

ML/DL for Everyone with PYTORCH

Lecture 2: Linear Model

Sung Kim <hunkim+ml@gmail.com> HKUST

Code: <https://github.com/hunkim/PyTorchZeroToAll>

Slides: <http://bit.ly/PyTorchZeroAll>

Videos: <http://bit.ly/PyTorchVideo>



Call for Comments

Please feel free to add comments directly on these slides.

Other slides: <http://bit.ly/PyTorchZeroAll>



ML/DL for Everyone with PYTORCH

Lecture 2: Linear Model

Sung Kim <hunkim+ml@gmail.com> HKUST

Code: <https://github.com/hunkim/PyTorchZeroToAll>

Slides: <http://bit.ly/PyTorchZeroAll>

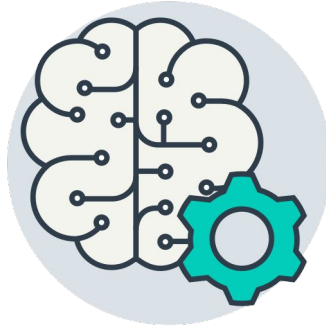
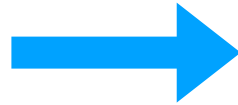
Videos: <http://bit.ly/PyTorchVideo>



Machine Learning

What would be the grade if I study 4 hours?

*4
hours*



?
points
Prediction

| Hours (x) | Points (y) |
|-----------|------------|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | ? |

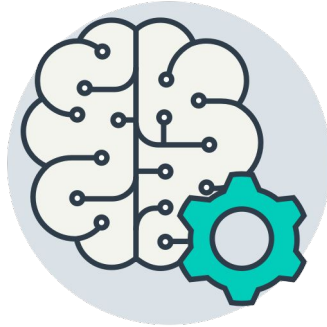
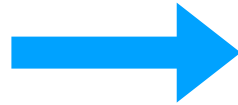
Training dataset

Test dataset

Machine Learning

What would be the grade if I study 4 hours?

*4
hours*



?
points
Prediction

| Hours (x) | Points (y) |
|-----------|------------|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | ? |

Training dataset

Test dataset

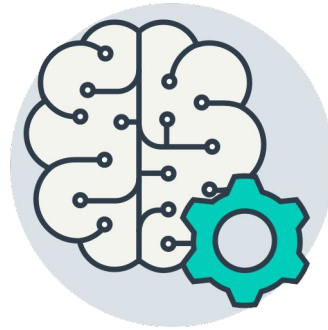


Supervised learning

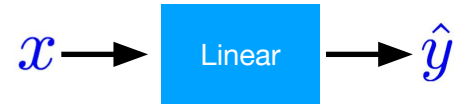
Model design

What would be the best model for the data? Linear?

| Hours (x) | Points (y) |
|-----------|------------|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | ? |



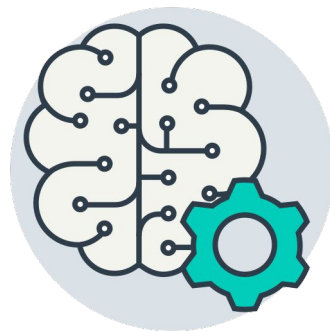
$$\hat{y} = x * w + b$$



Model design

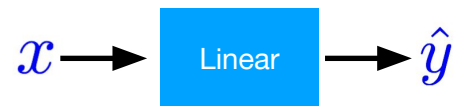
What would be the best model for the data? Linear?

| Hours (x) | Points (y) |
|-----------|------------|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | ? |



$$\hat{y} = x * w$$

$$\hat{y} = x * w + b$$

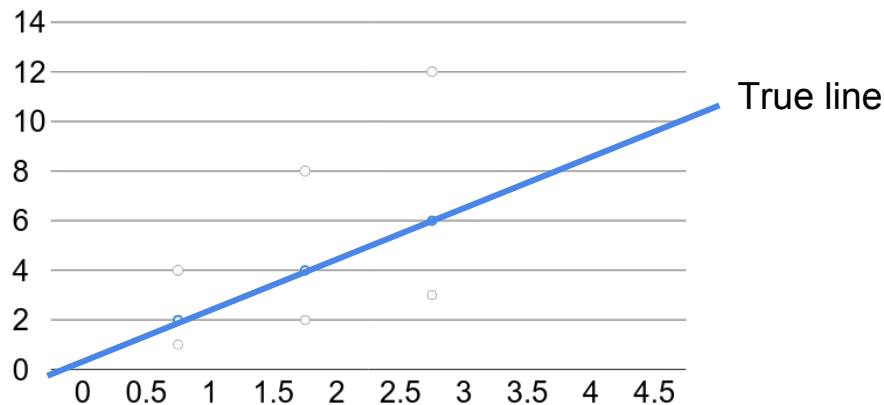


Linear Regression

$$\hat{y} = x * w$$

* The machine starts with **a random guess**, w =random value

| Hours (x) | Points (y) |
|-----------|------------|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| | |

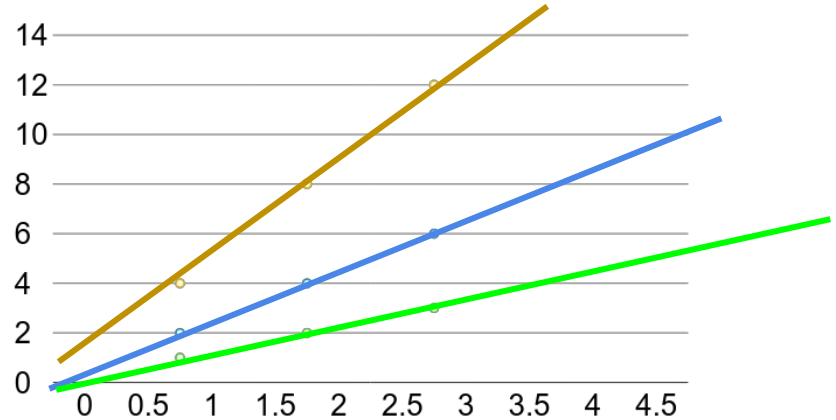


Linear Regression error?

$$\hat{y} = x * w$$

* The machine starts with a **random guess**, w =random value

| Hours (x) | Points (y) |
|-----------|------------|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| | |



Training Loss (error)

$$loss = (\hat{y} - y)^2 = (x * w - y)^2$$

| Hours, x | Points, y | Prediction, $\hat{y}(w=3)$ | Loss (w=3) |
|----------|-----------|----------------------------|------------|
| 1 | 2 | 3 | 1 |
| 2 | 4 | 6 | 4 |
| 3 | 6 | 9 | 9 |
| | | | mean=14/3 |

Training Loss (error)

$$loss = (\hat{y} - y)^2 = (x * w - y)^2$$

| Hours, x | Points, y | Prediction, $\hat{y}^{(w=4)}$ | Loss (w=4) |
|----------|-----------|-------------------------------|------------|
| 1 | 2 | 4 | 4 |
| 2 | 4 | 8 | 16 |
| 3 | 6 | 12 | 36 |
| | | | mean=56/3 |

Training Loss (error)

$$loss = (\hat{y} - y)^2 = (x * w - y)^2$$

| Hours, x | Points, y | Prediction, $\hat{y}(w=0)$ | Loss (w=0) |
|----------|-----------|----------------------------|------------|
| 1 | 2 | 0 | 4 |
| 2 | 4 | 0 | 16 |
| 3 | 6 | 0 | 36 |
| | | | mean=56/3 |

Training Loss (error)

$$loss = (\hat{y} - y)^2 = (x * w - y)^2$$

| Hours, x | Points, y | Prediction, $\hat{y}^{(w=1)}$ | Loss (w=1) |
|----------|-----------|-------------------------------|------------|
| 1 | 2 | 1 | 1 |
| 2 | 4 | 2 | 4 |
| 3 | 6 | 3 | 9 |
| | | | mean=14/3 |

Training Loss (error)

$$loss = (\hat{y} - y)^2 = (x * w - y)^2$$

| Hours, x | Points, y | Prediction, $\hat{y}^{(w=2)}$ | Loss (w=2) |
|----------|-----------|-------------------------------|------------|
| 1 | 2 | 2 | 0 |
| 2 | 4 | 4 | 0 |
| 3 | 6 | 6 | 0 |
| | | | mean=0 |

Training Loss (error)

$$loss = (\hat{y} - y)^2 = (x * w - y)^2 \quad loss = \frac{1}{N} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

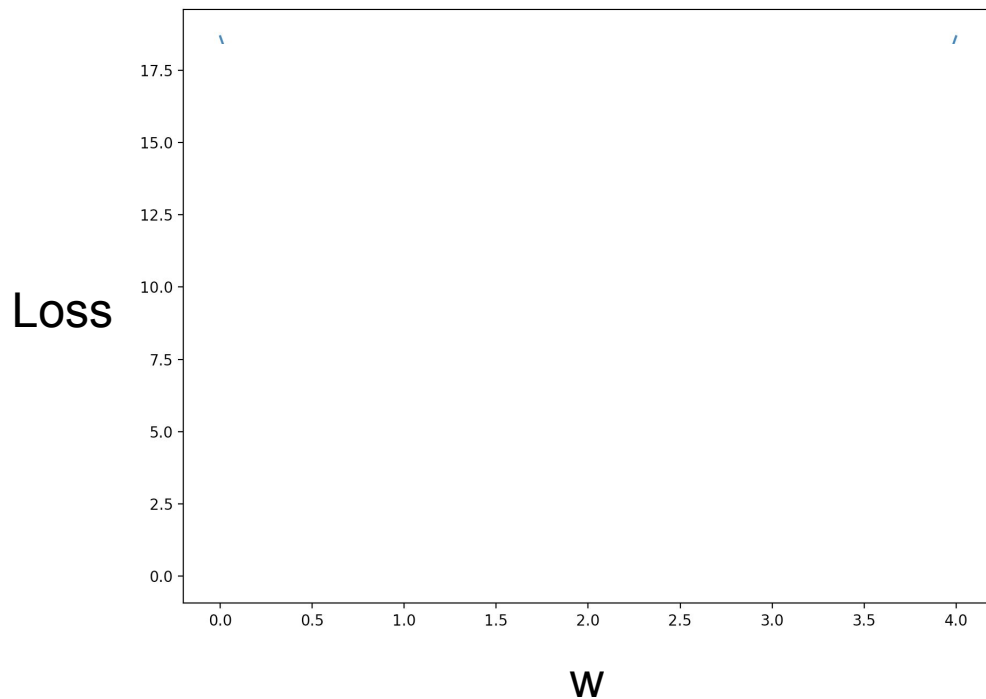
MSE, mean square error

| Hours, x | Loss (w=0) | Loss (w=1) | Loss (w=2) | Loss (w=3) | Loss (w=4) |
|----------|---------------|--------------|------------|--------------|---------------|
| 1 | 4 | 1 | 0 | 1 | 4 |
| 2 | 16 | 4 | 0 | 4 | 16 |
| 3 | 36 | 9 | 0 | 9 | 36 |
| | MSE=56/3=18.7 | MSE=14/3=4.7 | MSE=0 | MSE=14/3=4.7 | MSE=56/3=18.7 |

Loss graph

$$loss = \frac{1}{N} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

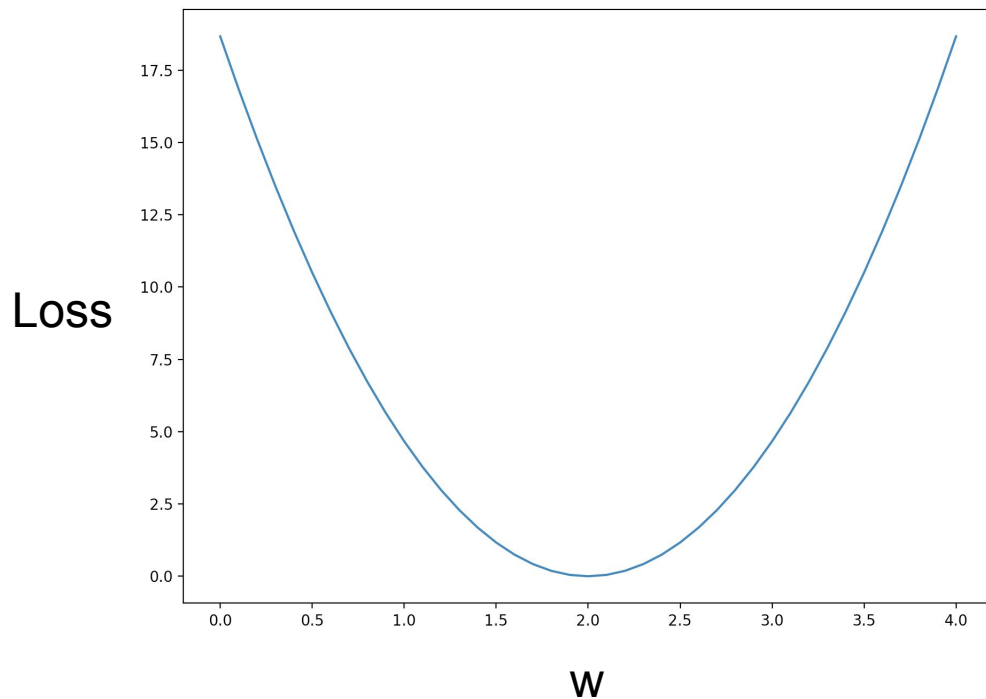
| Loss (w=0) | Loss (w=1) | Loss (w=2) | Loss (w=3) | Loss (w=4) |
|----------------|---------------|------------|---------------|----------------|
| mean=56/3=18.7 | mean=14/3=4.7 | mean=0 | mean=14/3=4.7 | mean=56/3=18.7 |



Loss graph

$$loss = \frac{1}{N} \sum_{n=1}^N (\hat{y}_n - y_n)^2$$

| Loss (w=0) | Loss (w=1) | Loss (w=2) | Loss (w=3) | Loss (w=4) |
|----------------|---------------|------------|---------------|----------------|
| mean=56/3=18.7 | mean=14/3=4.7 | mean=0 | mean=14/3=4.7 | mean=56/3=18.7 |



Model & Loss



$$\hat{y} = x * w$$

```
w = 1.0 # a random guess: random value
```

```
# our model for the forward pass
```

```
def forward(x):  
    return x * w
```

$$loss = (\hat{y} - y)^2$$

```
# Loss function
```

```
def loss(x, y):  
    y_pred = forward(x)  
    return (y_pred - y) * (y_pred - y)
```

Compute loss for w



```
for w in np.arange(0.0, 4.1, 0.1):  
    print("w=", w)  
    l_sum = 0  
    for x_val, y_val in zip(x_data, y_data):  
        y_pred_val = forward(x_val)  
        l = loss(x_val, y_val)  
        l_sum += l  
    print("\t", x_val, y_val, y_pred_val, l)  
  
print("MSE=", l_sum / 3)
```

```
w= 0.0  
    1.0 2.0 0.0 4.0  
    2.0 4.0 0.0 16.0  
    3.0 6.0 0.0 36.0  
NSE= 18.6666666667  
w= 0.1  
    1.0 2.0 0.1 3.61  
    2.0 4.0 0.2 14.44  
    3.0 6.0 0.3 32.49  
NSE= 16.8466666667  
w= 0.2  
    1.0 2.0 0.2 3.24  
    2.0 4.0 0.4 12.96  
    3.0 6.0 0.6 29.16  
NSE= 15.12  
w= 0.3  
    1.0 2.0 0.3 2.89  
    2.0 4.0 0.6 11.56  
    3.0 6.0 0.9 26.01  
NSE= 13.4866666667  
w= 0.4  
    1.0 2.0 0.4 2.56  
    2.0 4.0 0.8 10.24  
    3.0 6.0 1.2 23.04  
NSE= 11.9466666667  
w= 0.5  
    1.0 2.0 0.5 2.25  
    2.0 4.0 1.0 9.0  
    3.0 6.0 1.5 20.25  
NSE= 10.5
```

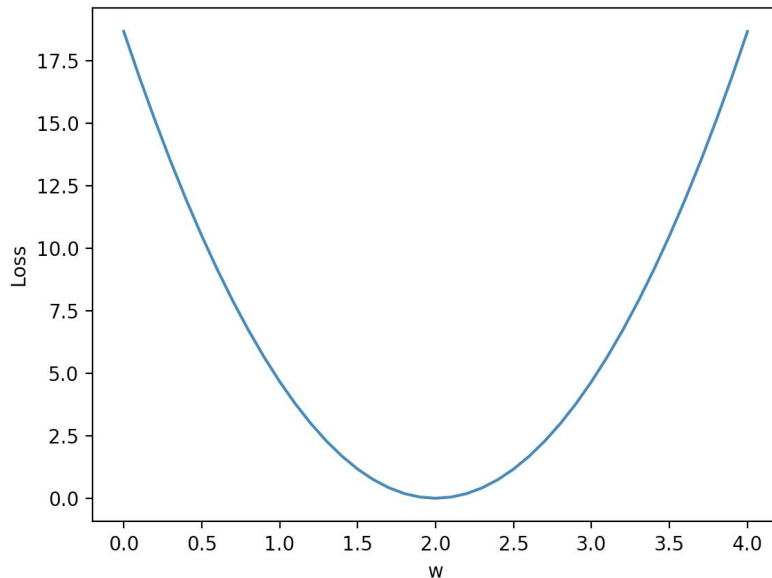
Plot graph



```
w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print("w=", w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        l = loss(x_val, y_val)
        l_sum += l
    print("\t", x_val, y_val, y_pred_val, l)

    print("MSE=", l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)

plt.plot(w_list, mse_list)
plt.ylabel('Loss')
plt.xlabel('w')
plt.show()
```



```

import numpy as np
import matplotlib.pyplot as plt
x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

w = 1.0 # a random guess: random value, 1.0

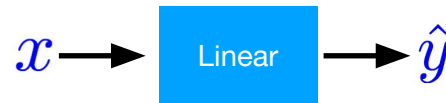
# our model for the forward pass
def forward(x):
    return x * w

# Loss function
def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print("w=", w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        l = loss(x_val, y_val)
        l_sum += l
    print("\t", x_val, y_val, y_pred_val, l)
    print("MSE=", l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)

plt.plot(w_list, mse_list)
plt.ylabel('Loss')
plt.xlabel('w')
plt.show()

```



Exercise 2-1

- Any other interesting linear prediction problems?
- Find some datasets for linear prediction
 - Draw the cost graph for one dataset



Lecture 3: Gradient Descent