



Modern Deep Learning with PyTorch

2. Understanding the PyTorch API (2:00 - 2:30 pm)

What is PyTorch?

What is PyTorch?

 **is free and open-source software**

<https://pytorch.org/>

1 Tensor library

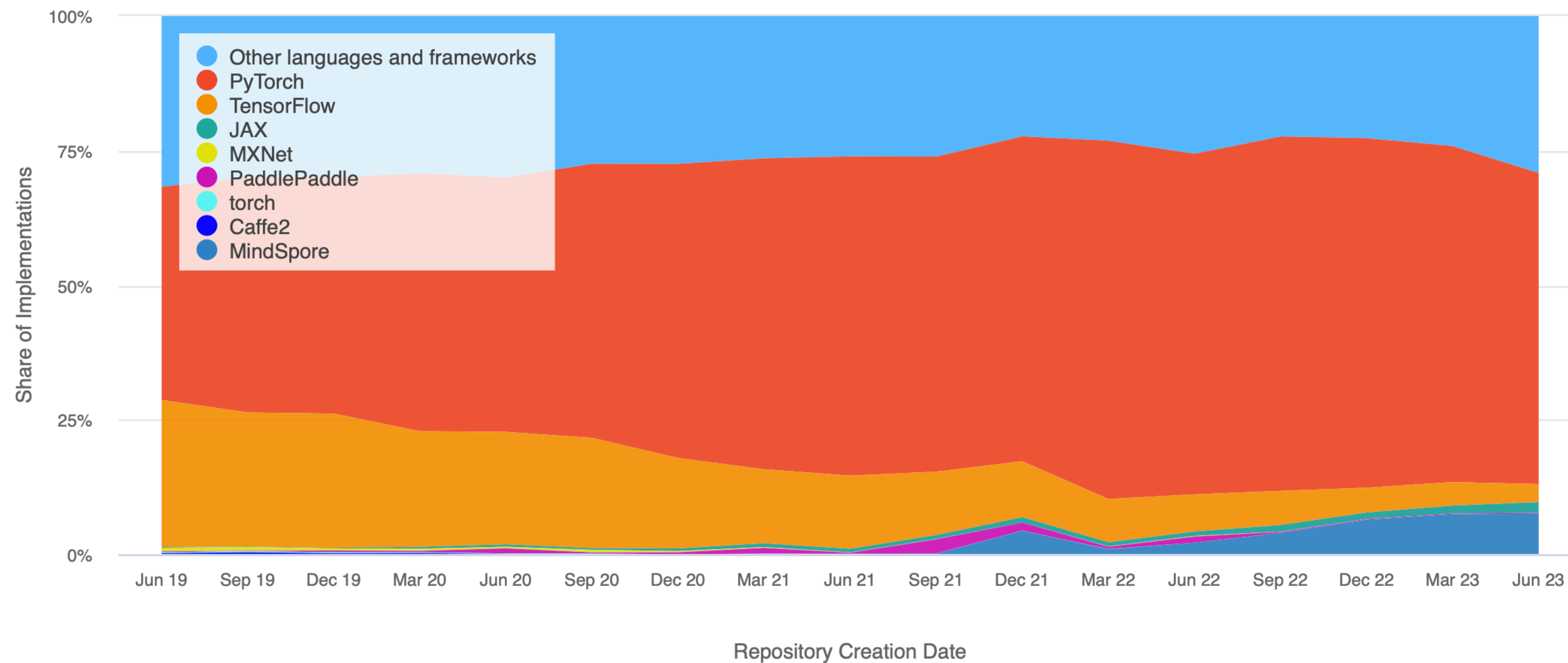
2 Automatic
differentiation engine

What is PyTorch?

3 Deep learning
library

PyTorch is widely used

Paper Implementations grouped by framework



<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` $\approx$ `numpy.array`

+ GPU support

+ automatic differentiation support

1 Tensor library

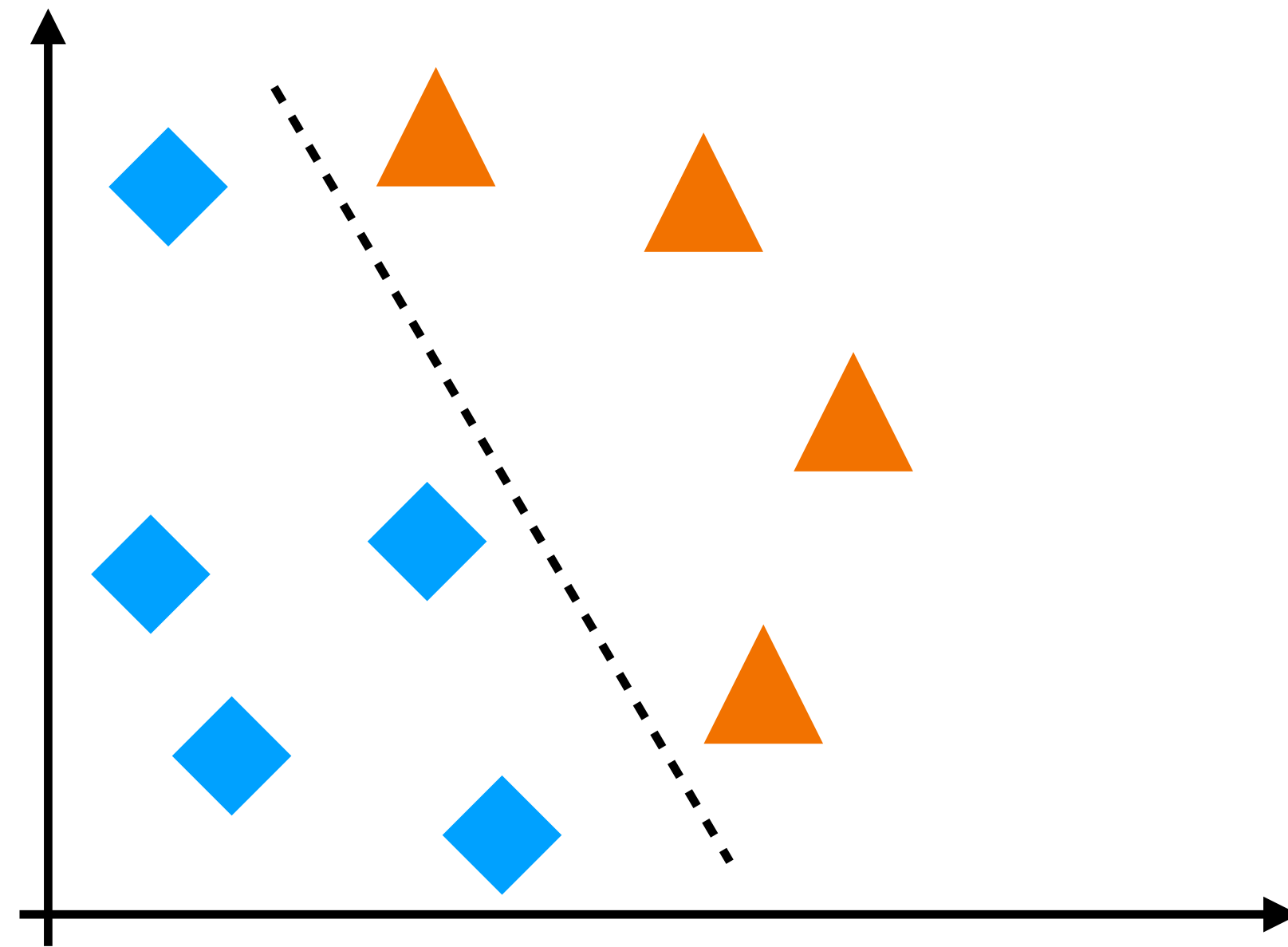
2 Automatic
differentiation engine

What is PyTorch?

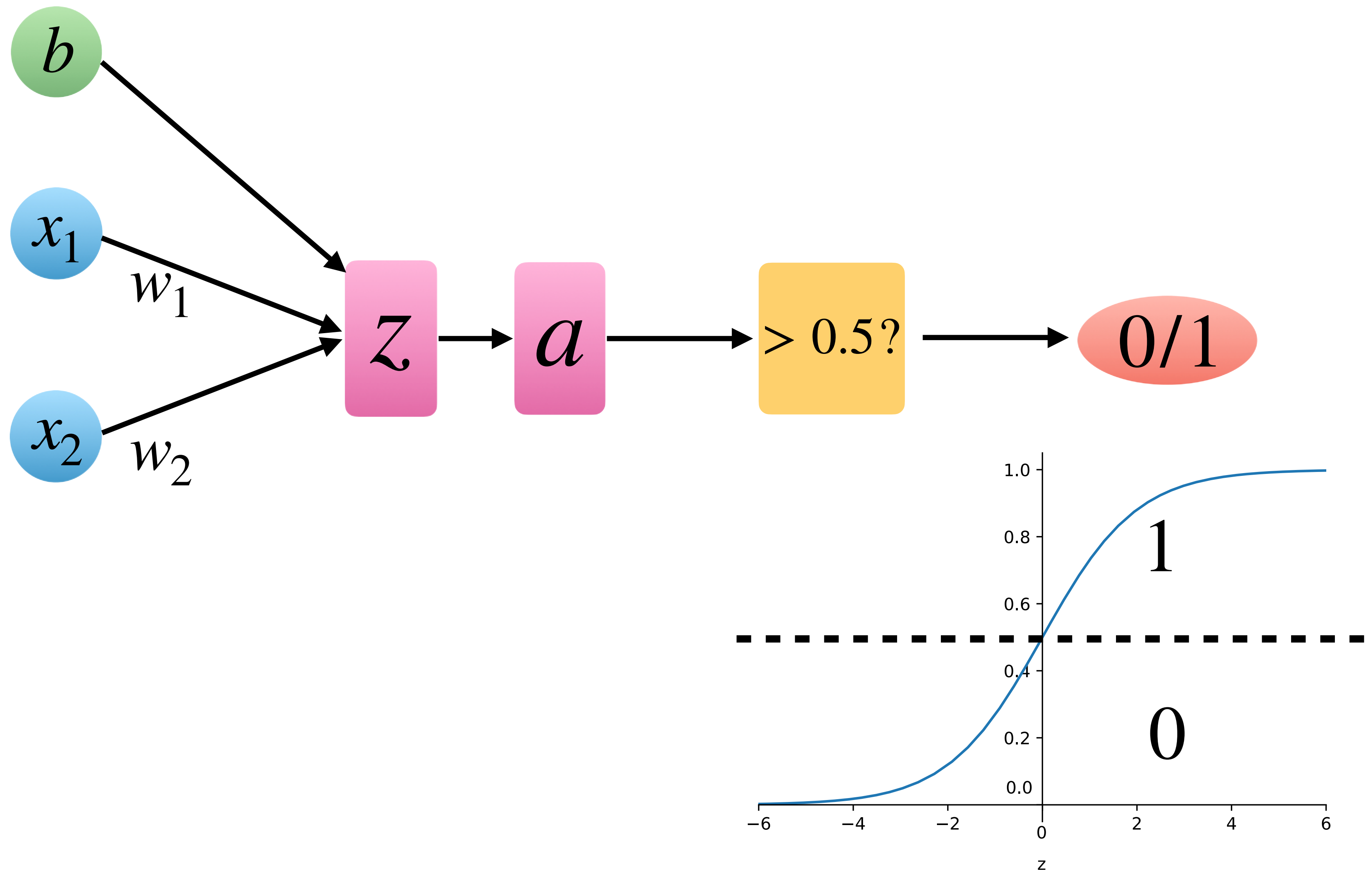
3 Deep learning
library

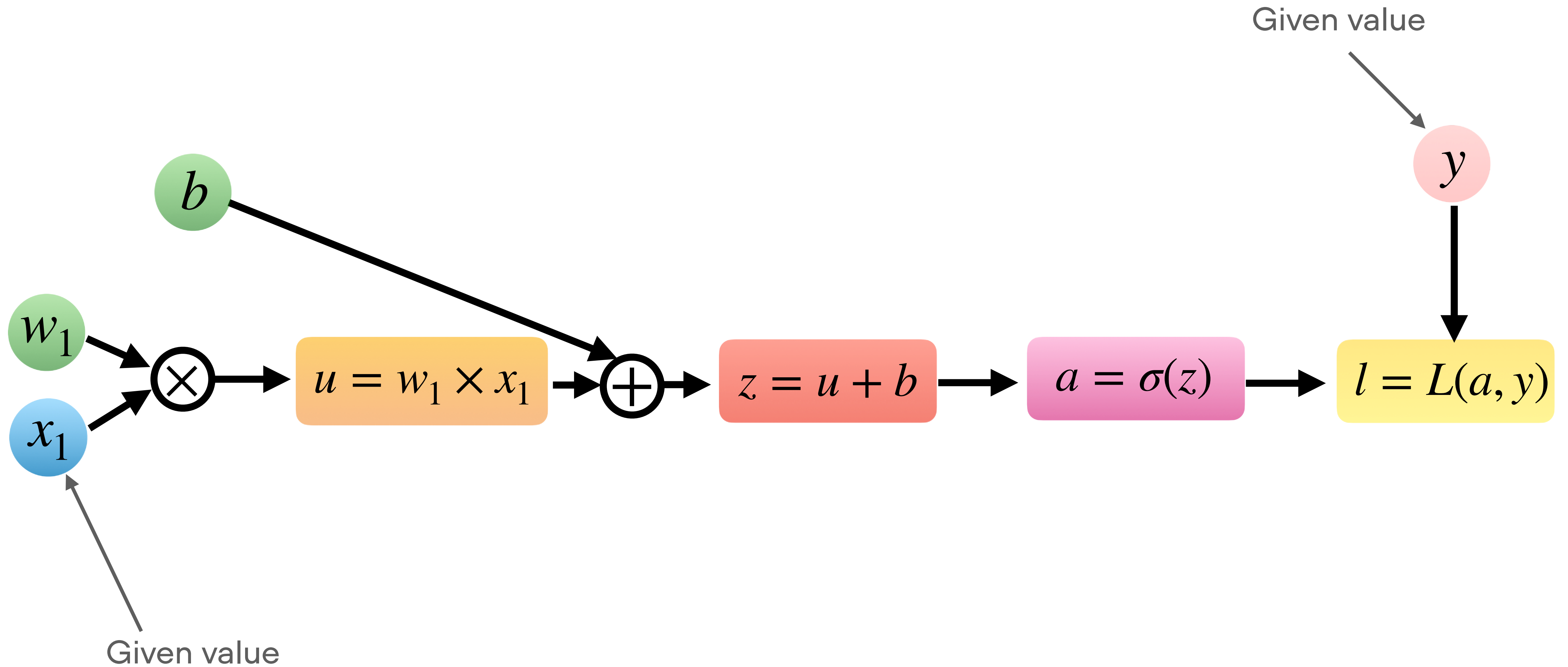
Advantage of Logistic Regression

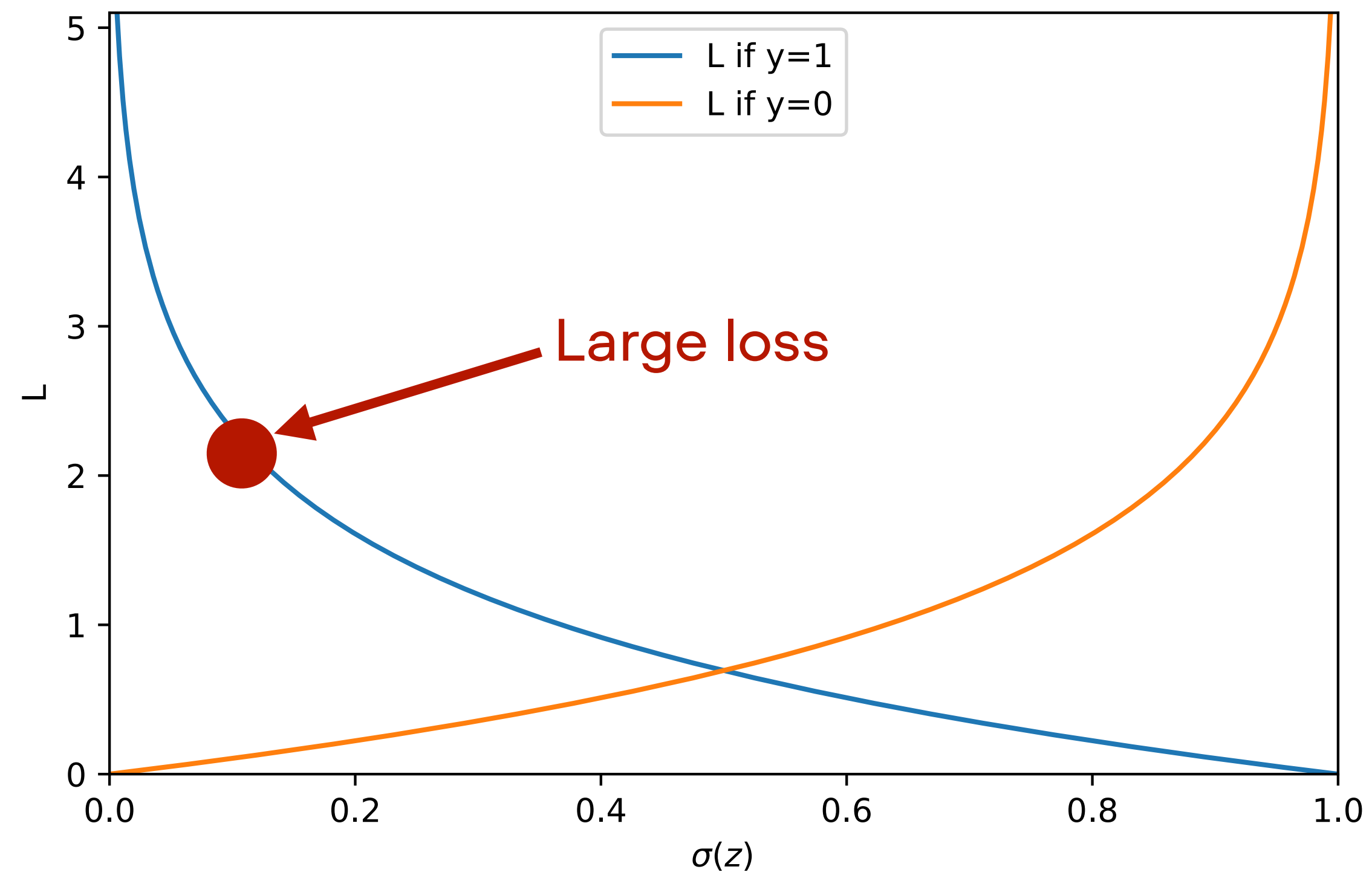
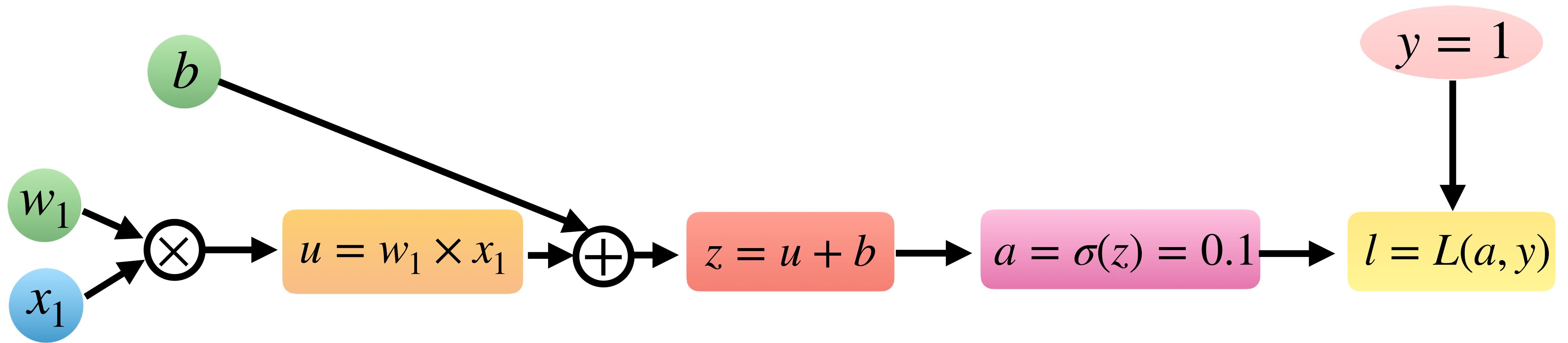
Logistic regression always converges

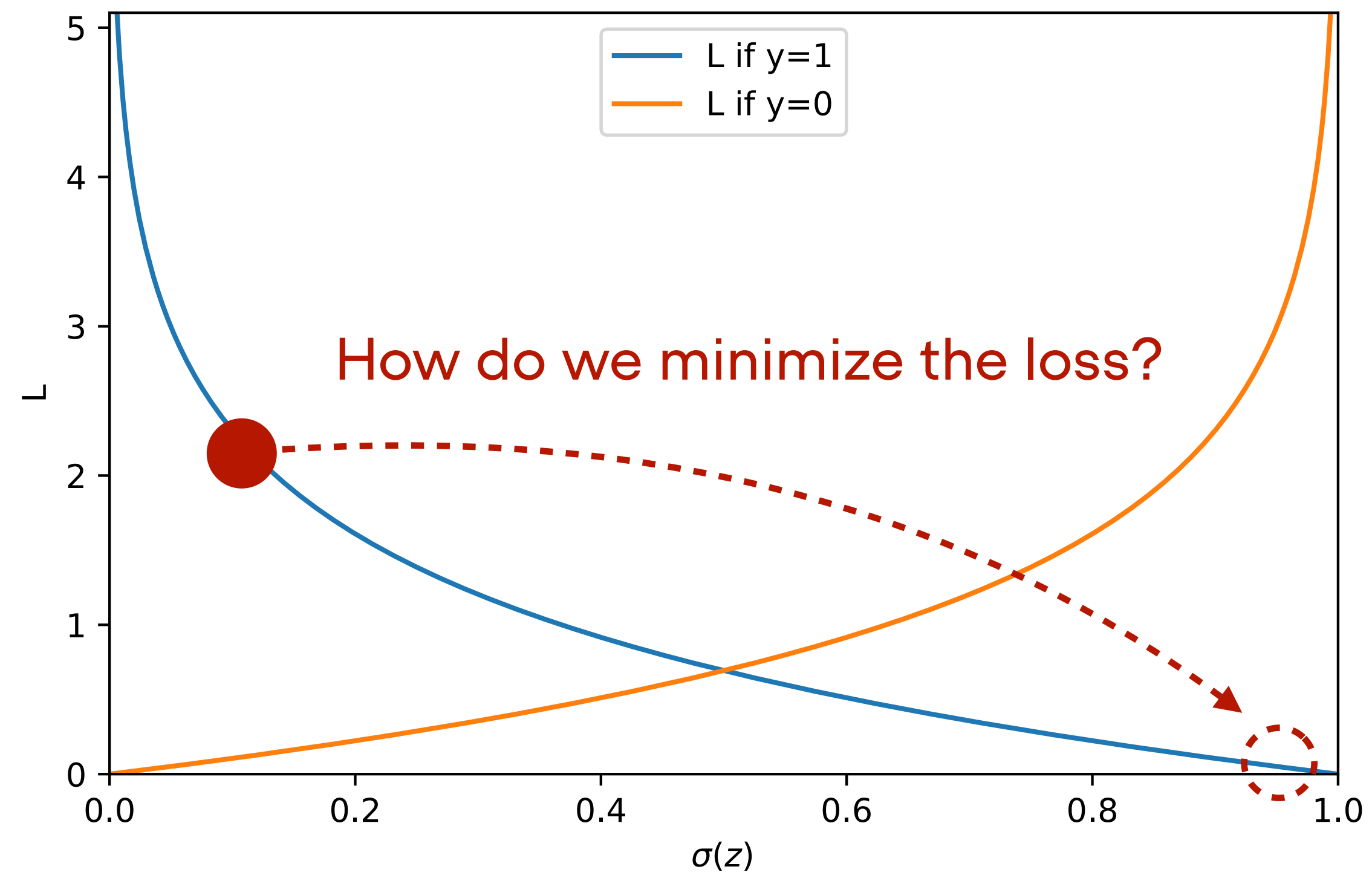
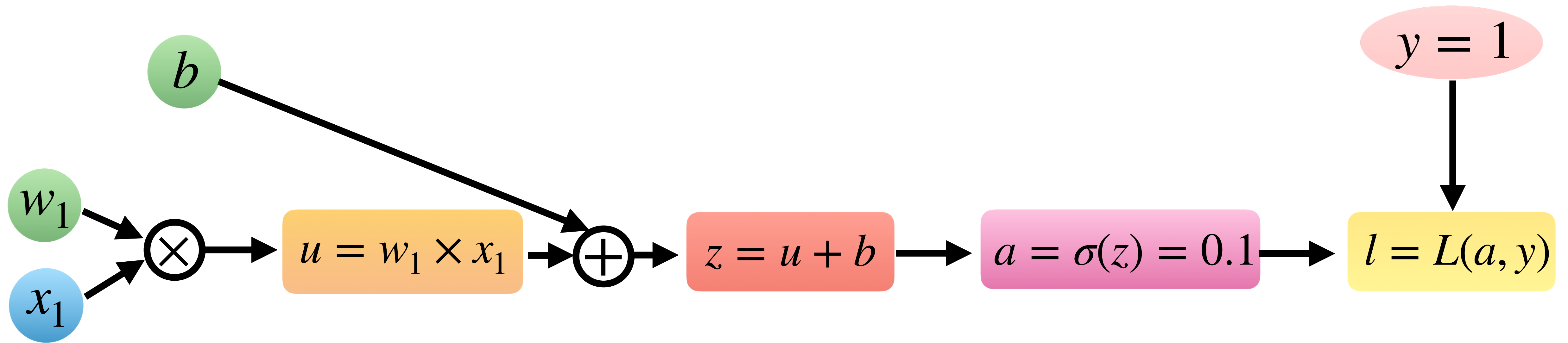


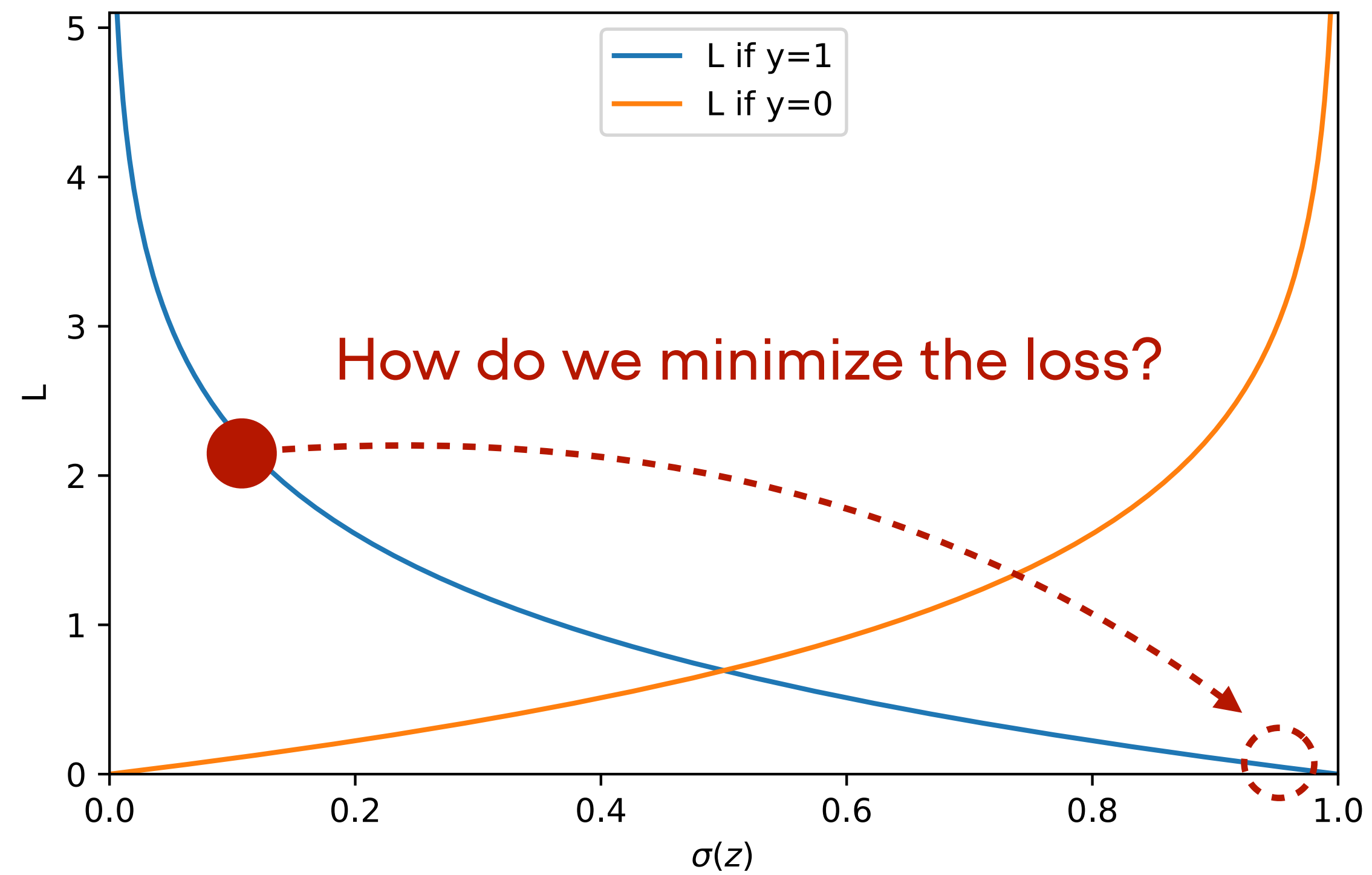
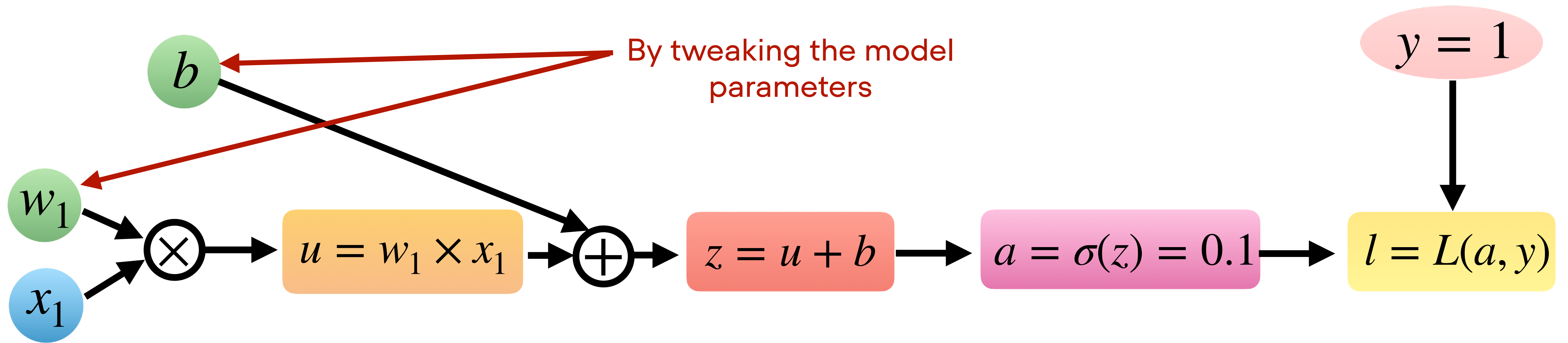
Thresholding the Logistic Sigmoid Function



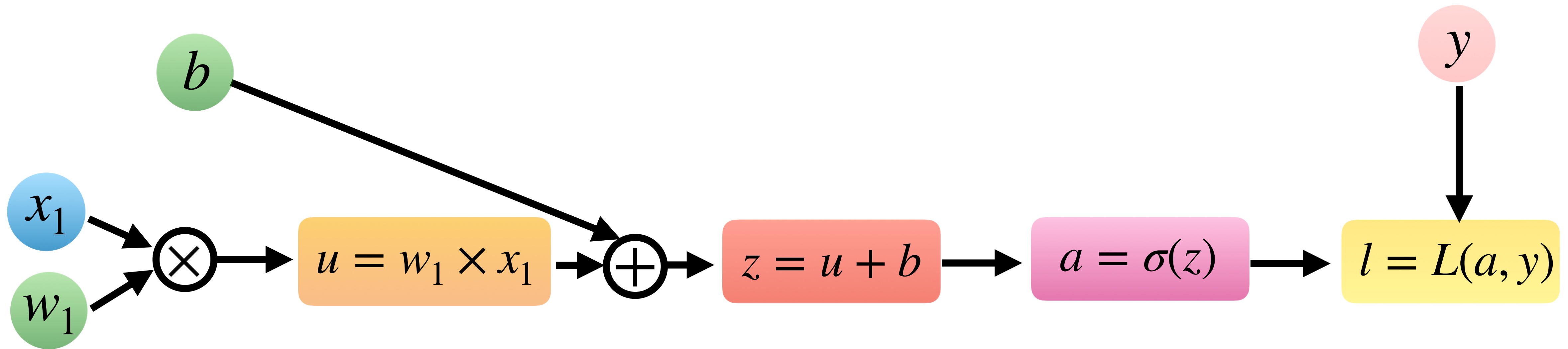






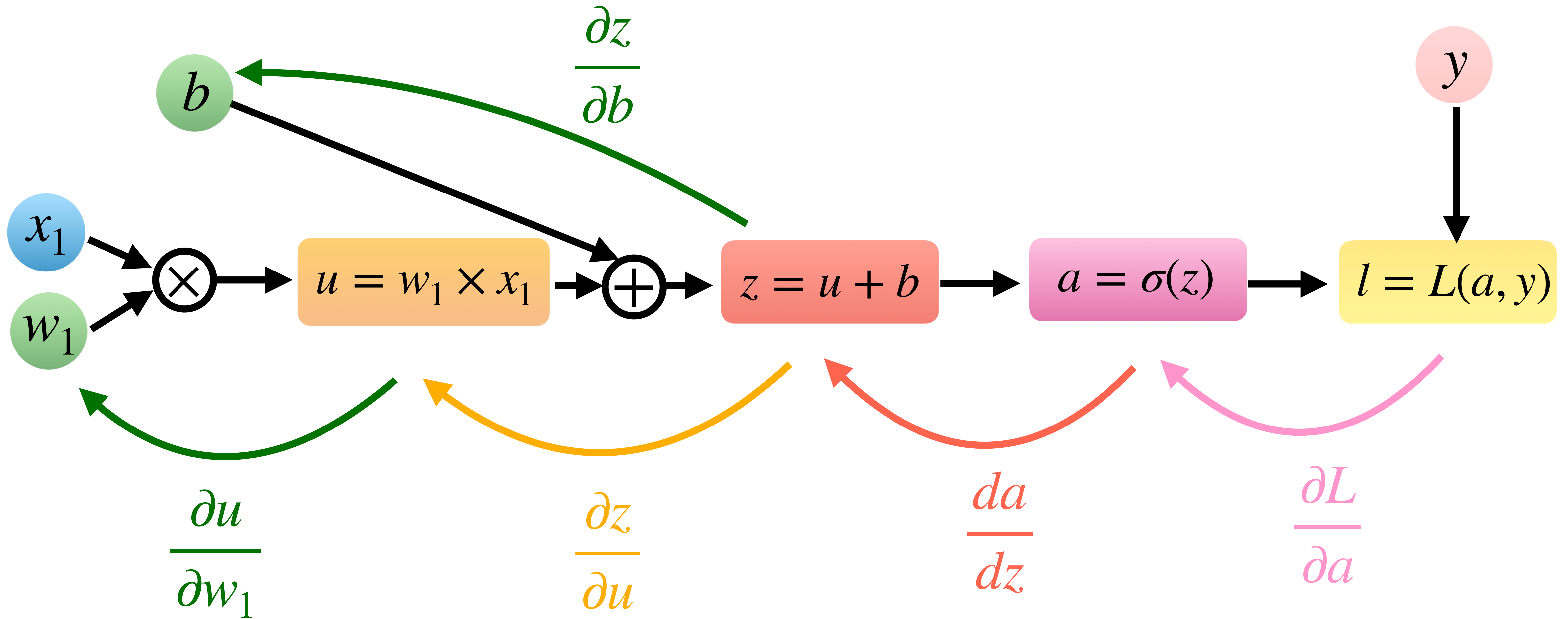


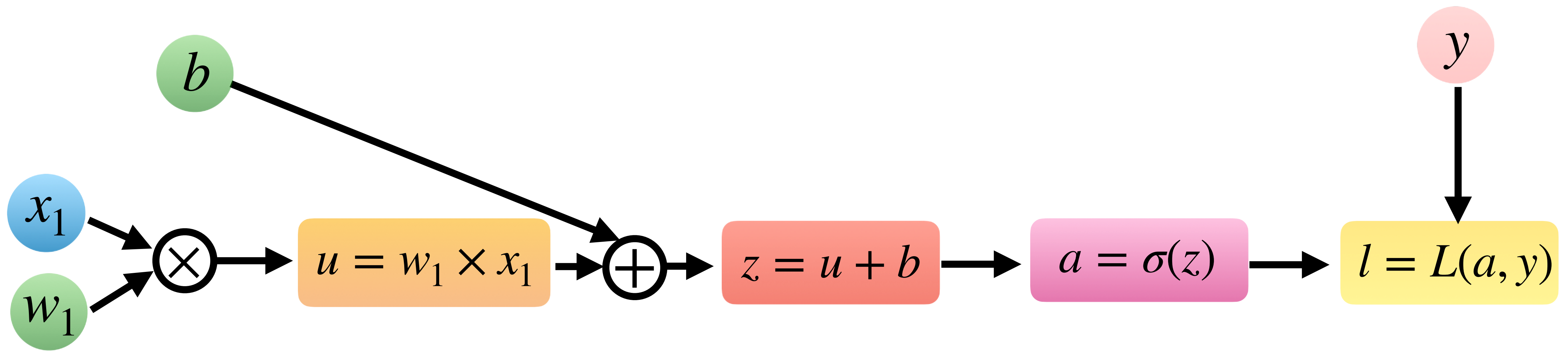
Chain Rule



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

Chain Rule





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

```

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



```
1  import torch
2
3  class MyClassifier(torch.nn.Module):
4      def __init__(self):
5          # define model parameters
6
7      def forward(self, x):
8          # define how the model
9          # produces outputs
10
11      return outputs
```

2) Train the Model



```
1  model = MyClassifier()
2  optimizer = torch.optim.SGD(...)
3
4  for epoch in range(num_epochs):
5      for x, y in train_data_loader:
6
7          # forward pass
8          outputs = model(x)
9          loss = loss_fn(outputs, y)
10
11         # backward pass (backpropagation)
12         optimizer.zero_grad()
13         loss.backward()
14
15         # update model parameters
16         optimizer.step()
```


Exercise

Implementing a logistic regression classifier