Schedule

Block 3 (1:30 - 3:00 pm)

(5) Introduction to Deep Learning

(6) Understanding PyTorch

(7) Training Deep Neural Networks

30 min break (3:00 - 3:30 pm)

What is PyTorch?

What is PyTorch?

is free and open-source software

https://pytorch.org/

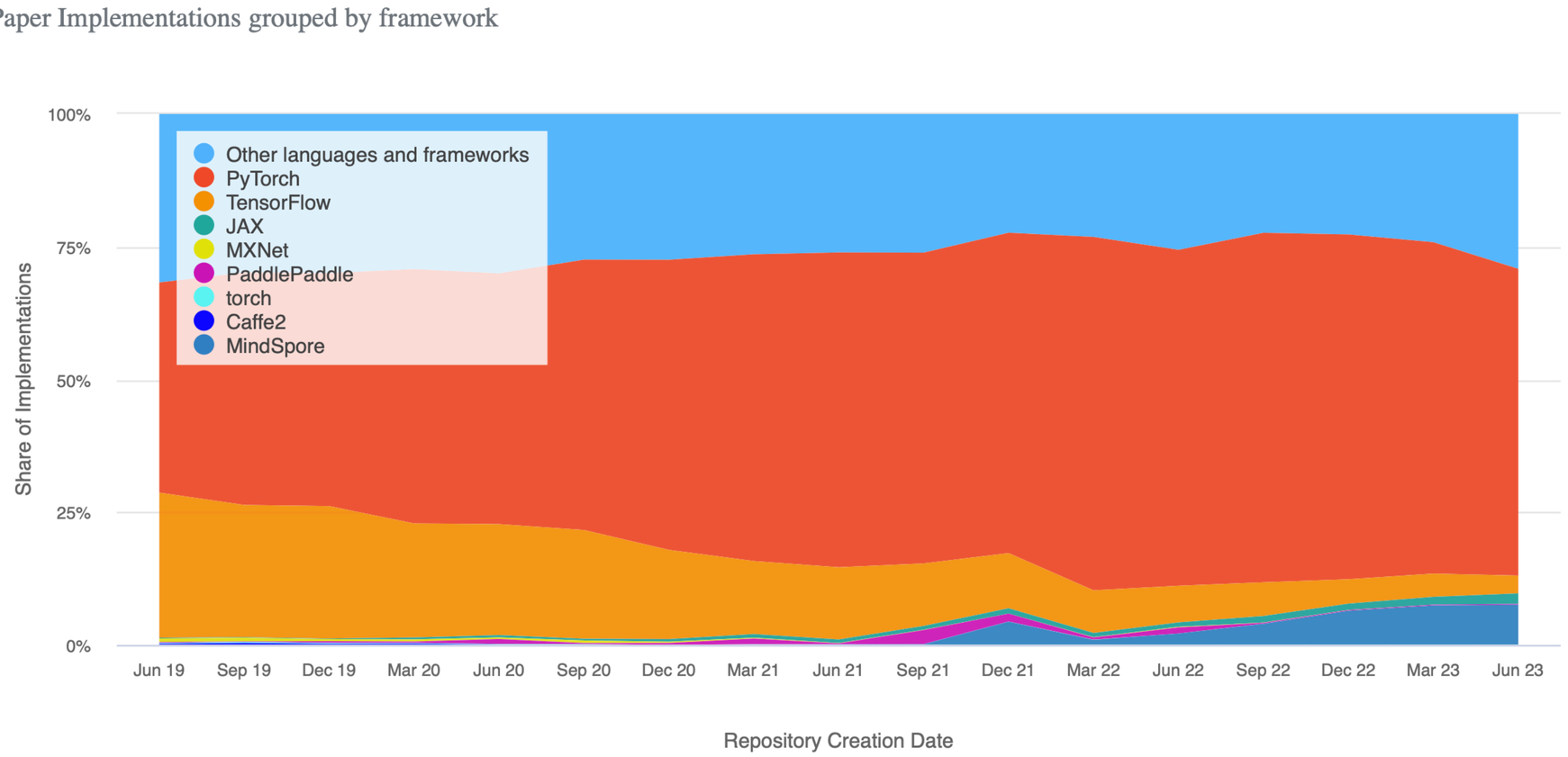
1 Tensor library

2 Automatic differentiation engine

What is PyTorch?

3 Deep learning library

PyTorch is widely used



https://paperswithcode.com/trends

What are tensors?

1 **Tensor library**

2 **Automatic differentiation engine**

What is PyTorch?

3 **Deep learning library**

Tensor library

Mathematically: a generalization of vectors, matrices, etc.

Computationally: a data container

Matrix
(rank-2 tensor)

```
a = torch.tensor([[1., 2., 3.],
                  [2., 3., 4.]])
a.shape
```

```
torch.Size([2, 3])
```

Tensor library

Tensor library == array library

Tensor library

torch.tensor ≈ numpy.array

+ GPU support

+ automatic differentiation support

1 Tensor library

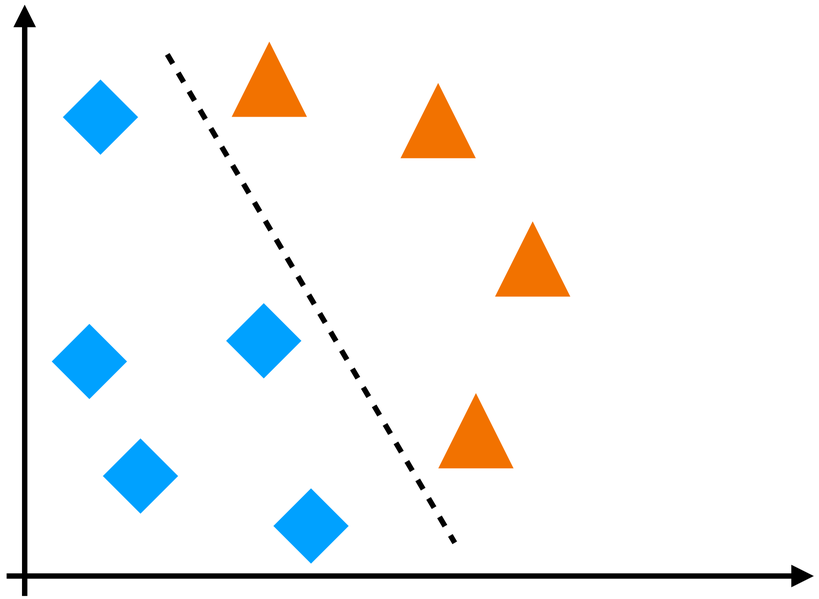
2 **Automatic differentiation engine**

What is PyTorch?

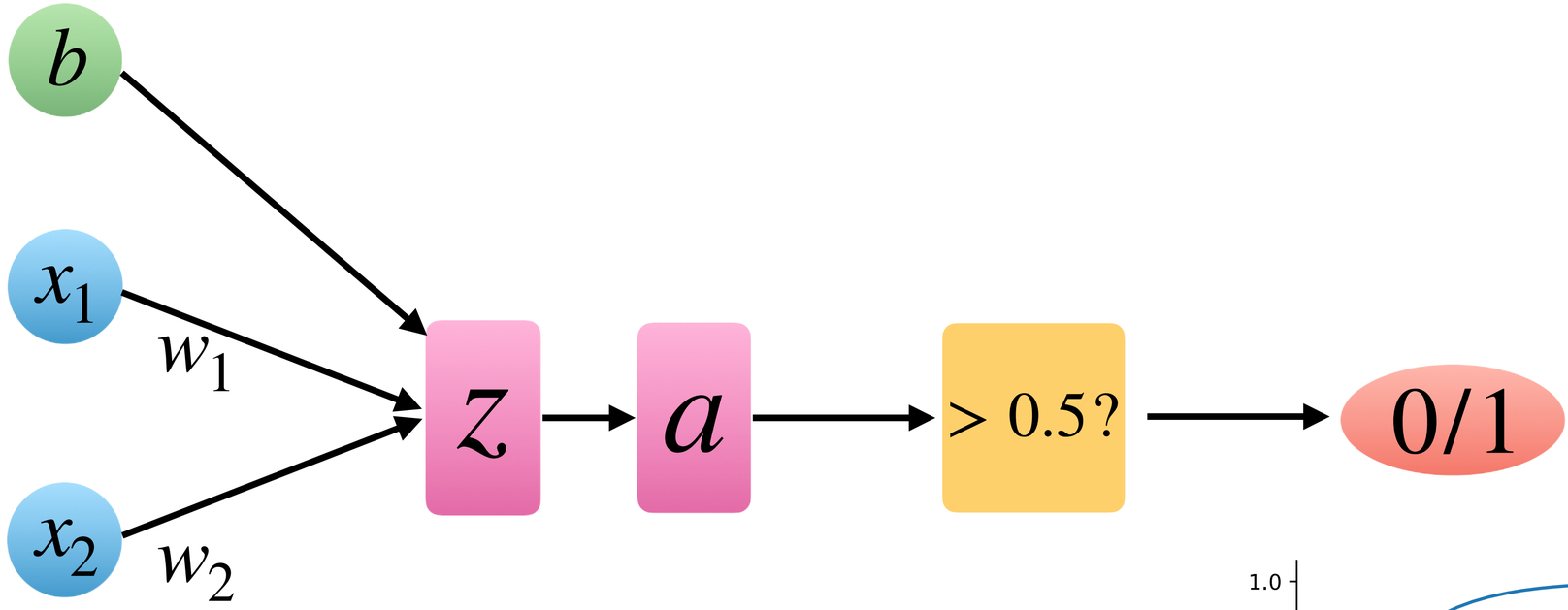
3 Deep learning library

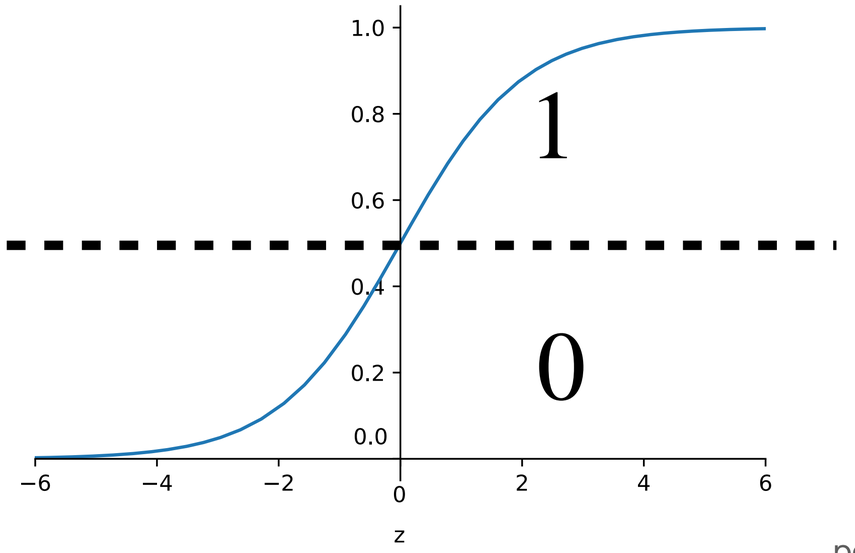
Advantage of Logistic Regression

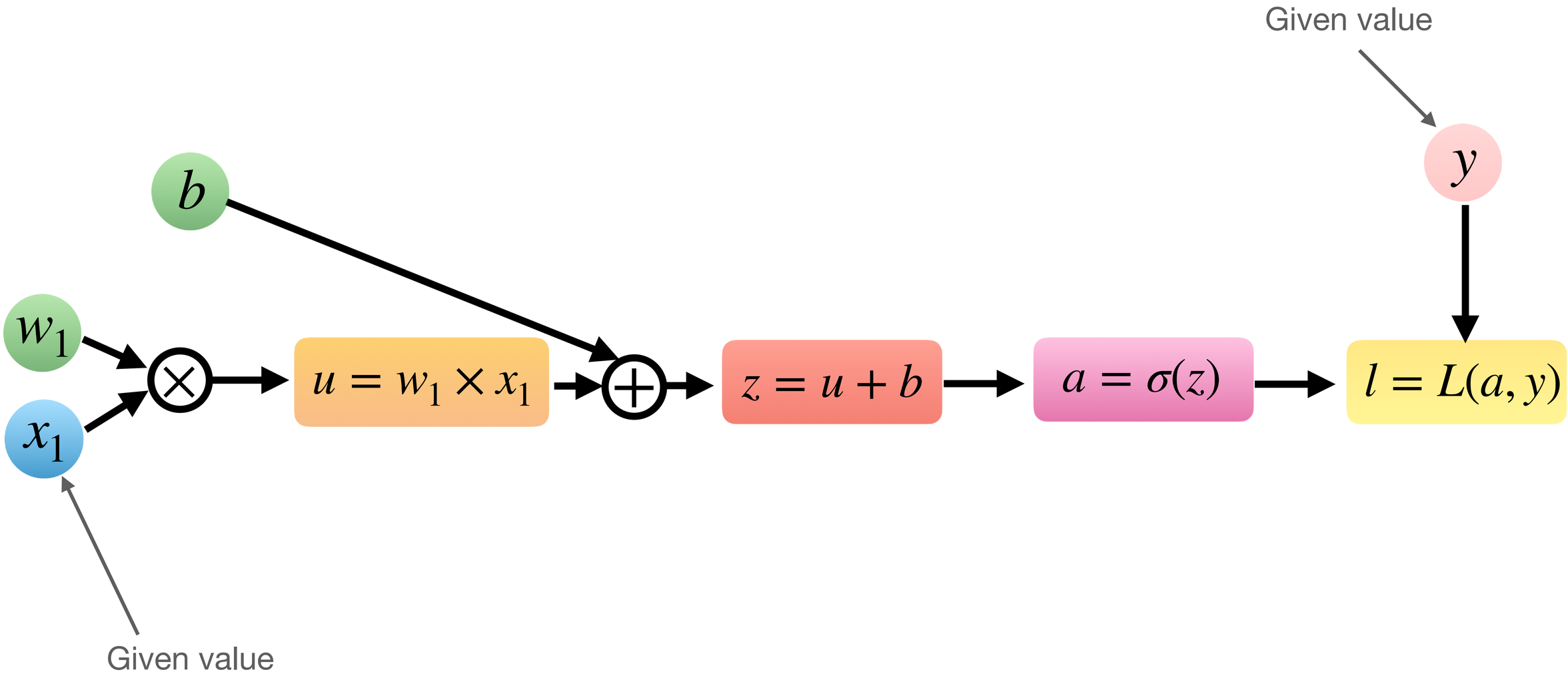
Logistic regression always converges

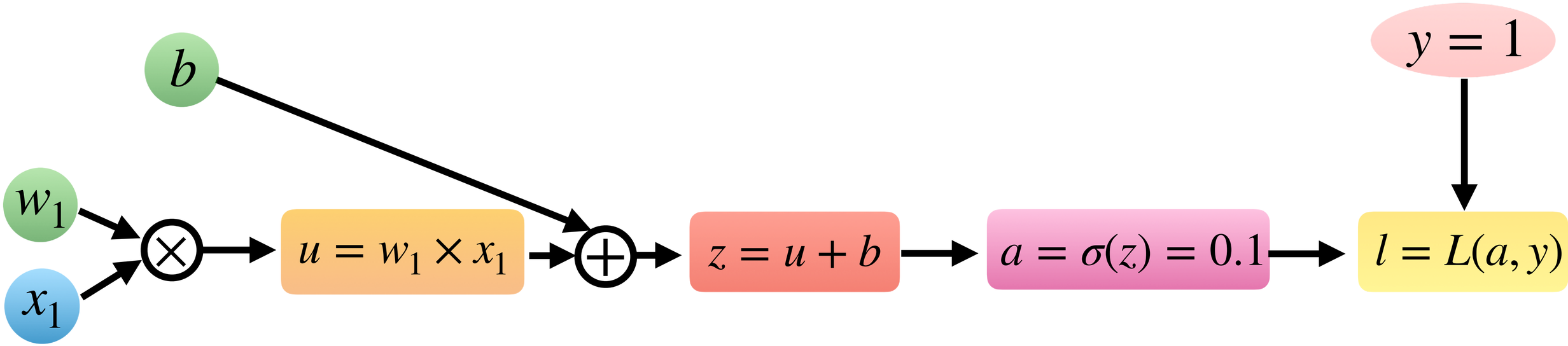


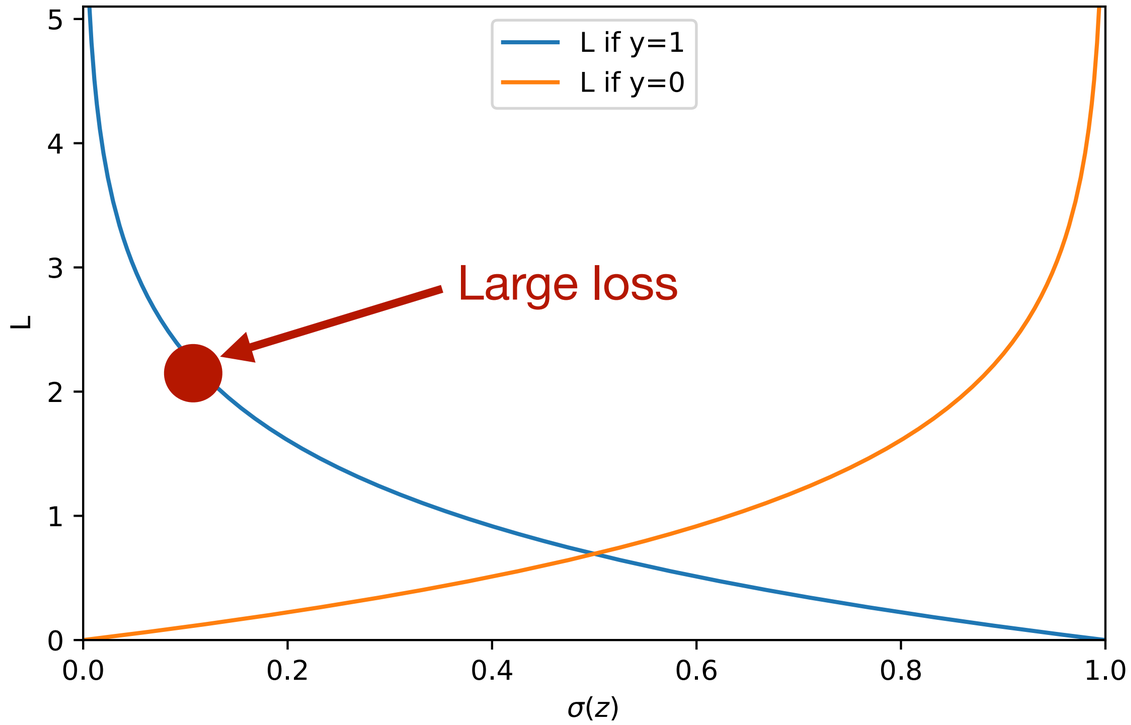
Thresholding the Logistic Sigmoid Function

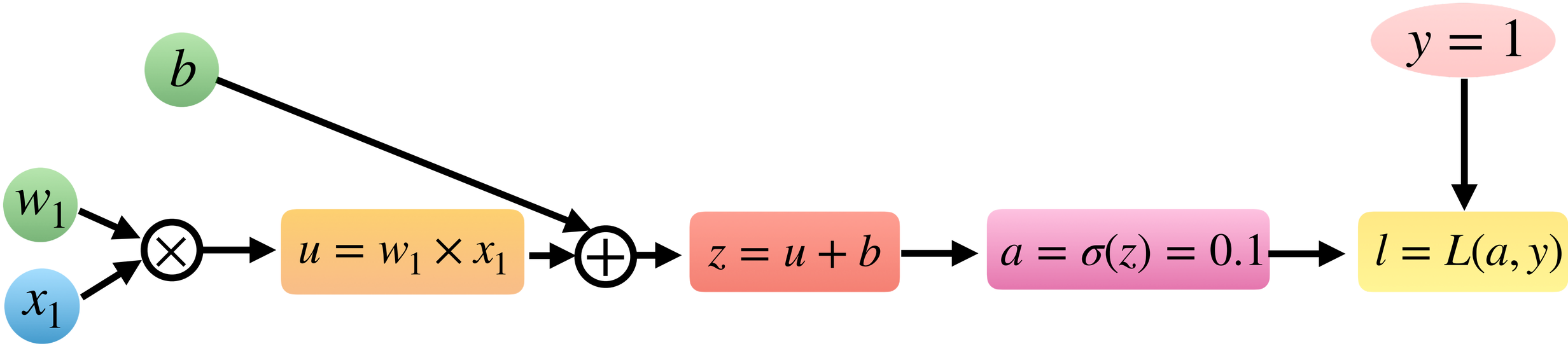


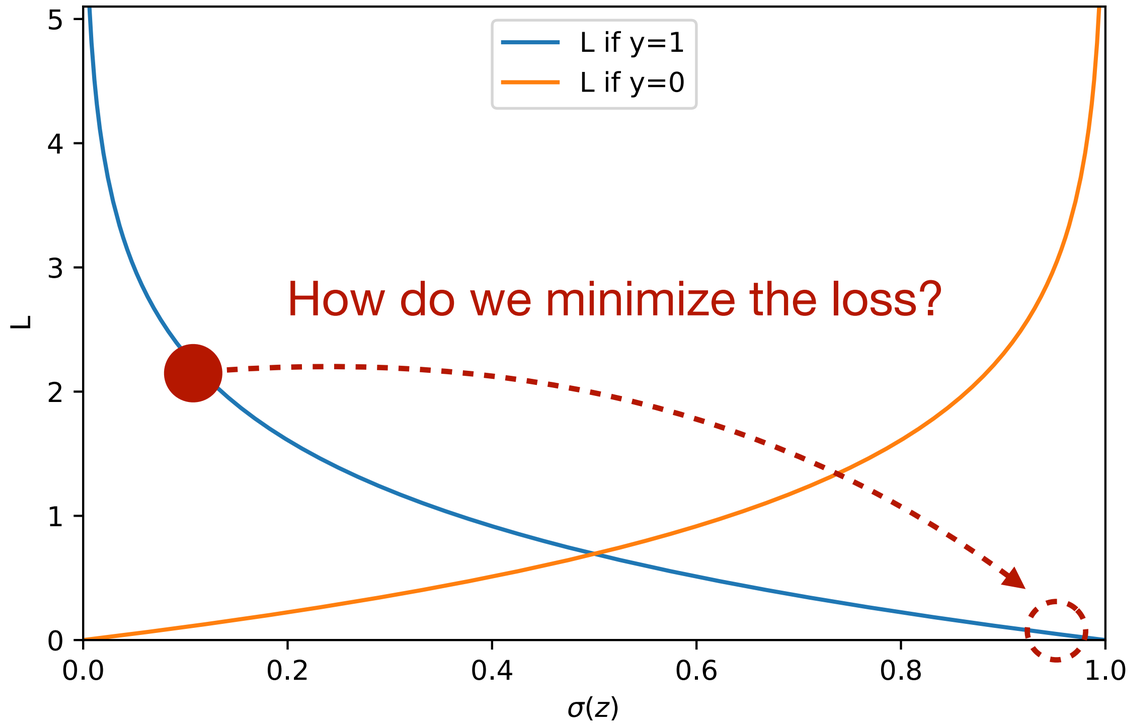


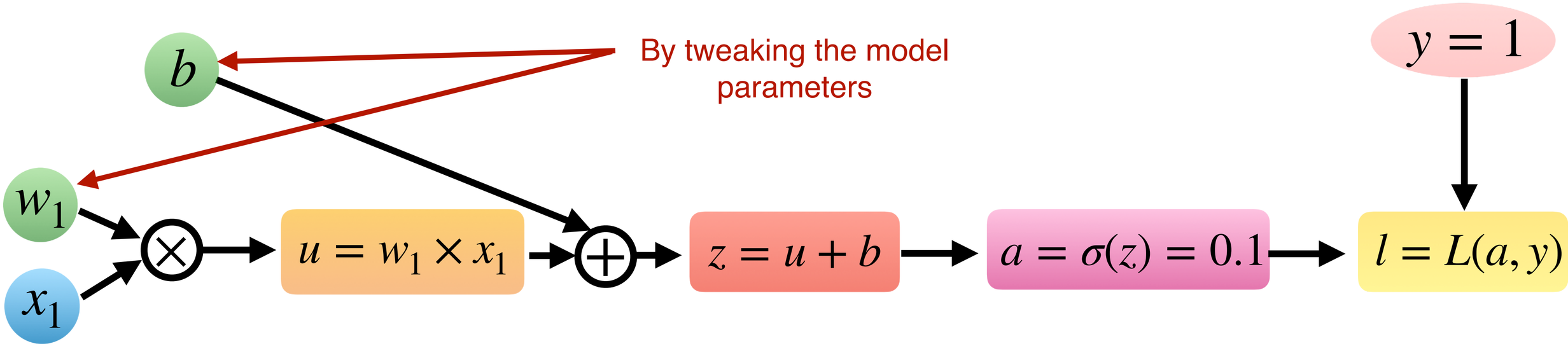

Given value
y
b
w_1
x_1
$u = w_1 \times x_1$
$z = u + b$
$a = \sigma(z)$
$l = L(a, y)$
Given value

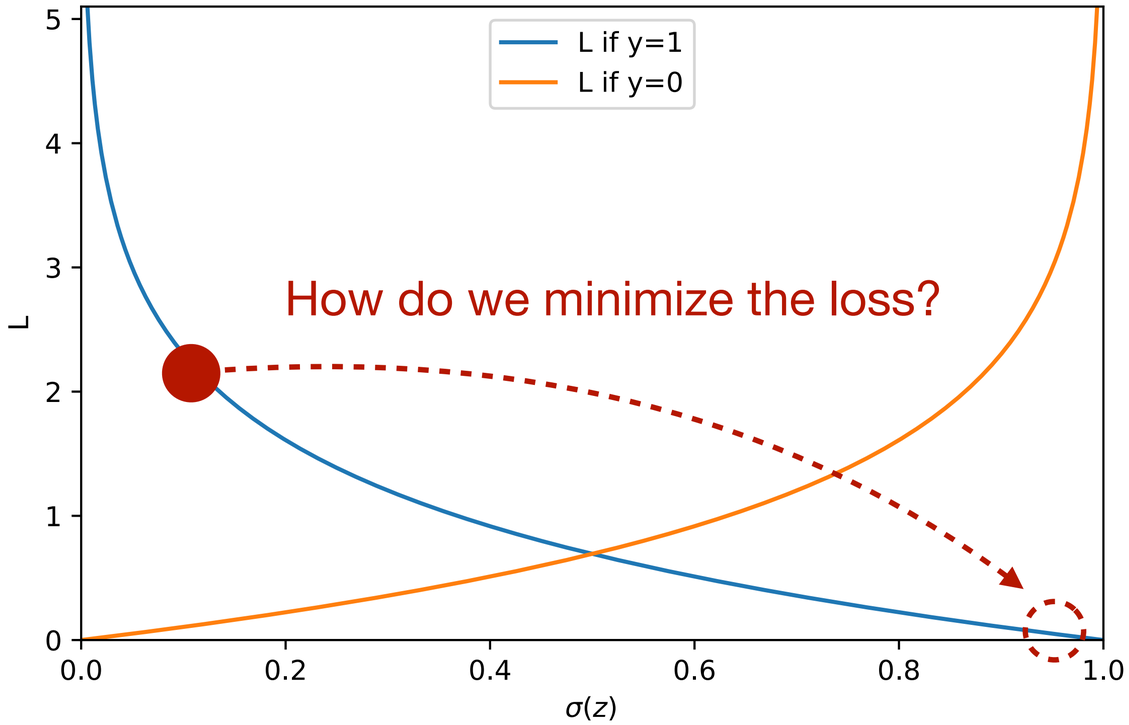

b
w_1
x_1
$u = w_1 \times x_1$
$z = u + b$
$a = \sigma(z) = 0.1$
$y = 1$
$l = L(a, y)$


L if y=1
L if y=0
Large loss
L
σ(z)
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0

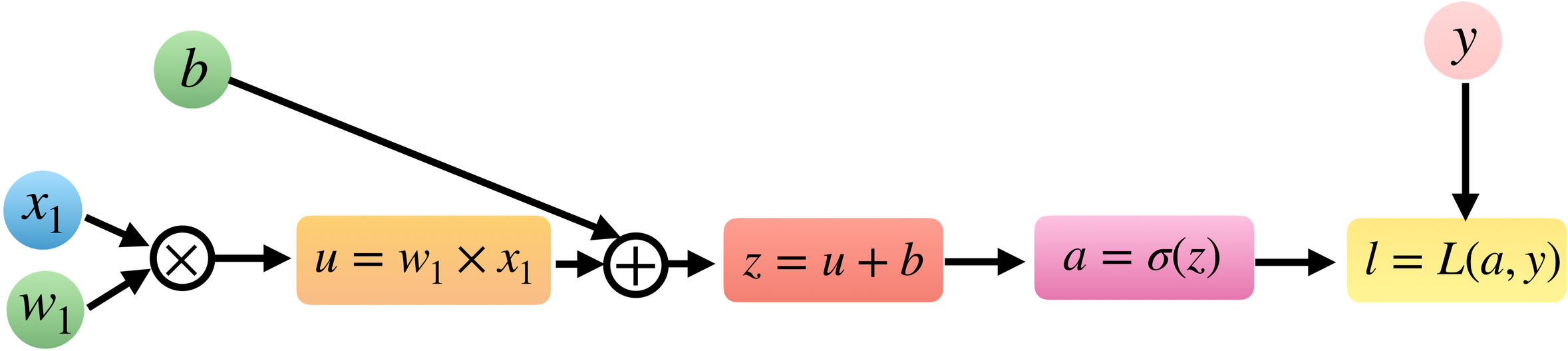

b
w_1
x_1
$u = w_1 \times x_1$
$z = u + b$
$a = \sigma(z) = 0.1$
$y = 1$
$l = L(a, y)$


L if y=1
L if y=0
How do we minimize the loss?
L
$\sigma(z)$
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0


By tweaking the model parameters
b
w_1
x_1
$u = w_1 \times x_1$
$z = u + b$
$a = \sigma(z) = 0.1$
$y = 1$
$l = L(a, y)$

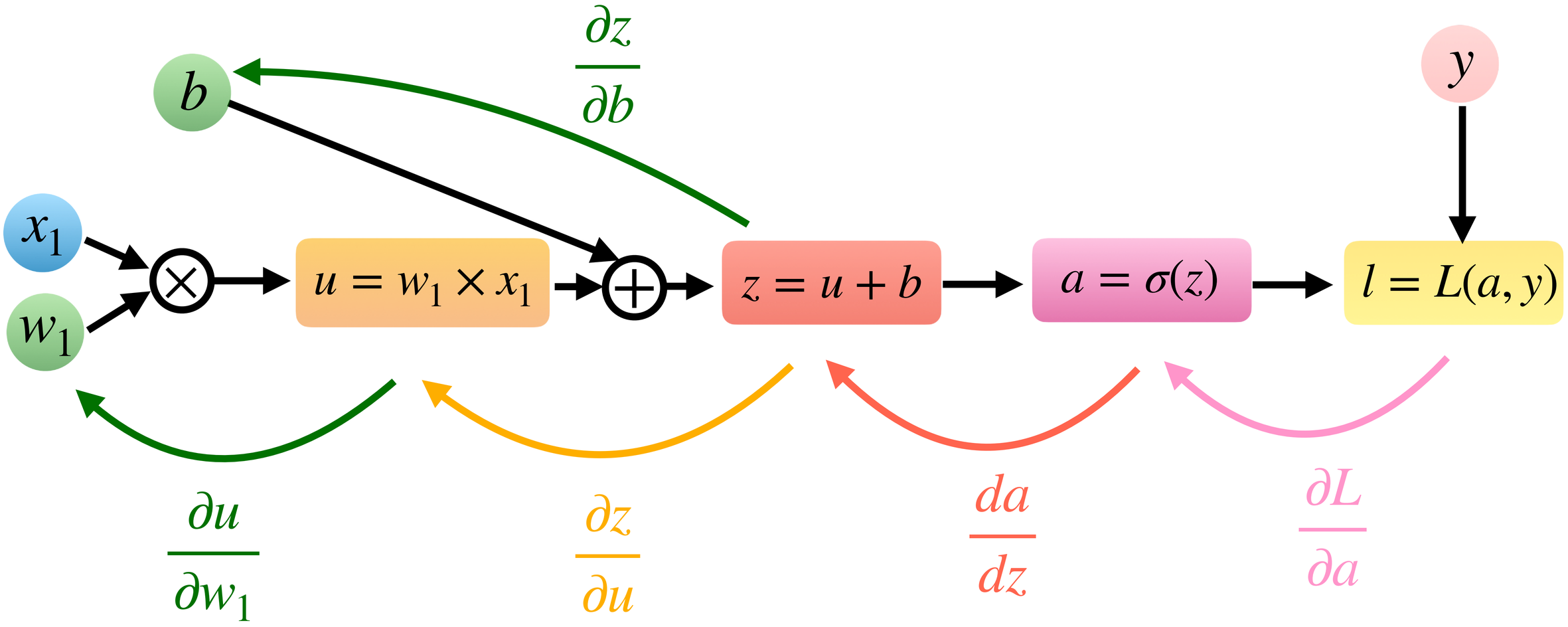

L if y=1
L if y=0
How do we minimize the loss?
L
$\sigma(z)$
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0

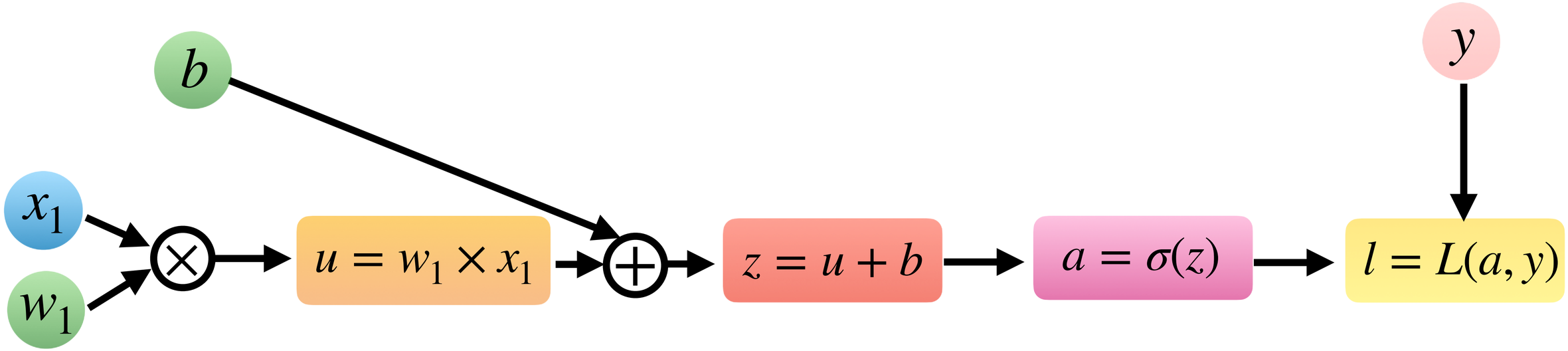
Chain Rule



$$\frac{\partial L}{\partial w_1} = \frac{\partial u}{\partial w_1} \times \frac{\partial z}{\partial u} \times \frac{da}{dz} \times \frac{\partial L}{\partial a}$$

Chain Rule


$\frac{\partial z}{\partial b}$
b
y
x_1
w_1
$u = w_1 \times x_1$
$z = u + b$
$a = \sigma(z)$
$l = L(a, y)$
$\frac{\partial u}{\partial w_1}$
$\frac{\partial z}{\partial u}$
$\frac{da}{dz}$
$\frac{\partial L}{\partial a}$



$$\frac{\partial L}{\partial w_1} =$$

```
from torch.autograd import grad

grad_L_w1 = grad(l, w_1, retain_graph=True)
print(grad_L_w1)
```

```
(tensor([-0.4987]),)
```

$$\frac{\partial L}{\partial b} =$$

```
grad_L_b = grad(l, b, retain_graph=True)
print(grad_L_b)
```

```
(tensor([-0.4054]),)
```

Or even easier: use .backward()

```
l.backward()
```

$$\frac{\partial L}{\partial w_1} =$$

```
w_1.grad
```

```
tensor([-0.4987])
```

$$\frac{\partial L}{\partial b} =$$

```
b.grad
```

```
tensor([-0.4054])
```

1 Tensor library

2 Automatic differentiation engine

What is PyTorch?

3 **Deep learning library**

1) Define the Model

```python
import torch

class MyClassifier(torch.nn.Module):
    def __init__(self):
        # define model parameters

    def forward(self, x):
        # define how the model
        # produces outputs

        return outputs
```

2) Train the Model

```
model = MyClassifier()
optimizer = torch.optim.SGD(...)

for epoch in range(num_epochs):
    for x, y in train_dataloader:

        # forward pass
        outputs = model(x)
        loss = loss_fn(outputs, y)

        # backward pass (backpropagation)
        optimizer.zero_grad()
        loss.backward()

        # update model parameters
        optimizer.step()
```

Exercise

Implementing a logistic regression classifier