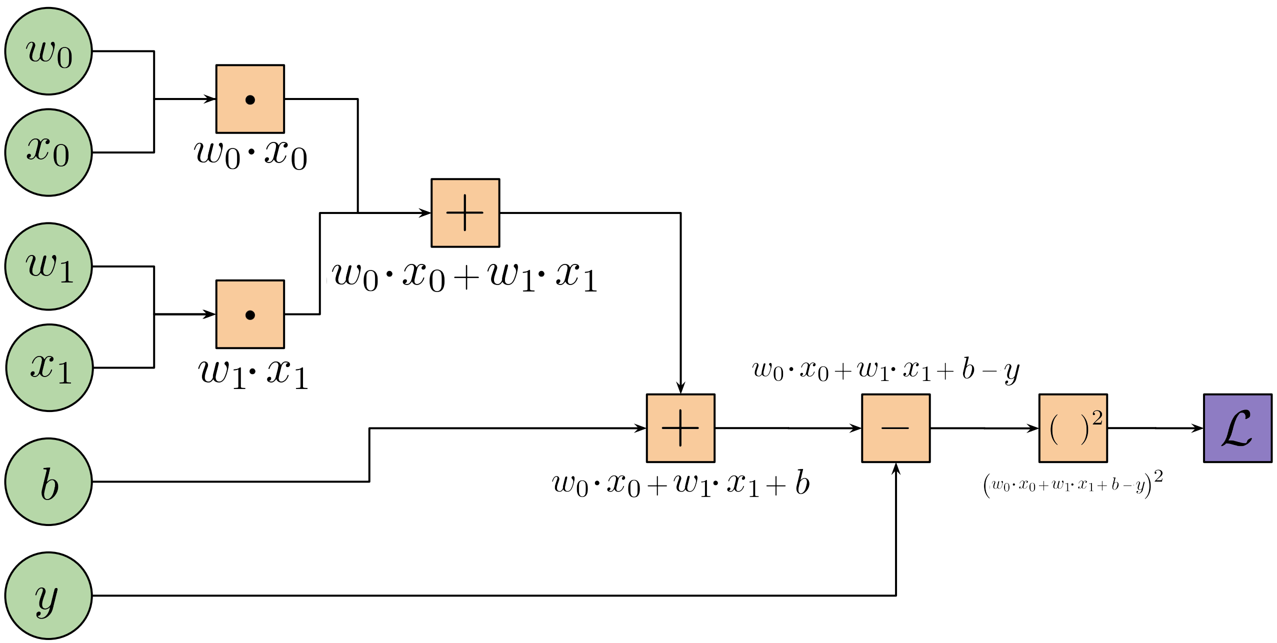
Computation graph: Graphical way of describing any function:

$$\mathcal{L} = (f(x, W, b) - y)^2$$



Backpropagation

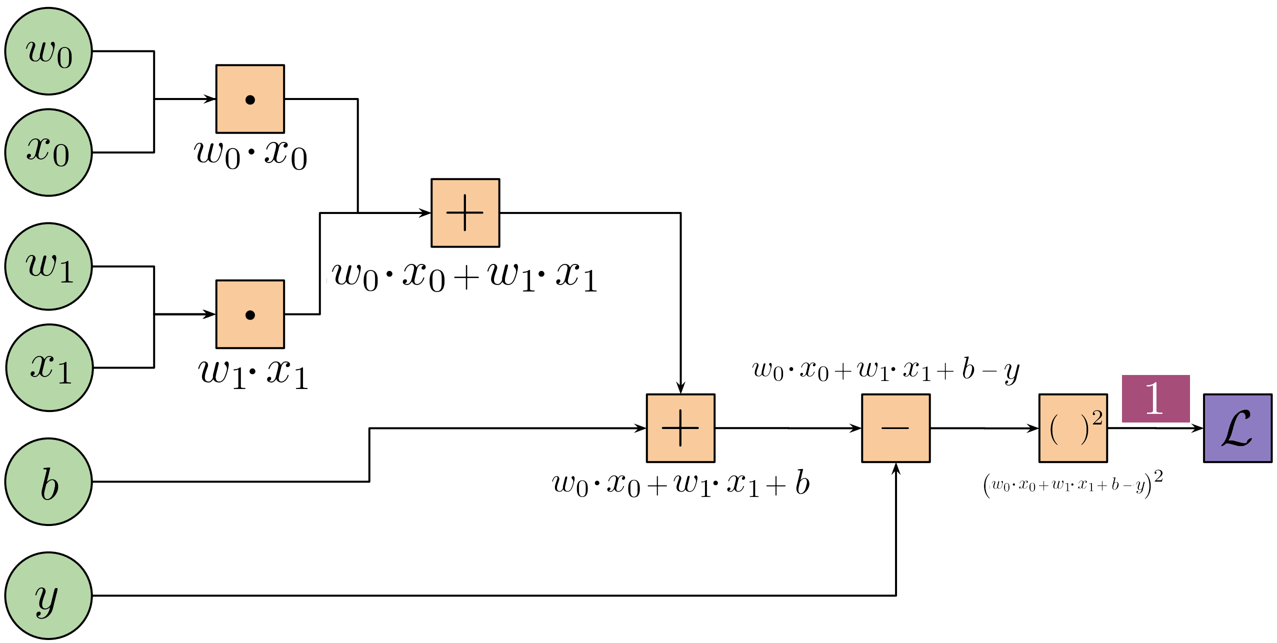
$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

Backpropagation

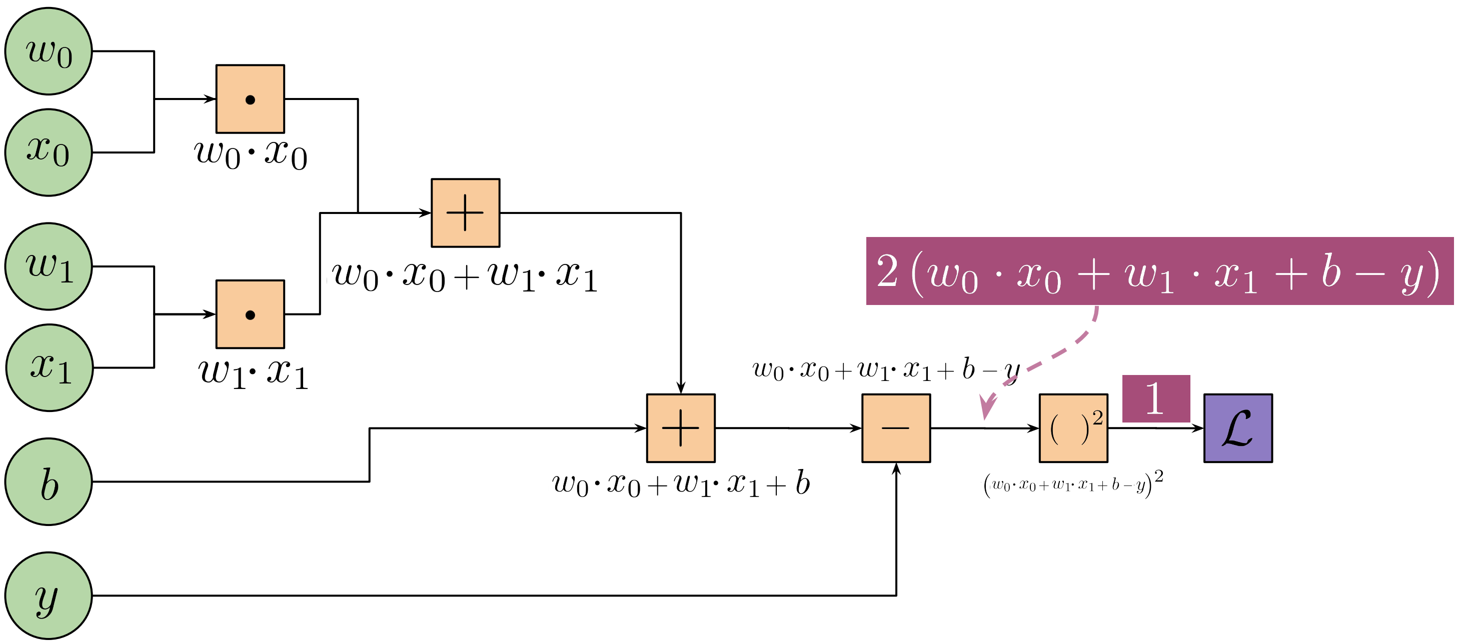
$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

Backpropagation

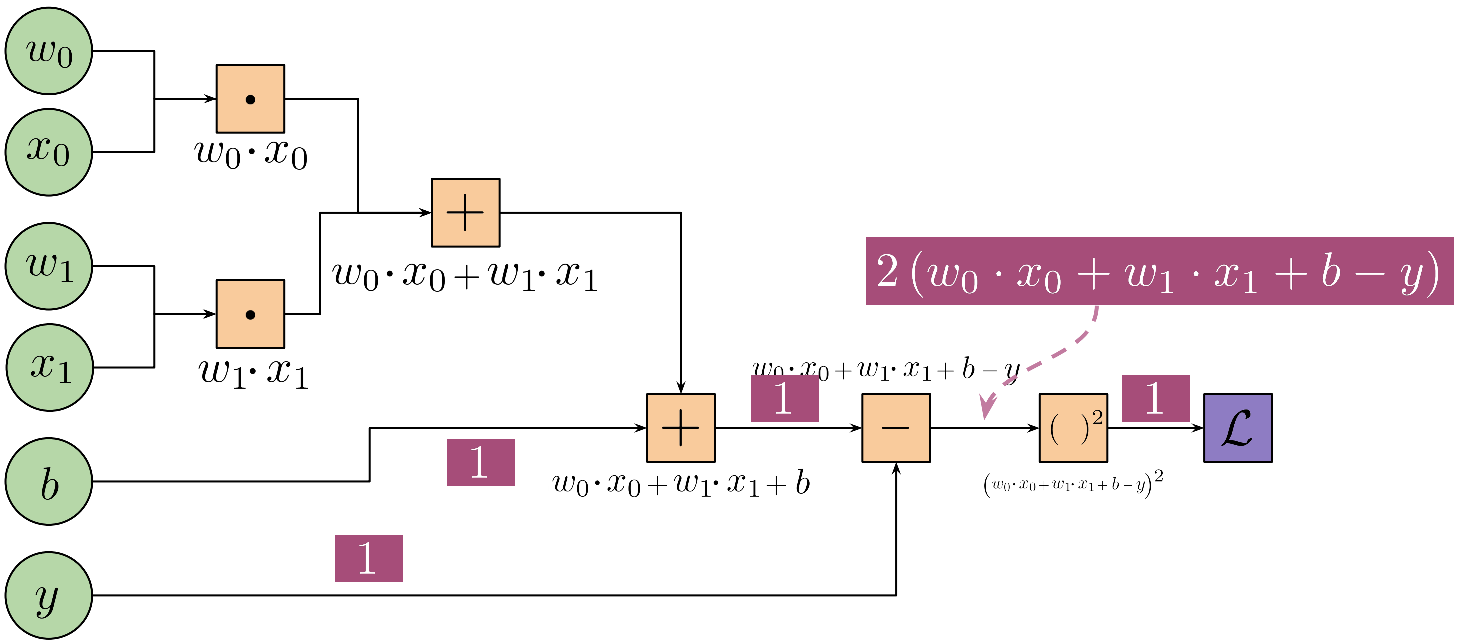
$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

Backpropagation

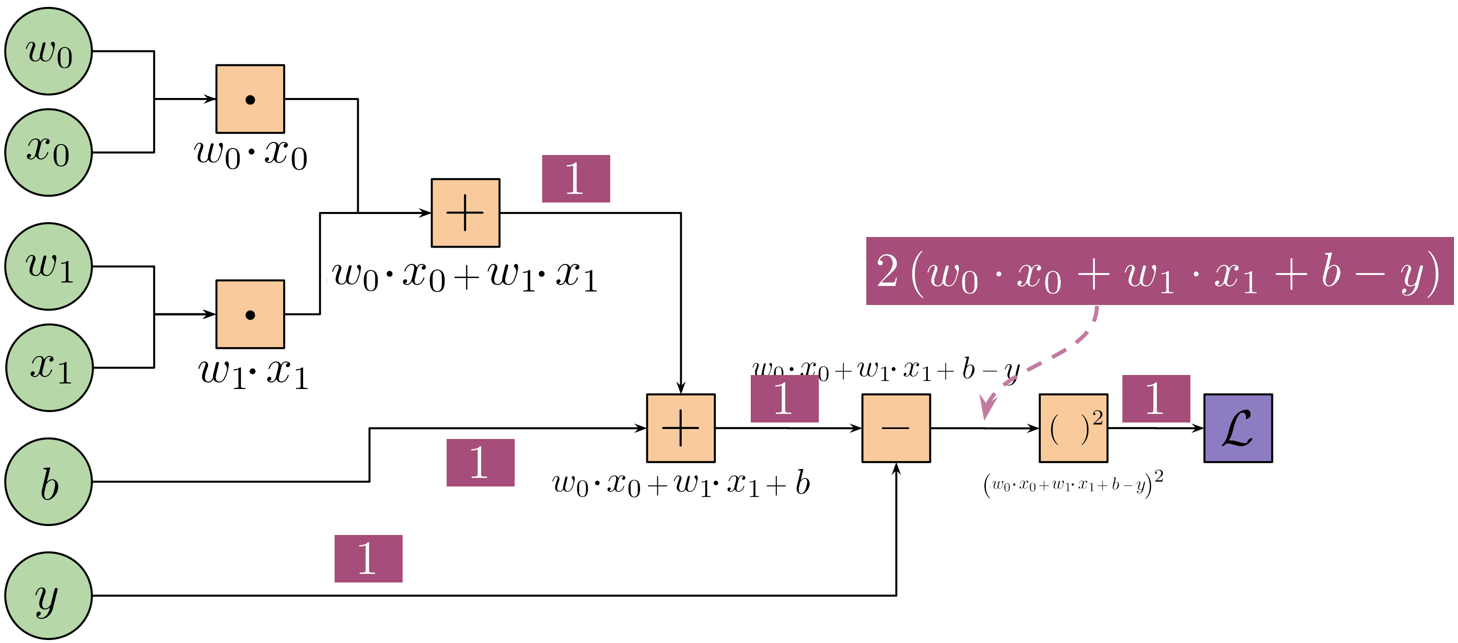
$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

Backpropagation

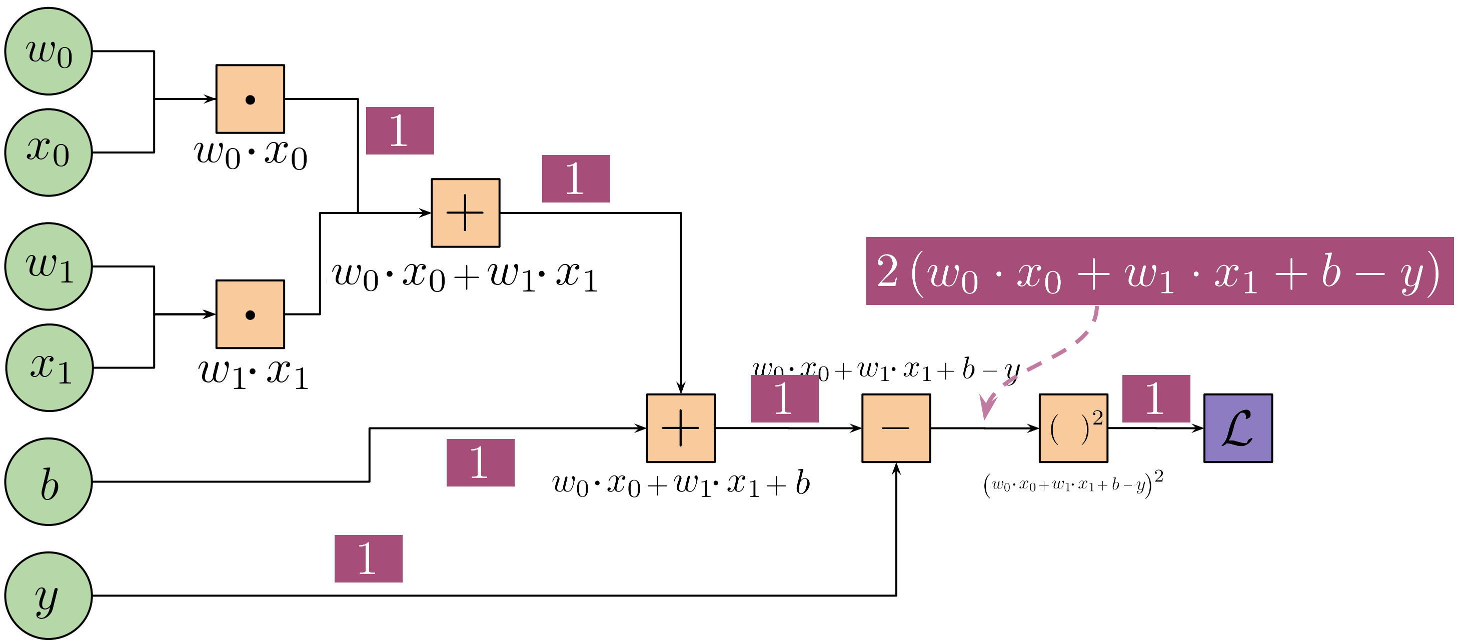
$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

Backpropagation

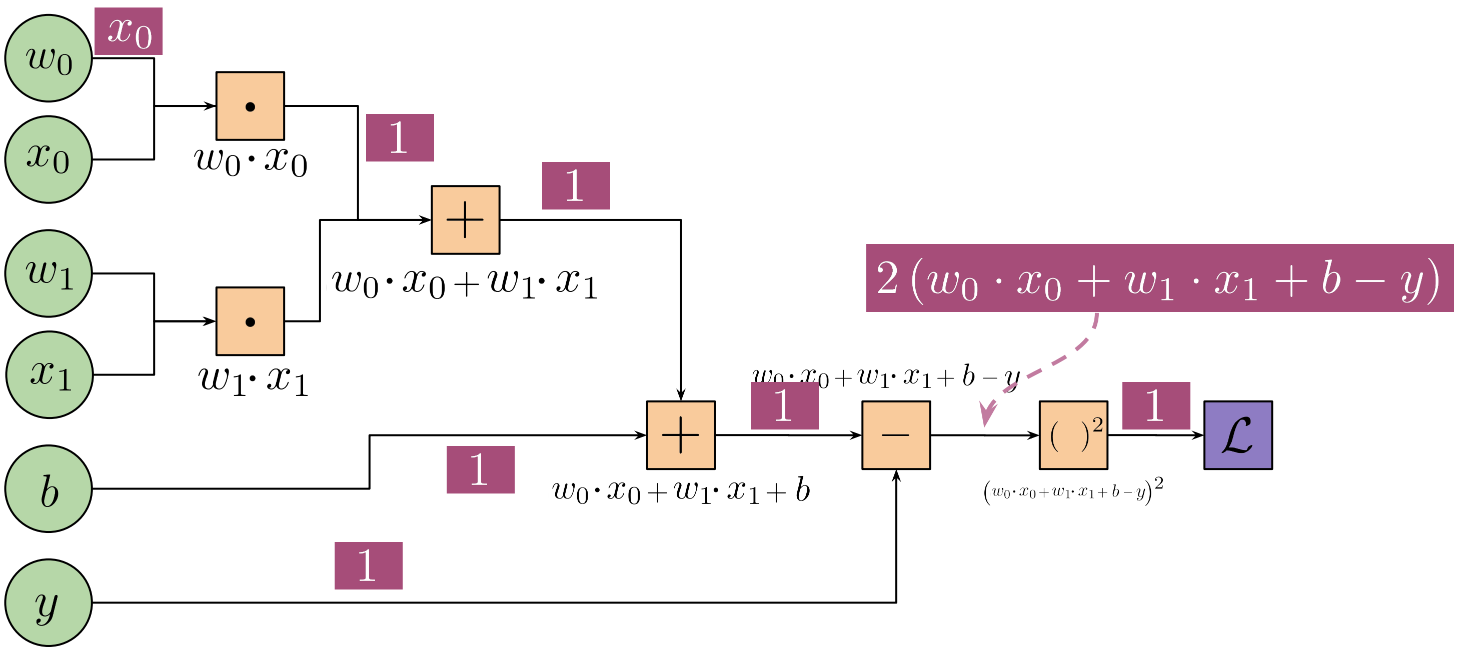
$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

Backpropagation

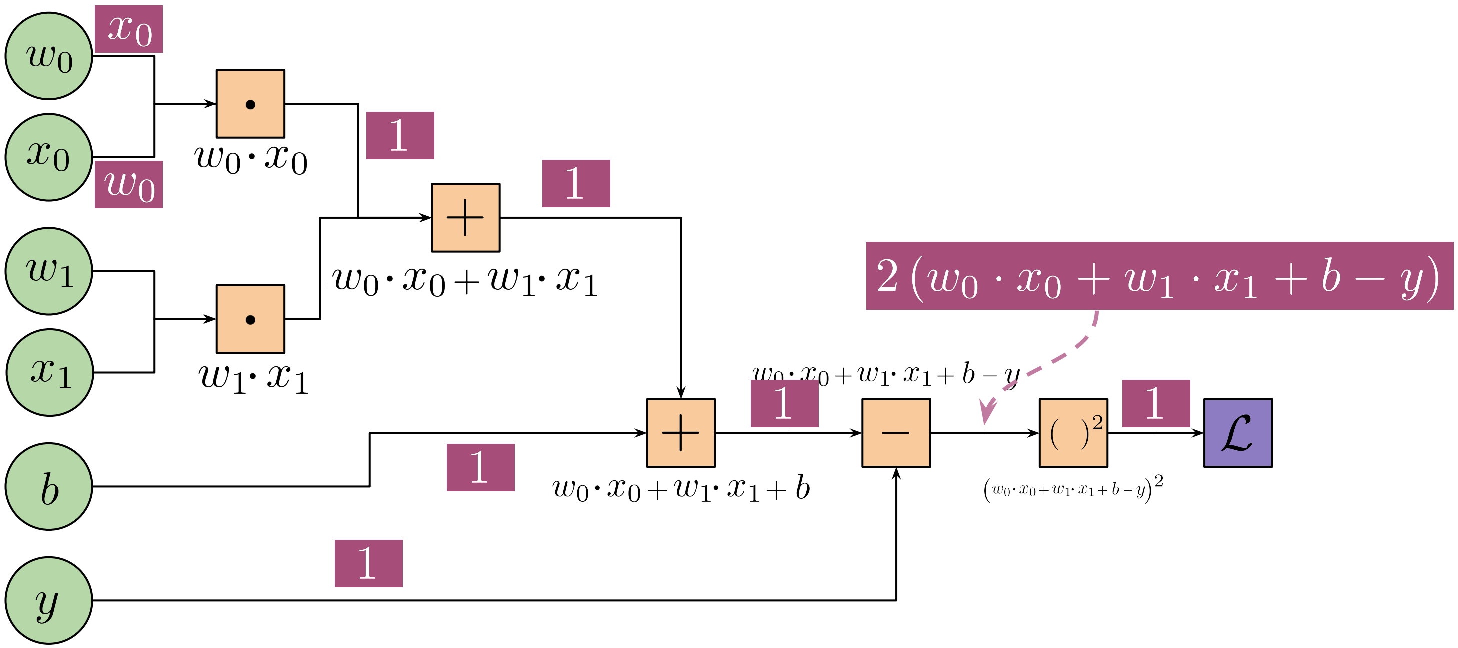
$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

Backpropagation

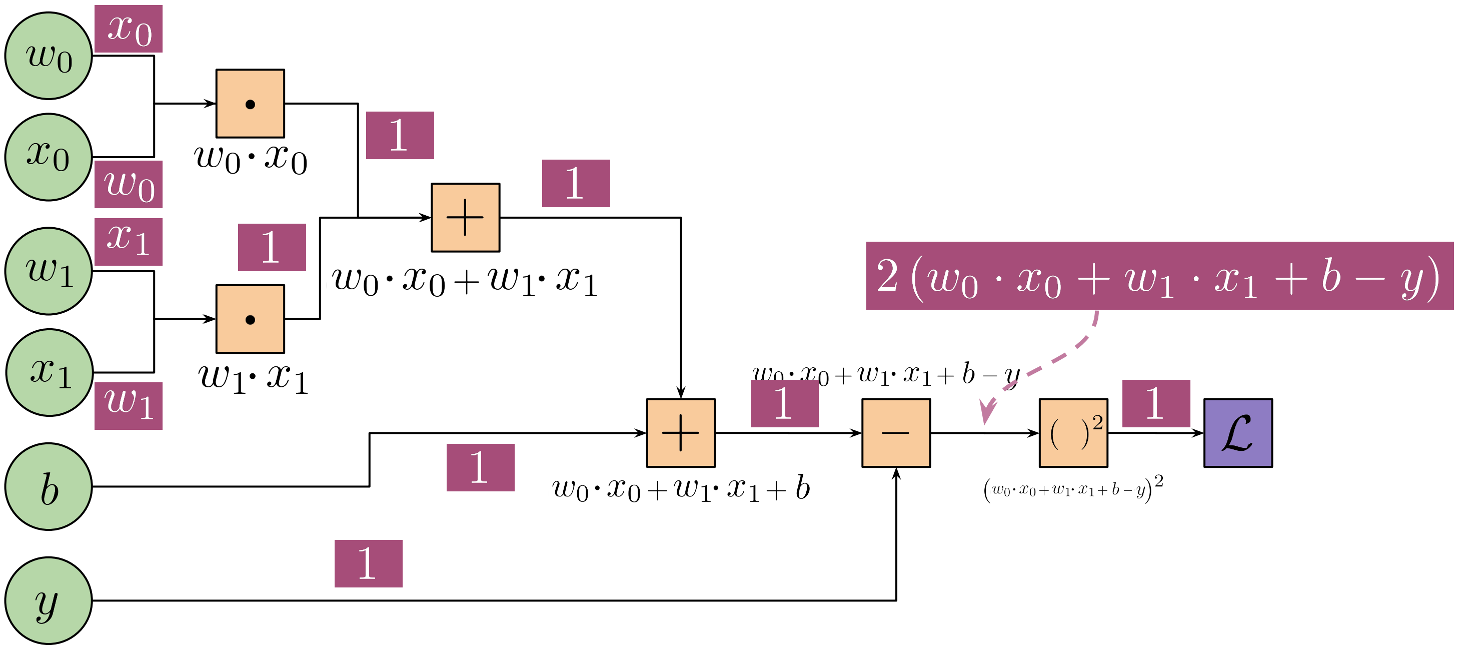
$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

Backpropagation

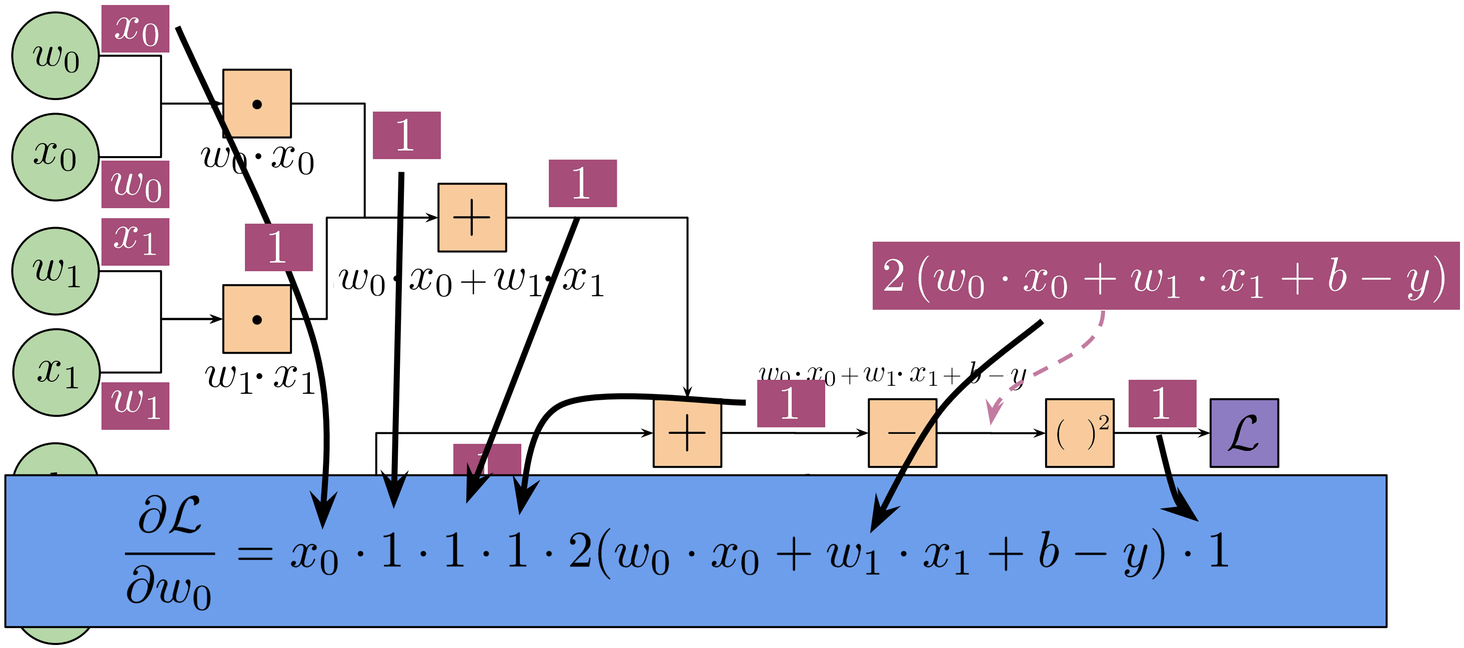
$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

Backpropagation

$$\mathcal{L} = (f(x, W, b) - y)^2$$



We can now use the chain rule to compute the gradient, and work our way backwards from the output to the inputs!

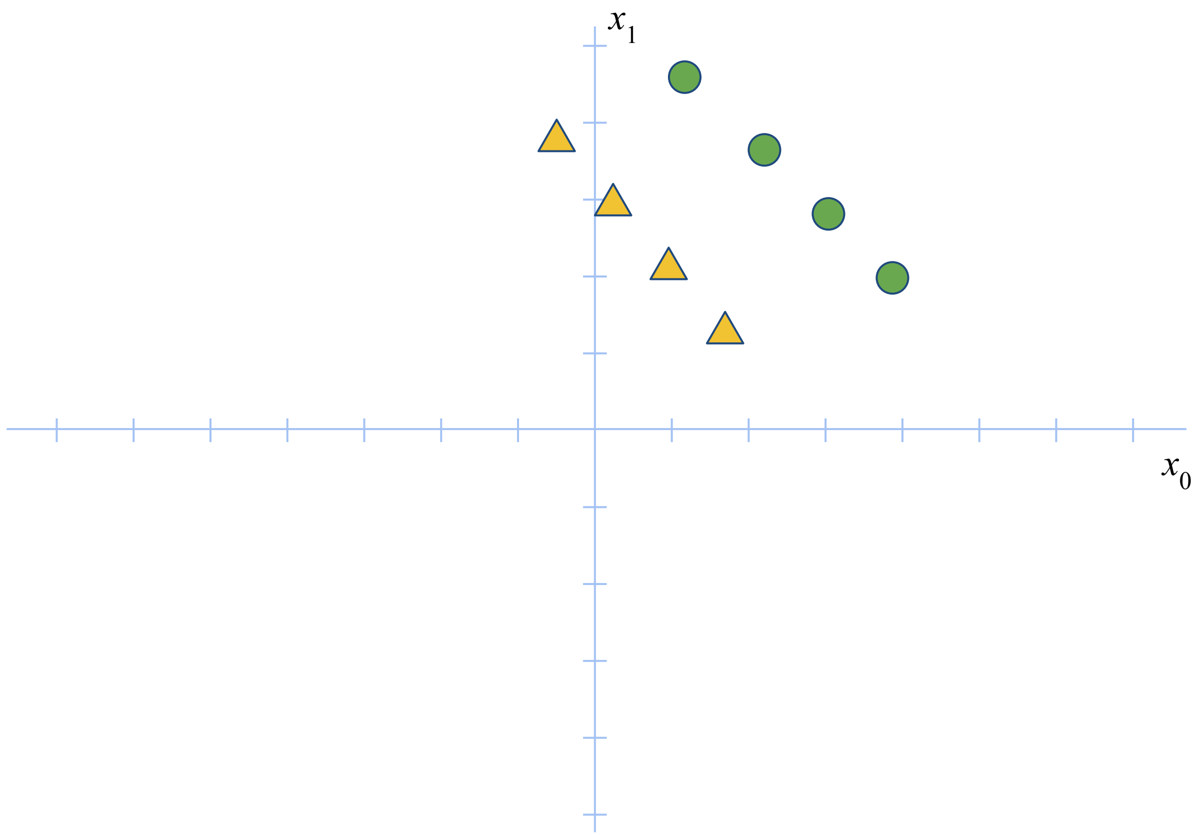
Bias

Bias

- What was the “b”?
  - A parameter that allows you to “shift” your decision boundary
  - In the case of a linear boundary (`f = Wx + b`), the W can only control the slope of the boundary - but that may not be enough

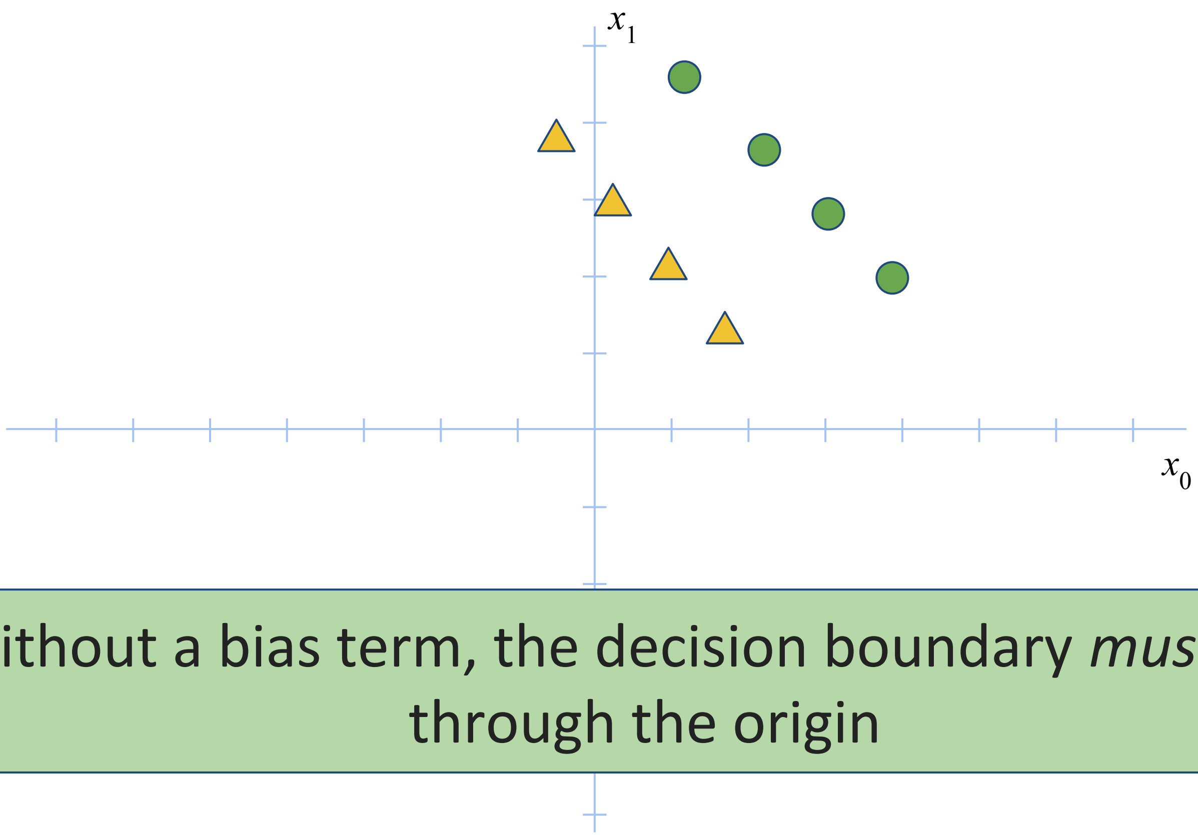
Bias

Consider the following dataset:



Bias

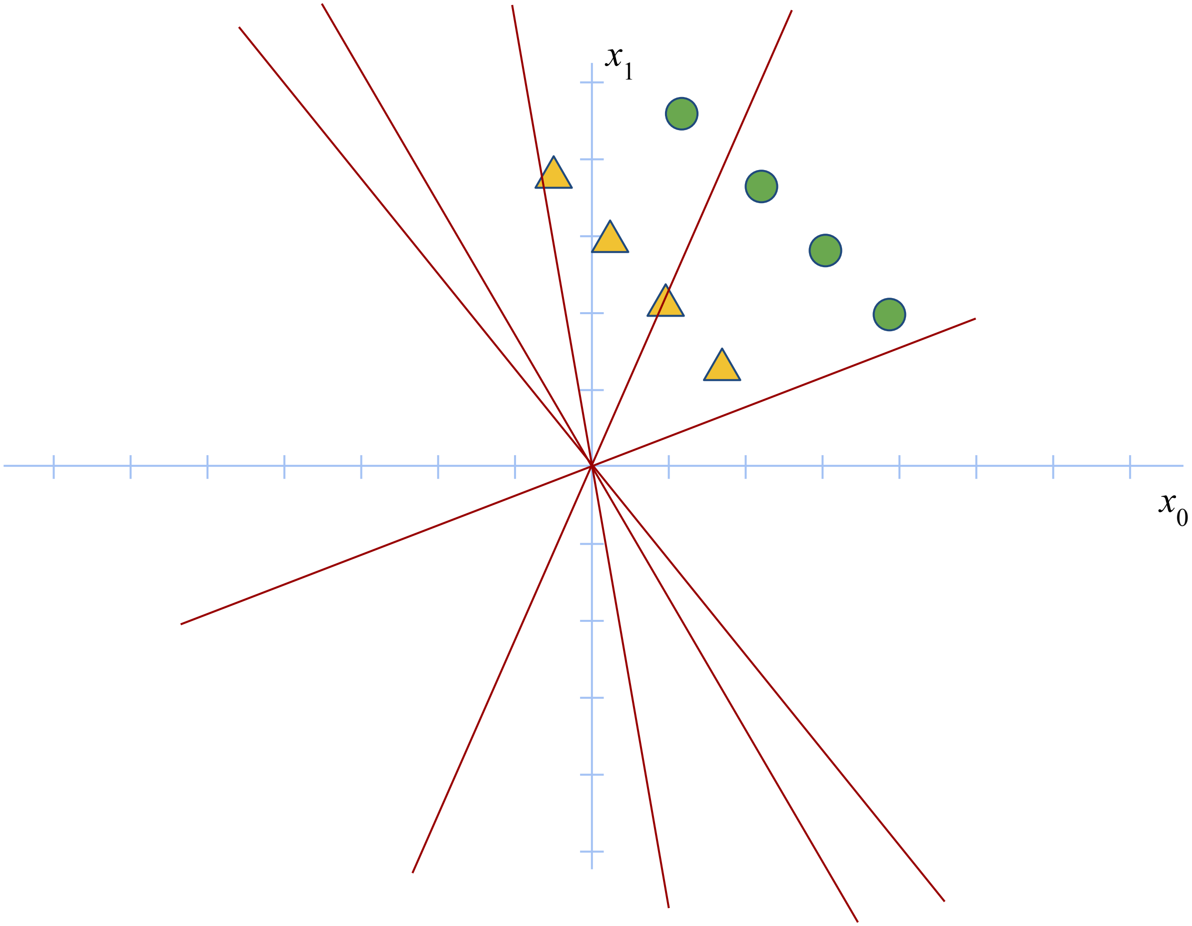
Consider the following dataset:



Without a bias term, the decision boundary *must* pass through the origin

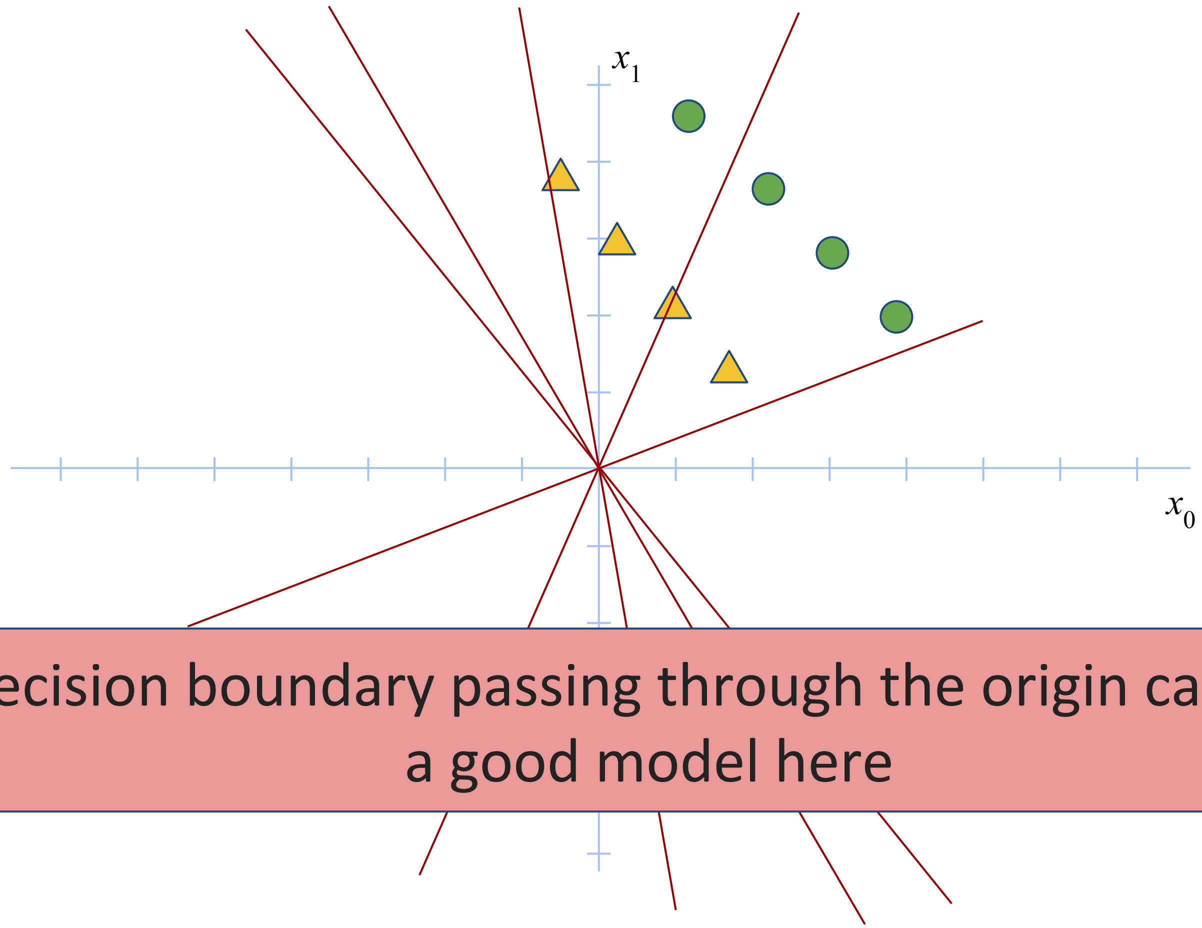
Bias

Consider the following dataset:



Bias

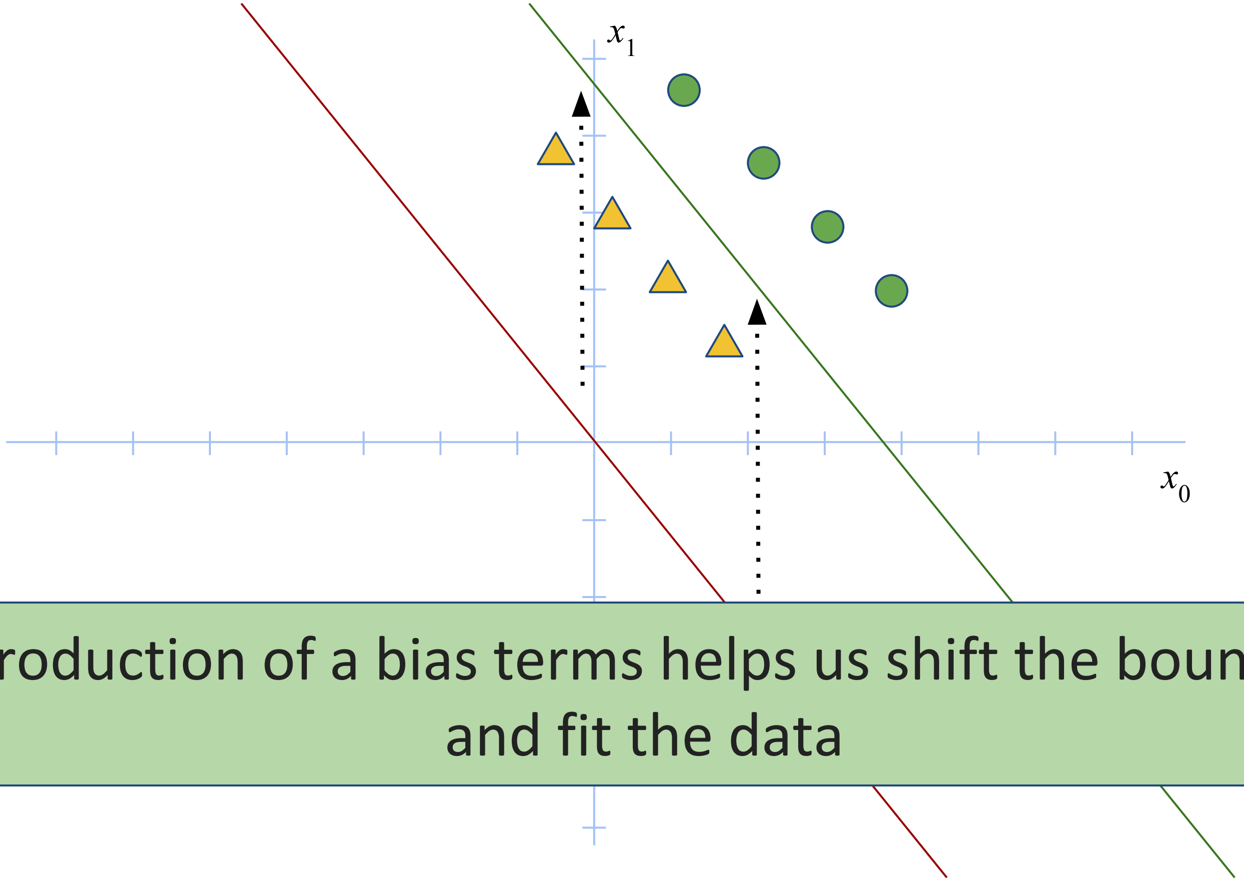
Consider the following dataset:



No decision boundary passing through the origin can lead to a good model here

Bias

Consider the following dataset:



Introduction of a bias terms helps us shift the boundary and fit the data

Parameter Initialization

Parameter Initialization

- We’ve learned that we need to move in the direction of the gradient to reach the minimum value for a given loss function
- But where do we start?

Parameter Initialization

- Initial values of W and b dictate where in the terrain we begin
- If we start near a minima, we can optimize very quickly - If we start too far, it may take a long time to find a good model
- We may even start near a local minimum and never find the global minimum for a given function

Parameter Initialization

- Zero initialization?
- Random initialization?
- Something more complicated?

Parameter Initialization

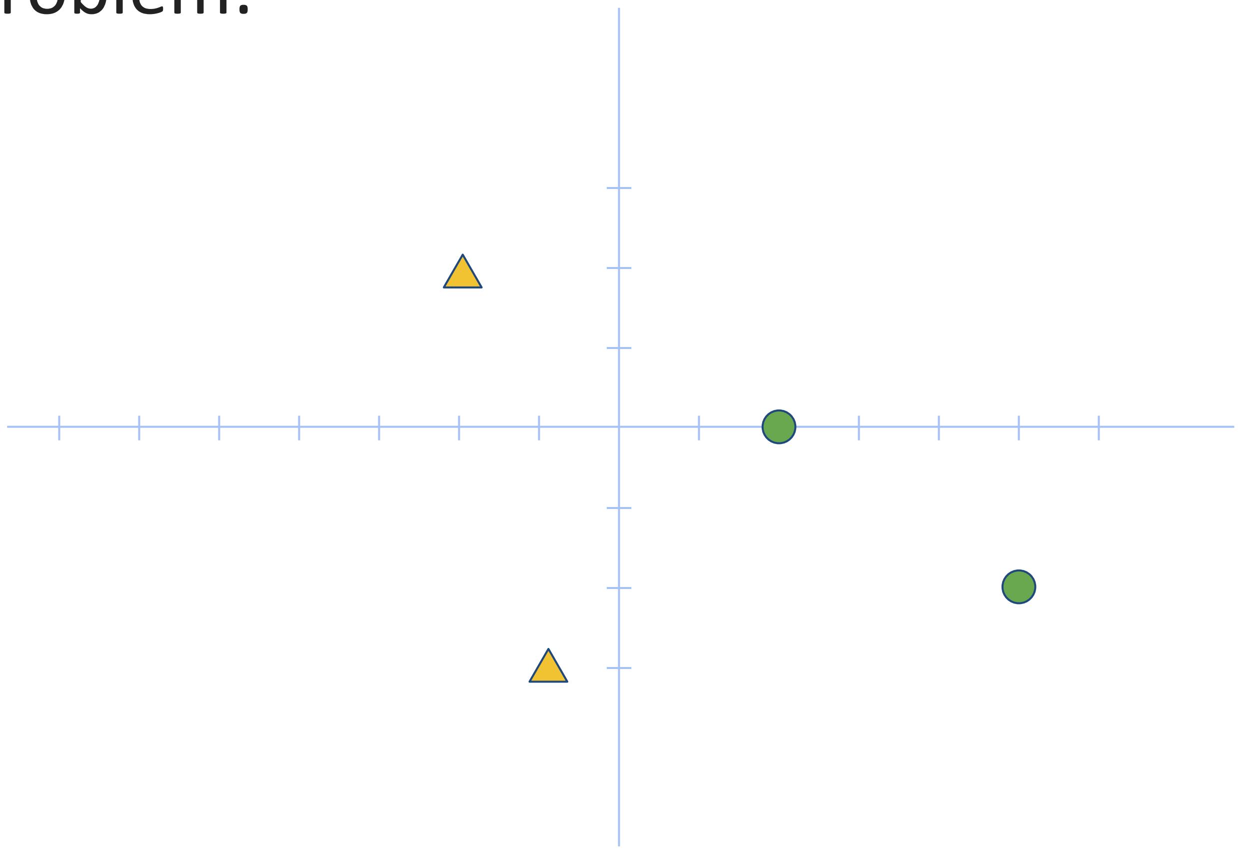
- Zero initialization
- Random initialization
- Something more complicated:
  - Gaussian distributed
  - Xavier Initialization

More on this later!

Regularization

Regularization

As we've seen, there are many potential solutions to a problem:



Regularization

As we've seen, there are many potential solutions to a problem:

$P_1: w_0 = 3; w_1 = 1; b = 3$

$P_2: w_0 = 300; w_1 = 100; b = 300$

$P_3: w_0 = 300; w_1 = 99; b = 300$

Which set of parameters is better here?

Regularization

Some loss functions are sensitive to the magnitude of weights:

```
Average losses (MSE)
```

$L(P_1) = 73.25$ $L(P_2) = 866051.0$ $L(P_3) = 867304.75$

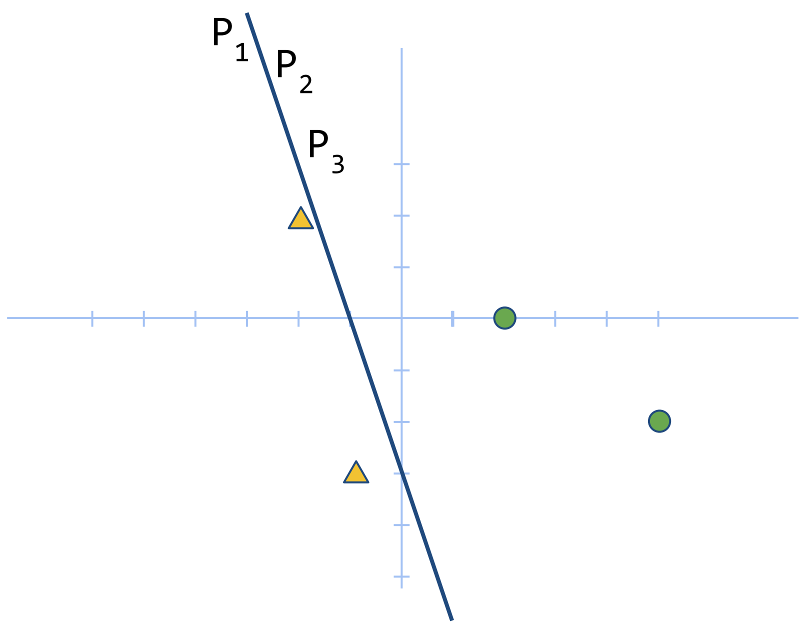
Regularization

Some loss functions are sensitive to the magnitude of weights:

```
Average losses (MSE)
```

$L(P_1) = 73.25$ $L(P_2) = 866051.0$ $L(P_3) = 867304.75$

But all three represent almost exactly the same boundary!



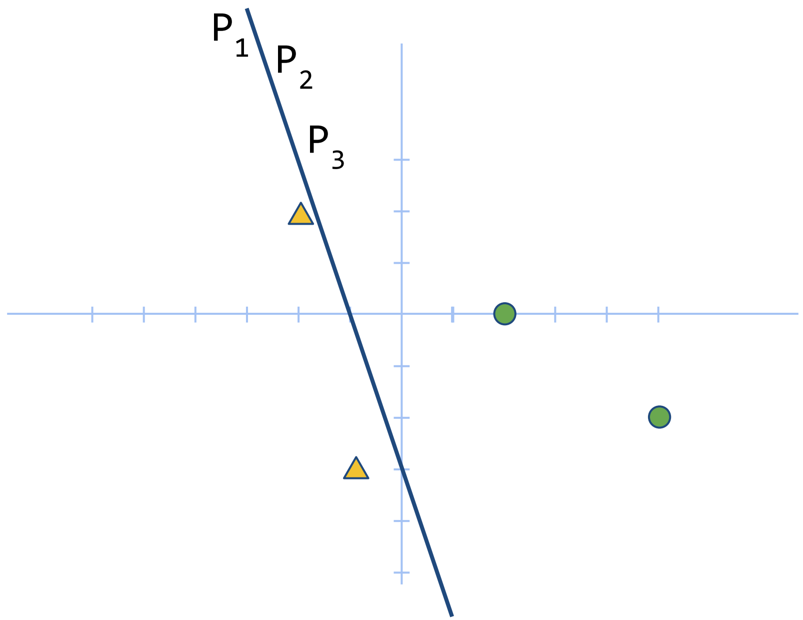
Regularization

Some loss functions are sensitive to the magnitude of weights:

Average losses (MSE)

$L(P_1) = 73.25$ $L(P_2) = 866051.0$ $L(P_3) = 867304.75$

Loss is different for each set of parameters, even though conceptually they are all equally good



Regularization

Solution: Since all these solutions are equally good, constrain our model to weights with small magnitude

Regularization

Solution: Since all these solutions are equally good, constrain our model to weights with small magnitude

$$L = \texttt{Normal loss} + \lambda \sum_{w} w_i^2$$

Penalizes weights that are too large
λ defines how much importance you want to give to regularization