

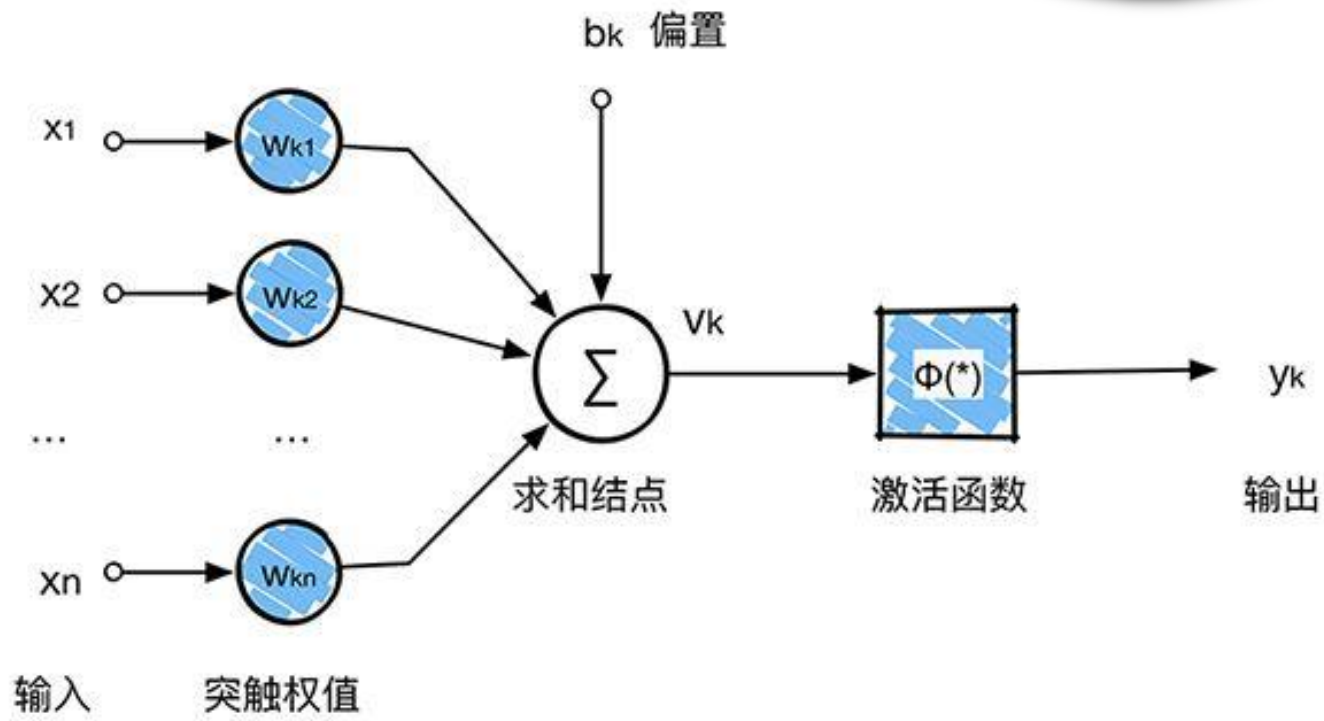
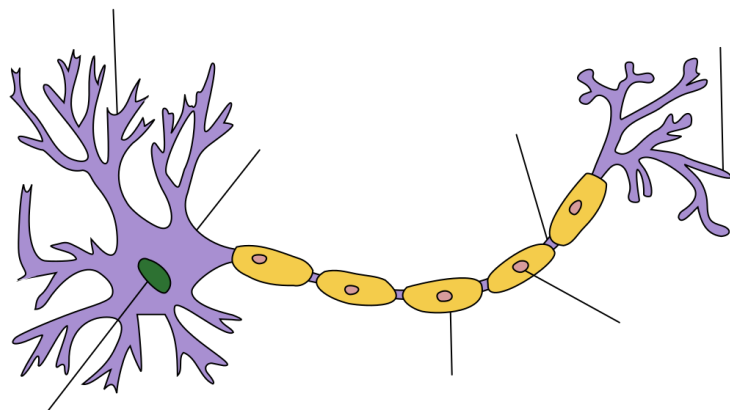
Python Programming

Jian Zhang



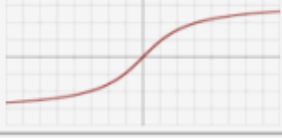


Introduction to Deep Learning

- Fully-Connected Layer
- PyTorch and AutoGrad
- Regression and Classification

Neuron



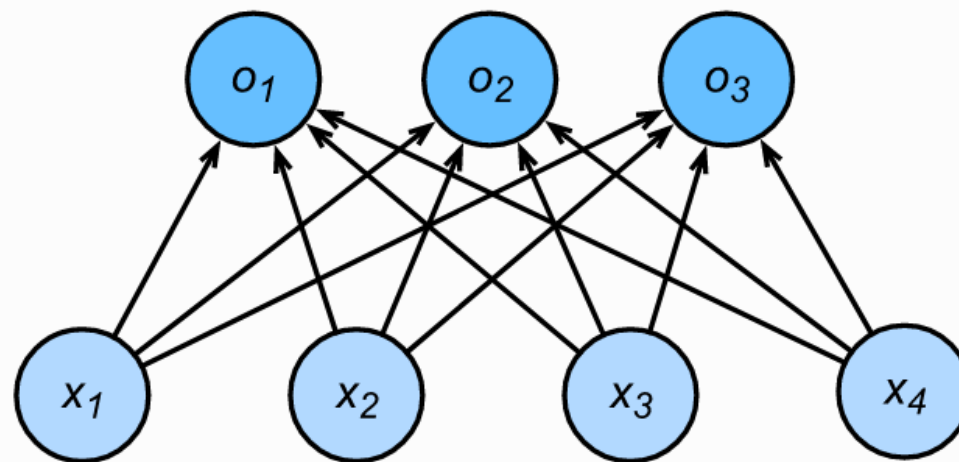
Activation Function

Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU) ^[7]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) ^[8]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

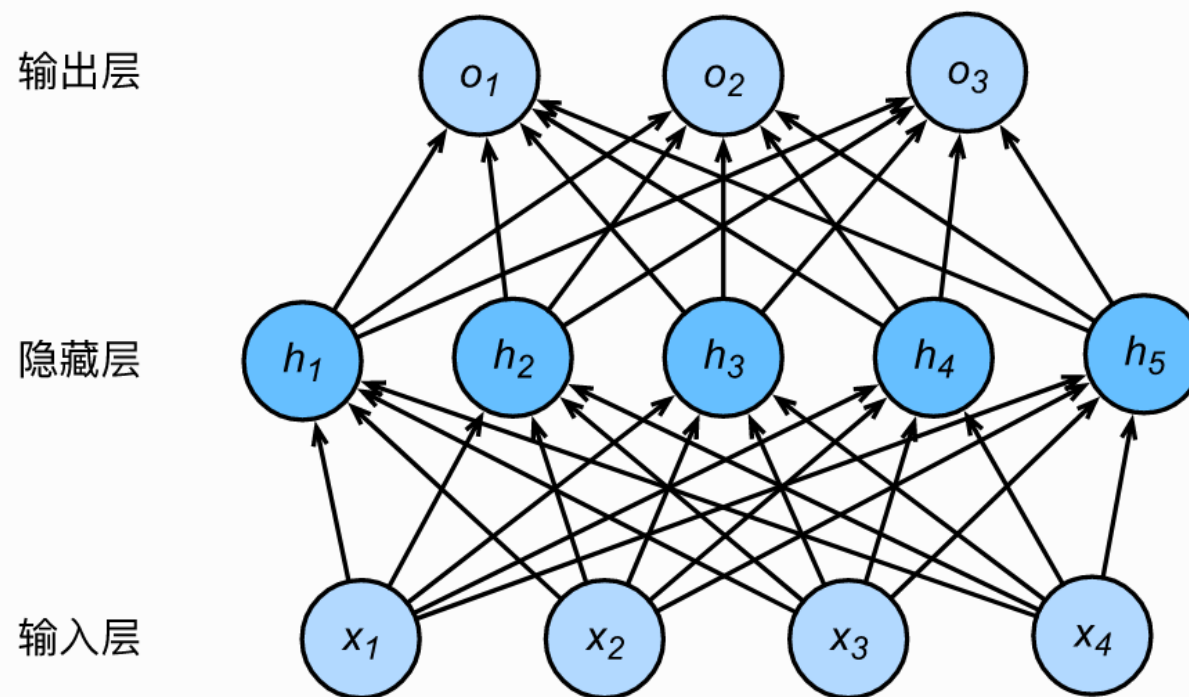
One-layer Fully Connected Network

输出层

输入层

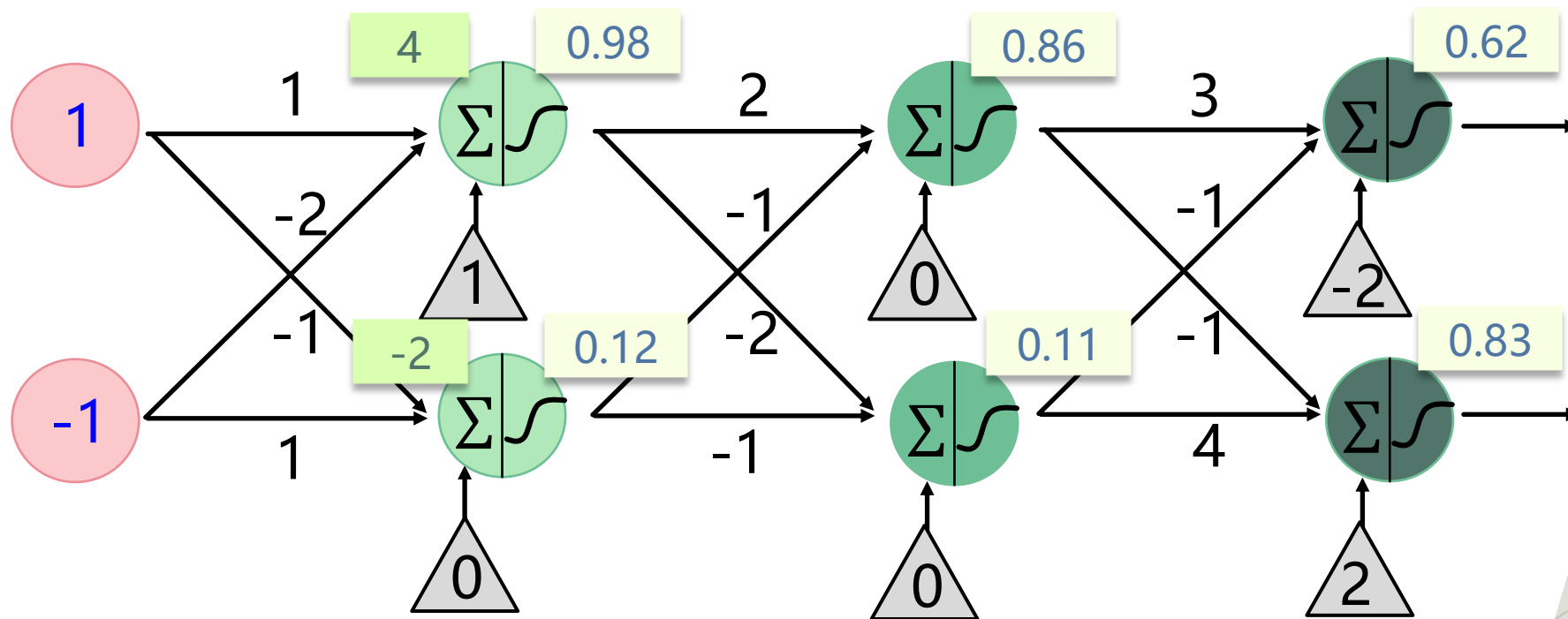


Two-layer Fully Connected Network



$$\begin{aligned} \mathbf{H} &= \phi(\mathbf{XW}_h + \mathbf{b}_h), \\ \mathbf{O} &= \mathbf{HW}_o + \mathbf{b}_o, \end{aligned}$$

Example:



函数: $f\left(\begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix}$

Two-layer Fully Connected Network

DATA

Which dataset do you want to use?



Ratio of training to test data: 50%



Noise: 0



Batch size: 10



REGENERATE

FEATURES

Which properties do you want to feed in?

X_1



X_2



X_1^2



X_2^2



$X_1 X_2$



$\sin(X_1)$



$\sin(X_2)$



+

-

2 HIDDEN LAYERS

+

-

4 neurons

+

-

2 neurons

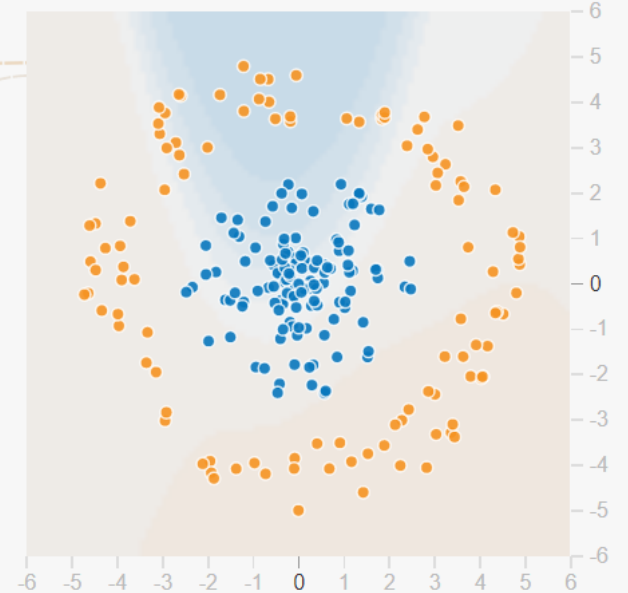
The outputs are mixed with varying **weights**, shown by the thickness of the lines.

This is the output from one **neuron**. Hover to see it larger.

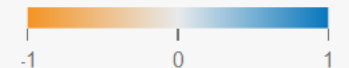
OUTPUT

Test loss 0.472

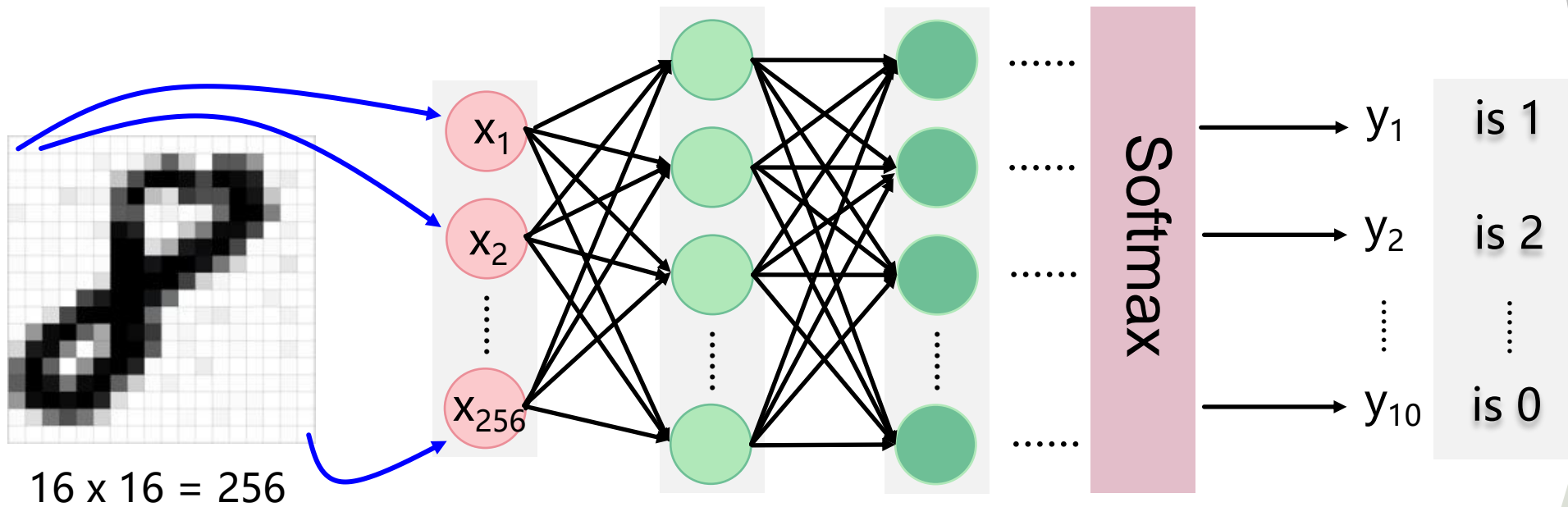
Training loss 0.477



Colors shows data, neuron and weight values.

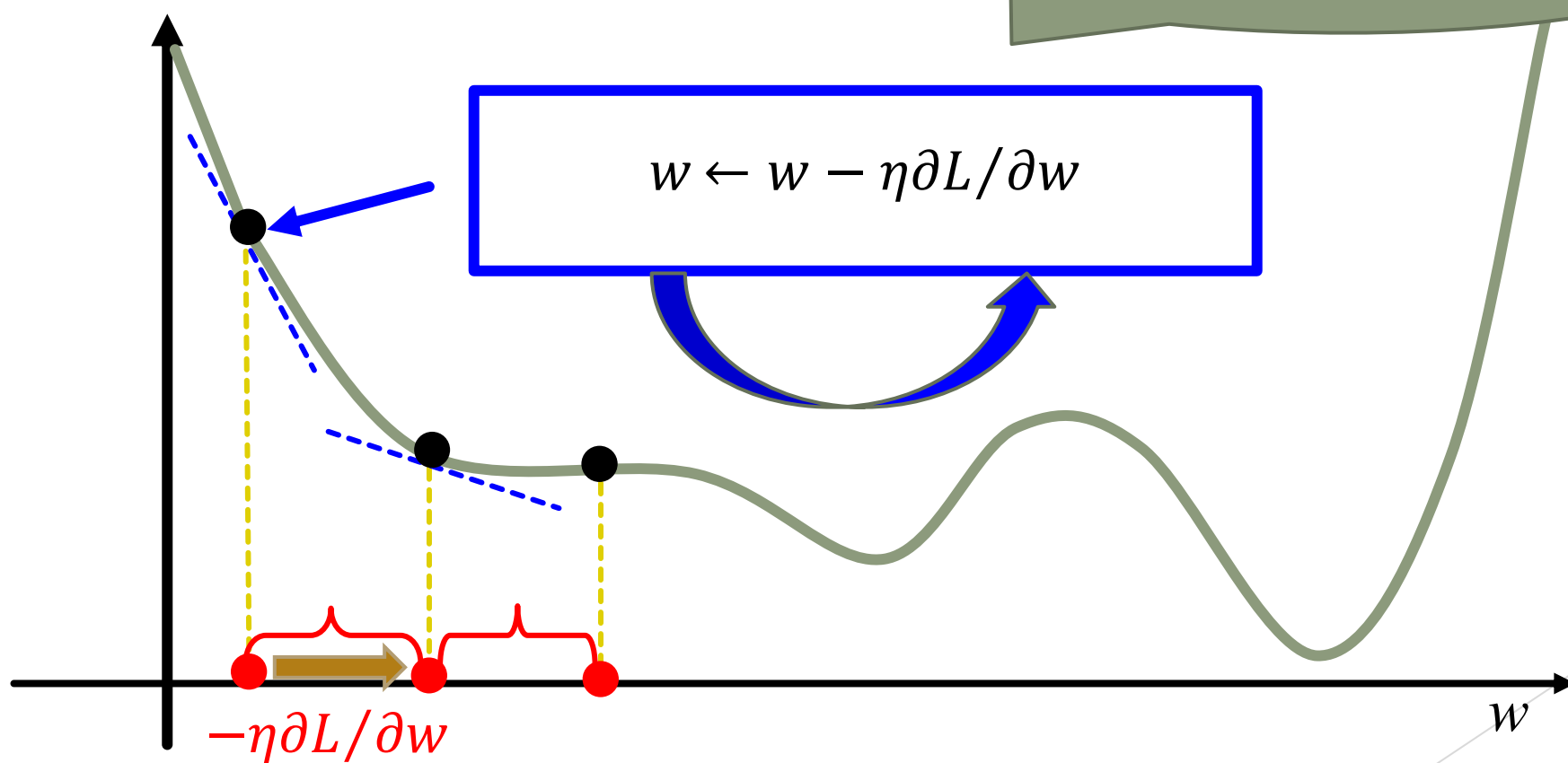


Loss Function

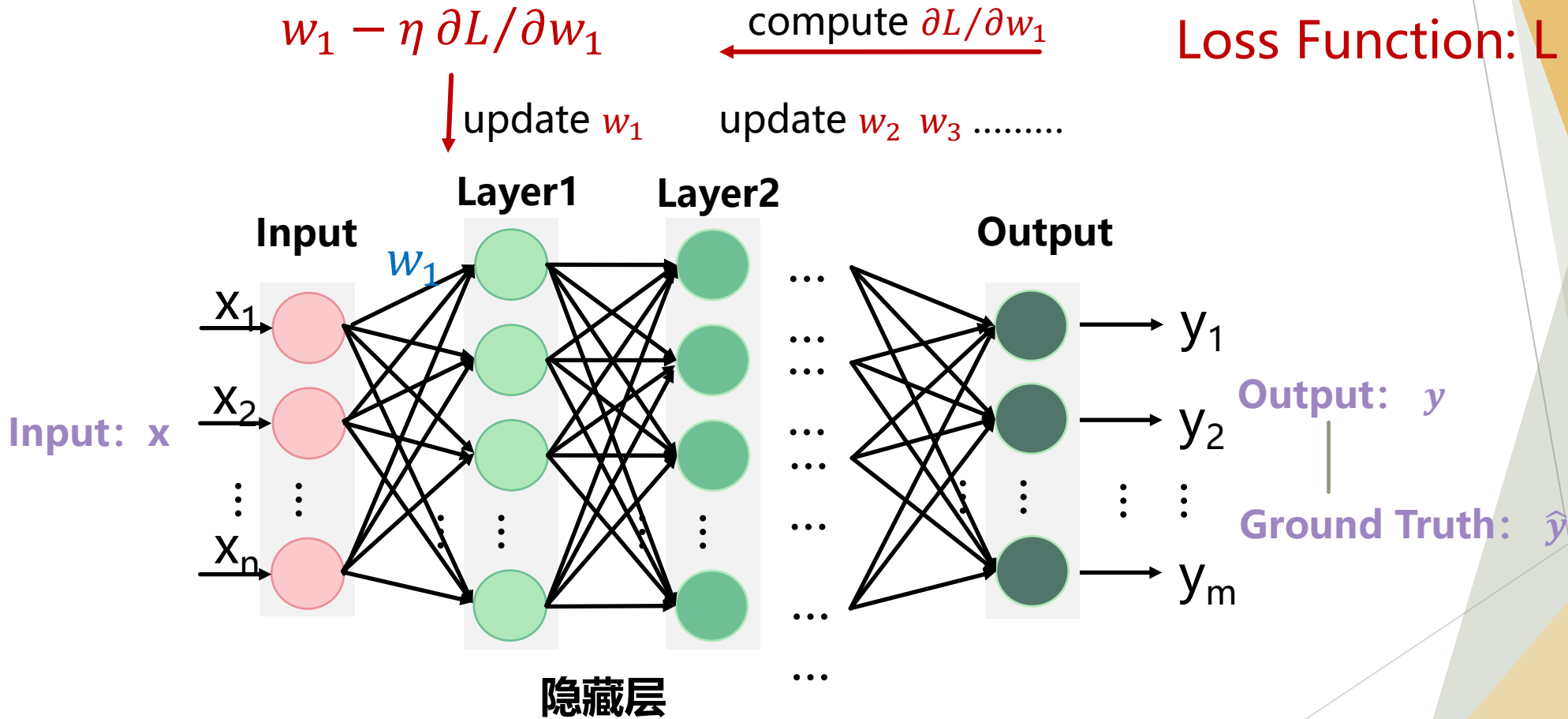


Gradient Descent

$$\theta = \{w_1, w_2, \dots, b_1, b_2, \dots\}$$



Backpropagation



PyTorch

<https://pytorch.org/>

[Get Started](#)[Ecosystem](#) ▾[Mobile](#)[Blog](#)[Tutorials](#)[Docs](#) ▾[Resources](#) ▾[GitHub](#)

Additional support or warranty for some PyTorch Stable and LTS binaries are available through the [PyTorch Enterprise Support Program](#).

PyTorch Build

Stable (1.10.1)

Preview (Nightly)

LTS (1.8.2)

Your OS

Linux

Mac

Windows

Package

Conda

Pip

LibTorch

Source

Language

Python

C++ / Java

Compute Platform

CUDA 10.2

CUDA 11.3

ROCm 4.2 (beta)

CPU

Run this Command:

```
pip3 install torch torchvision torchaudio
```

[Alibaba Cloud](#)[Amazon Web Services](#)[Google Cloud
Platform](#)[Microsoft
Azure - *PyTorch
Enterprise
Program*](#)

Computational Graphs

Numpy

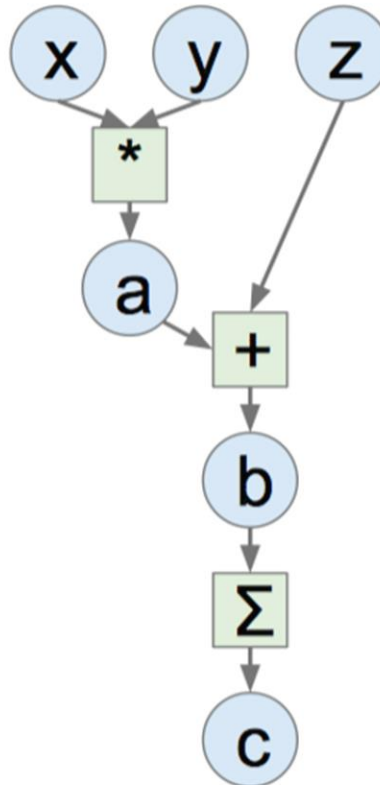
```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```



PyTorch

```
import torch

N, D = 3, 4
x = torch.randn(N, D, requires_grad=True)
y = torch.randn(N, D)
z = torch.randn(N, D)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()
print(x.grad)
```

PyTorch handles gradients for us!

Build Fully Connected Layer by PyTorch

```
class torch.nn.Linear(in_features, out_features, bias=True) \[source\]
```

Applies a linear transformation to the incoming data: $y = xA^T + b$

Parameters:

- `in_features` – size of each input sample
- `out_features` – size of each output sample
- `bias` – If set to `False`, the layer will not learn an additive bias. Default: `True`

Shape:

- Input: $(N, *, in_features)$ where $*$ means any number of additional dimensions
- Output: $(N, *, out_features)$ where all but the last dimension are the same shape as the input.

Variables:

- `weight` – the learnable weights of the module of shape $(out_features \times in_features)$
- `bias` – the learnable bias of the module of shape $(out_features)$

Build Fully Connected Layer by PyTorch

```
import torch.nn as nn
input = torch.randn(10,100) # (BatchSize, length)
fc1 = nn.Linear(100, 200)
output_fc1 = fc1(input)

print("Size of Input is", input.shape)
print("Size of fc1 Output is", output_fc1.shape)

params = list(fc1.parameters())
print("Parameter Number of fc1 is %d " % len(params))

for name, parameters in fc1.named_parameters():
    print(name, ':', parameters.size())
```

```
Size of Input is torch.Size([10, 100])
Size of fc1 Output is torch.Size([10, 200])
The Number of fc1 is 2
weight : torch.Size([200, 100])
bias : torch.Size([200])
```


Build Fully Connected Layer by PyTorch

```
import torch.nn as nn
input = torch.randn(10,100) # (BatchSize, length)
fc1 = nn.Linear(100, 200, bias=False)
output_fc1 = fc1(input)

print("Size of Input is", input.shape)
print("Size of fc1 Output is", output_fc1.shape)

params = list(fc1.parameters())
print("Parameter Number of fc1 is %d " % len(params))

for name, parameters in fc1.named_parameters():
    print(name, ':', parameters.size())
```

```
Size of Input is torch.Size([10, 100])
Size of fc1 Output is torch.Size([10, 200])
The Number of fc1 is 1
weight : torch.Size([200, 100])
```

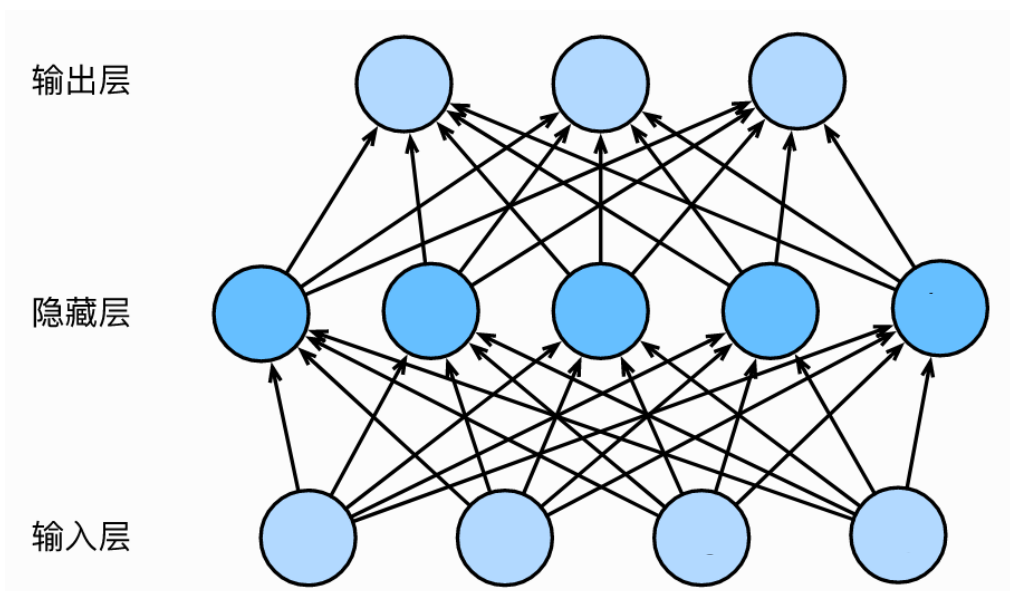

Regression (From Numpy to PyTorch)

$$h = XW_1$$

$$h_{\text{relu}} = \max(0, h)$$

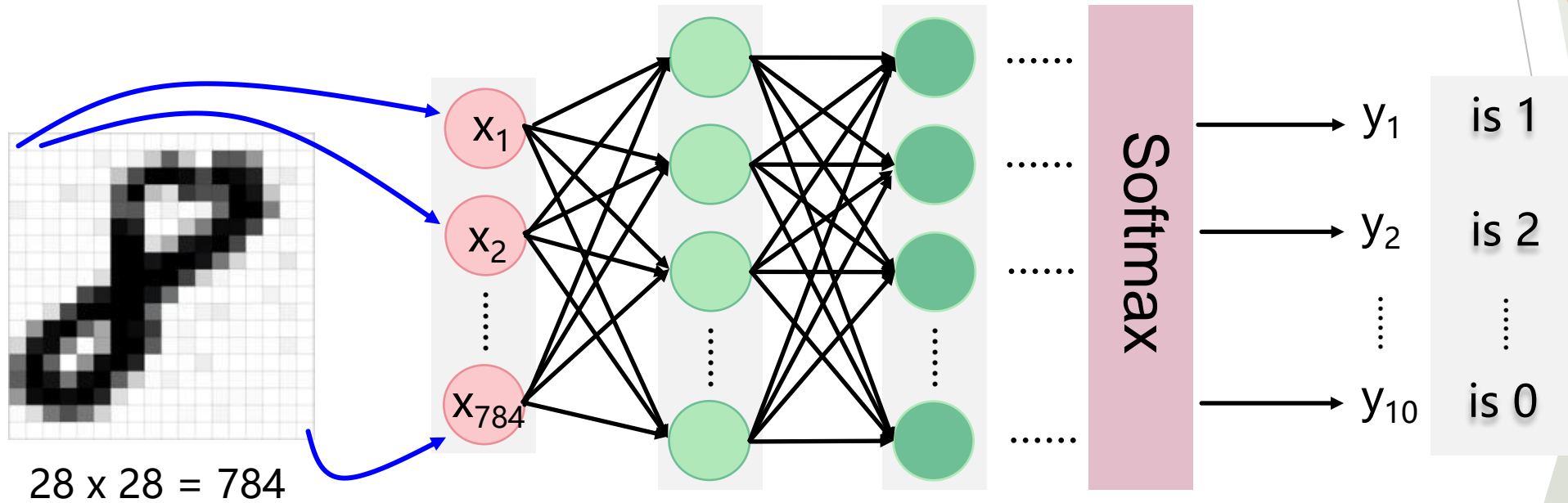
$$Y_{\text{pred}} = h_{\text{relu}} W_2$$

$$f = ||Y - Y_{\text{pred}}||_F^2$$



From_Numpy_To_PyTorch.ipynb

MNIST Classification



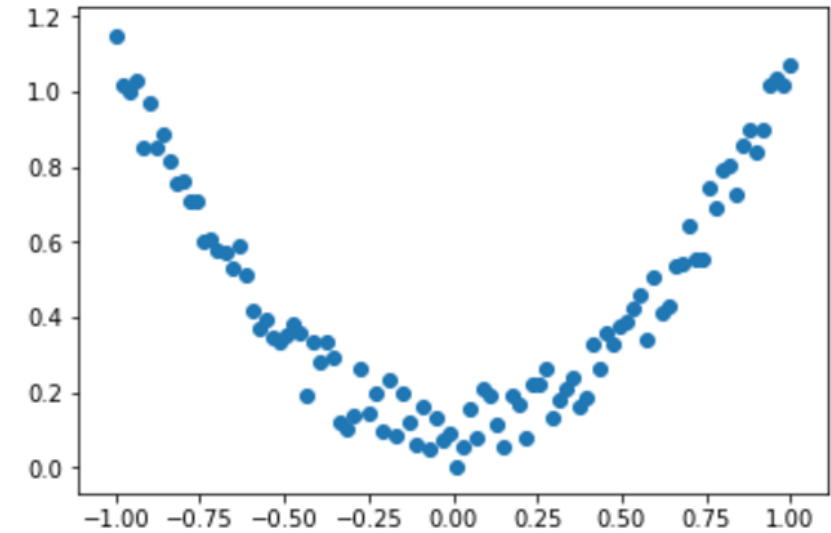
MNIST_FC.ipynb

Homework

Homework.ipynb or Homework.py

```
torch.manual_seed(1)  # reproducible
```

```
x = torch.unsqueeze(torch.linspace(-1, 1, 100), dim=1)
y = x.pow(2) + 0.2*torch.rand(x.size())
```



```
1 class Net(torch.nn.Module):
2     def __init__(self, n_feature, n_hidden, n_output):
3         super(Net, self).__init__()
4         |
5         |
6     def forward(self, x):
7         |
8         return x
```

```
1 net = Net(n_feature=1, n_hidden=20, n_output=1)  # define the network
2 print(net)  # net architecture
```



Questions?