SUMMER SCHOOL 2020

MACHINE LEARNING DAY 2

DEEP LEARNING

Session II: Linear regression

Isaac Ye, HPTC @ York University

Isaac@sharcnet.ca

Session II

- Categories of ML problems
- Linear regression (single variable)
- Cost function / gradient decent algorithm / learning rate
- *Lab 2A: linear regression (single variable)*
- Linear regression (multi-variables)
- PyTorch model/cost function/optimizer
- *Lab 2B: linear regression*

Categories of ML problems

	Supervised	Unsupervised	Reinforcement
Discrete	Classification	Clustering	Action space agent
Continuous	Regression	Dimensionality reduction	Action space agent

Categories of ML problems

	Supervised	Unsupervised	Reinforcement
Discrete	Classification	Clustering	Action space agent
Continuous	Regression	Dimensionality reduction	Action space agent

Reinforcement learning

Discrete

Action space agent

Continuous

Action space agent

Reinforcement

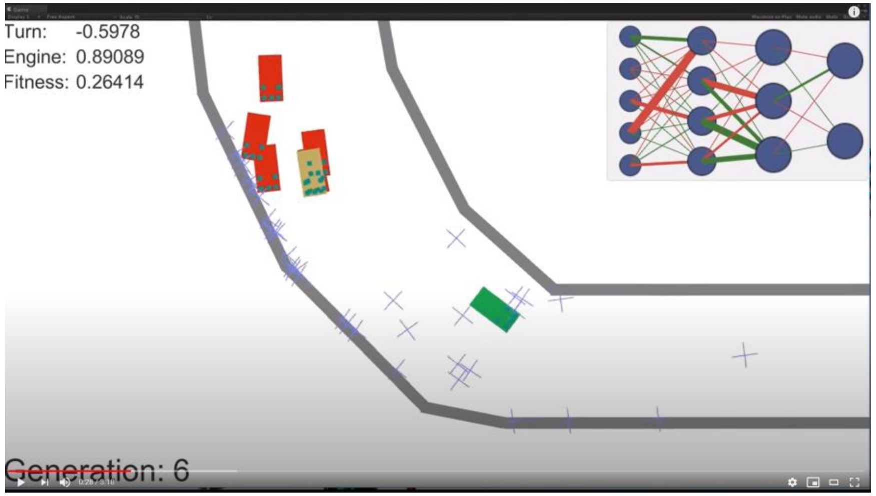
Goal	Train an agent to achieve a goal through state/action/reward
Examples	Path finder, decision making problem

Reinforcement learning

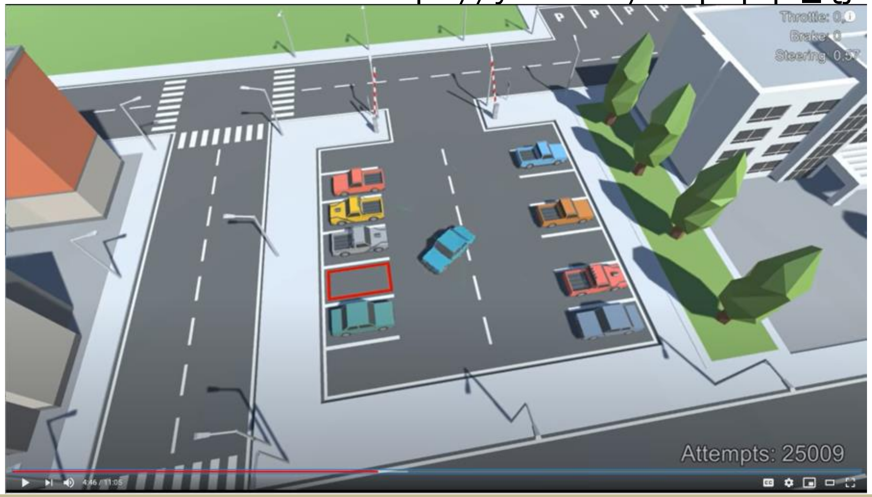
Reinforcement

Goal	Train an agent to achieve a goal through state/action/reward
Examples	Path finder, decision making problem

https://youtu.be/Aut32pR5PQA



https://youtu.be/VMp6pq6_QjI



Categories of ML problems

	Supervised	Unsupervised	Reinforcement
Discrete	Classification	Clustering	Action space agent
Continuous	Regression	Dimensionality reduction	Action space agent

Unsupervised learning

Discrete	**Clustering**
Continuous	**Dimensionality reduction**

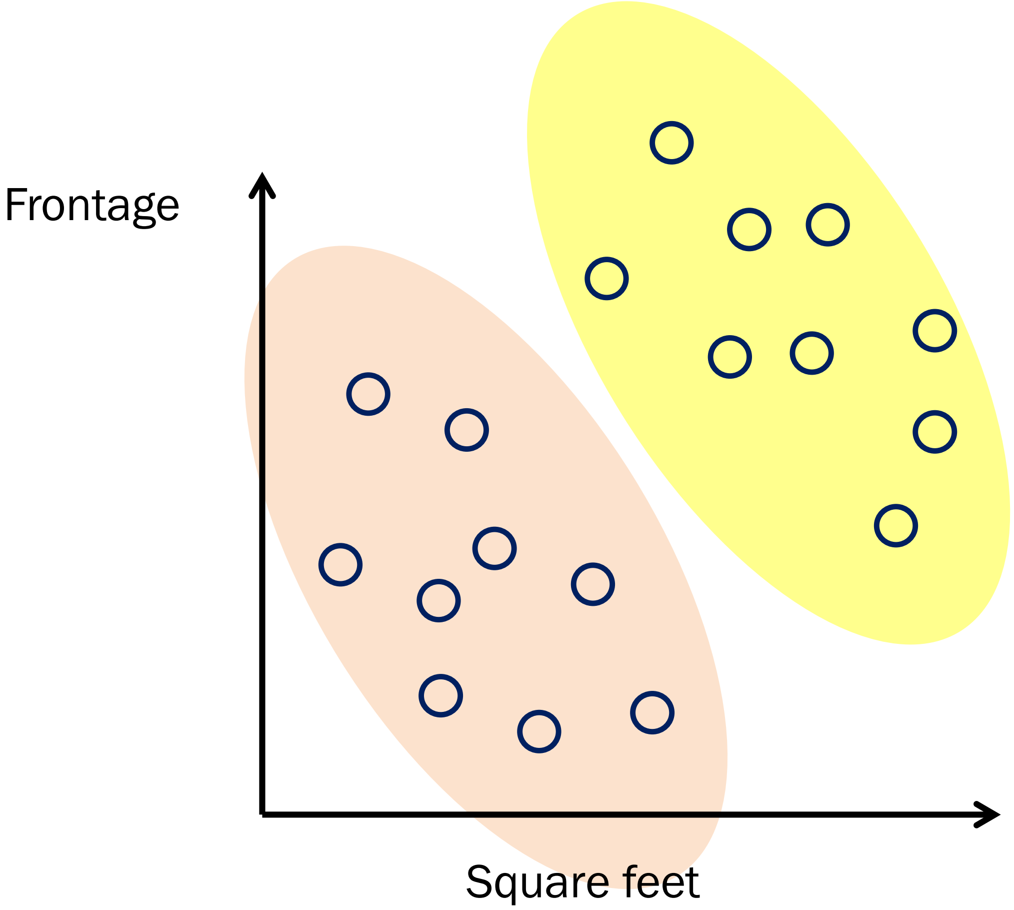
Unsupervised

Input	Data x
Goal	learn some underlying hidden structure of the data
Examples	Clustering, dimensionality reduction, feature learning, density estimation

Unsupervised learning

Unsupervised

Input	Data x
Goal	learn some underlying hidden structure of the data
Examples	Clustering, dimensionality reduction, feature learning, density estimation



Categories of ML problems

	Supervised	Unsupervised	Reinforcement
Discrete	Classification	Clustering	Action space agent
Continuous	Regression	Dimensionality reduction	Action space agent

Supervised learning

Discrete	Classification
Continuous	Regression

Supervised

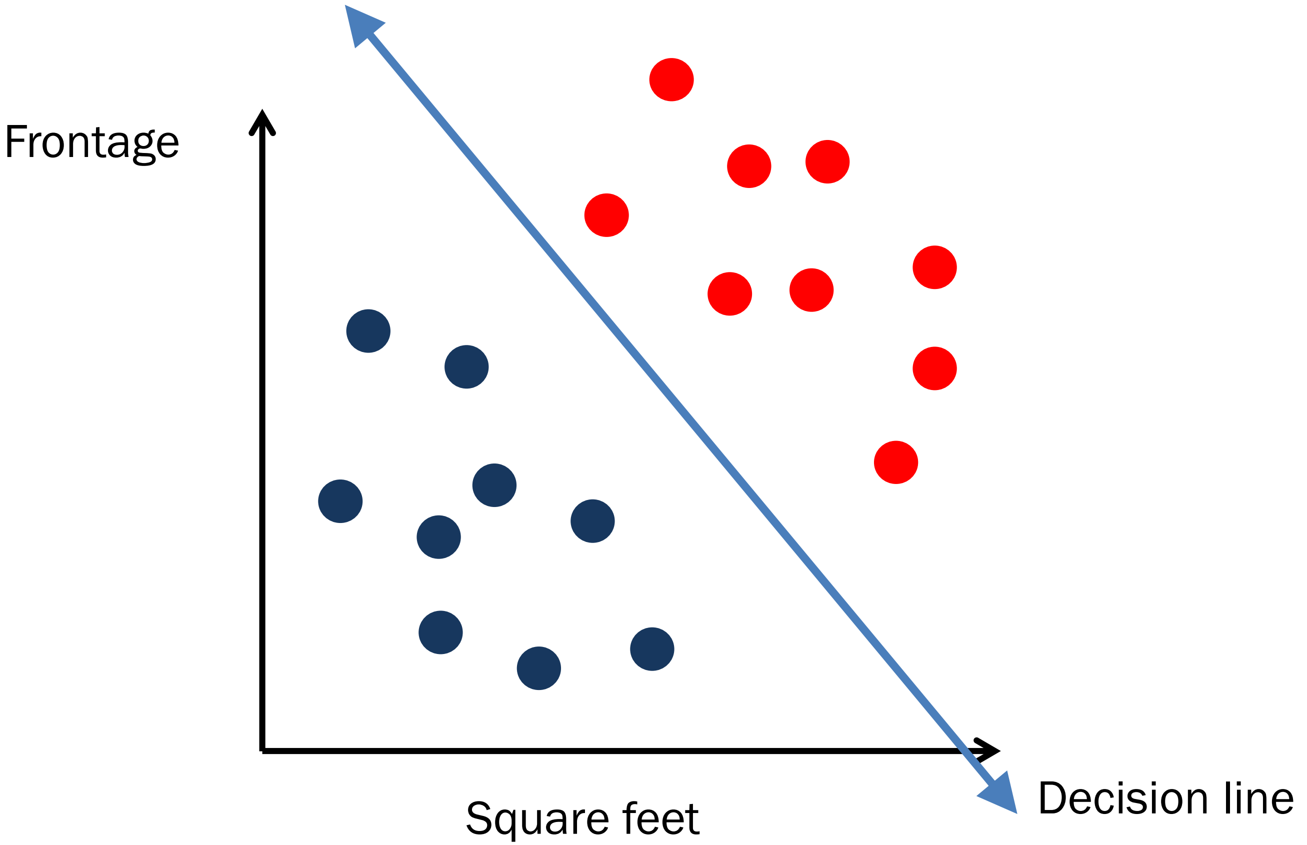
Input	Data (x, y) x: data, y: label
Goal	learn a function to map x to y
Examples	classification, regression, object detection semantic segmentation, image captioning

Supervised learning

Supervised

Input	Data (x, y) x: data, y: label
Goal	learn a function to map x to y
Examples	classification, regression, object detection semantic segmentation, image captioning

Label: Red(> $1M), Blue(< $1M)

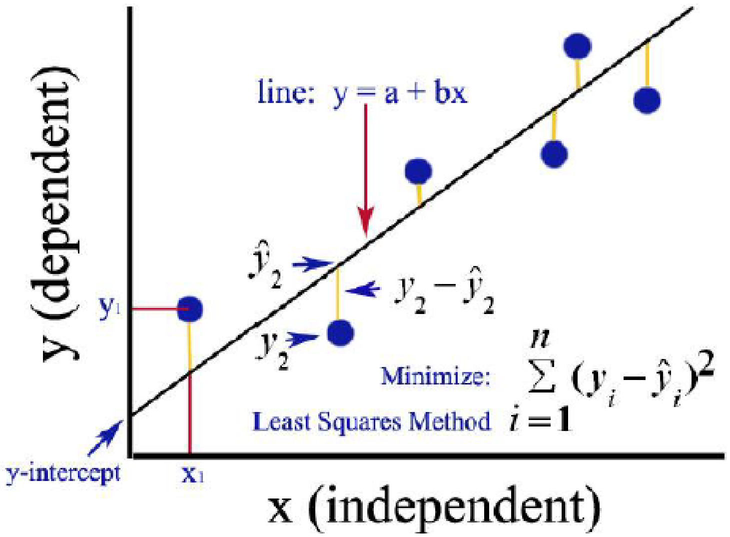


Categories of ML problems

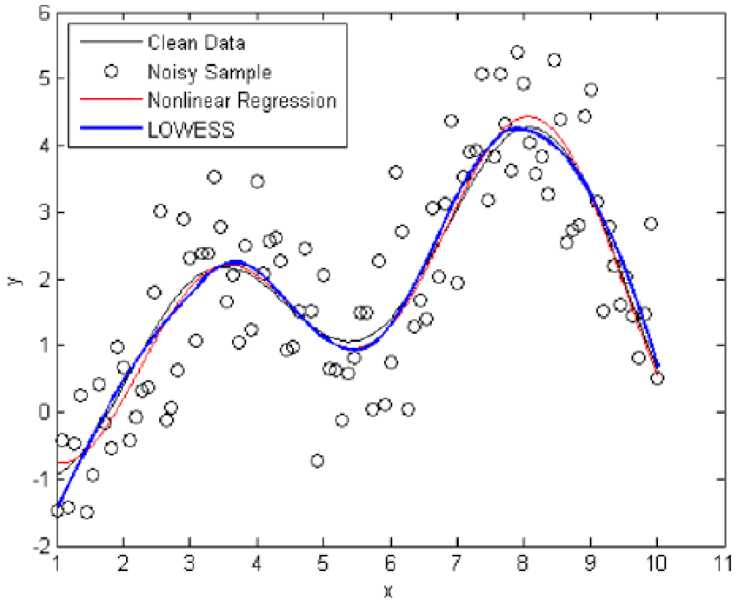
	Supervised	Unsupervised	Reinforcement
Discrete	Classification	Clustering	Action space agent
Continuous	Regression	Dimensionality reduction	Action space agent

Regression problem

Fit the prediction function f(x) to the training data to predict continuous real value



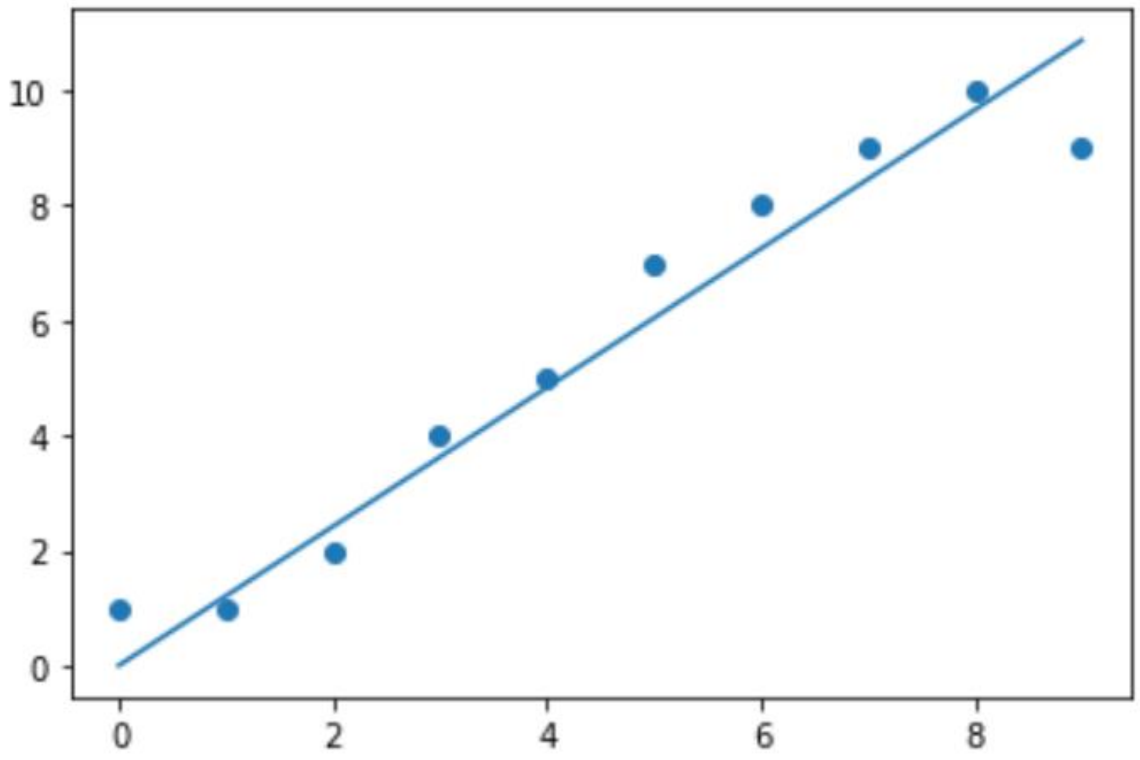
Linear regression



Nonlinear regression

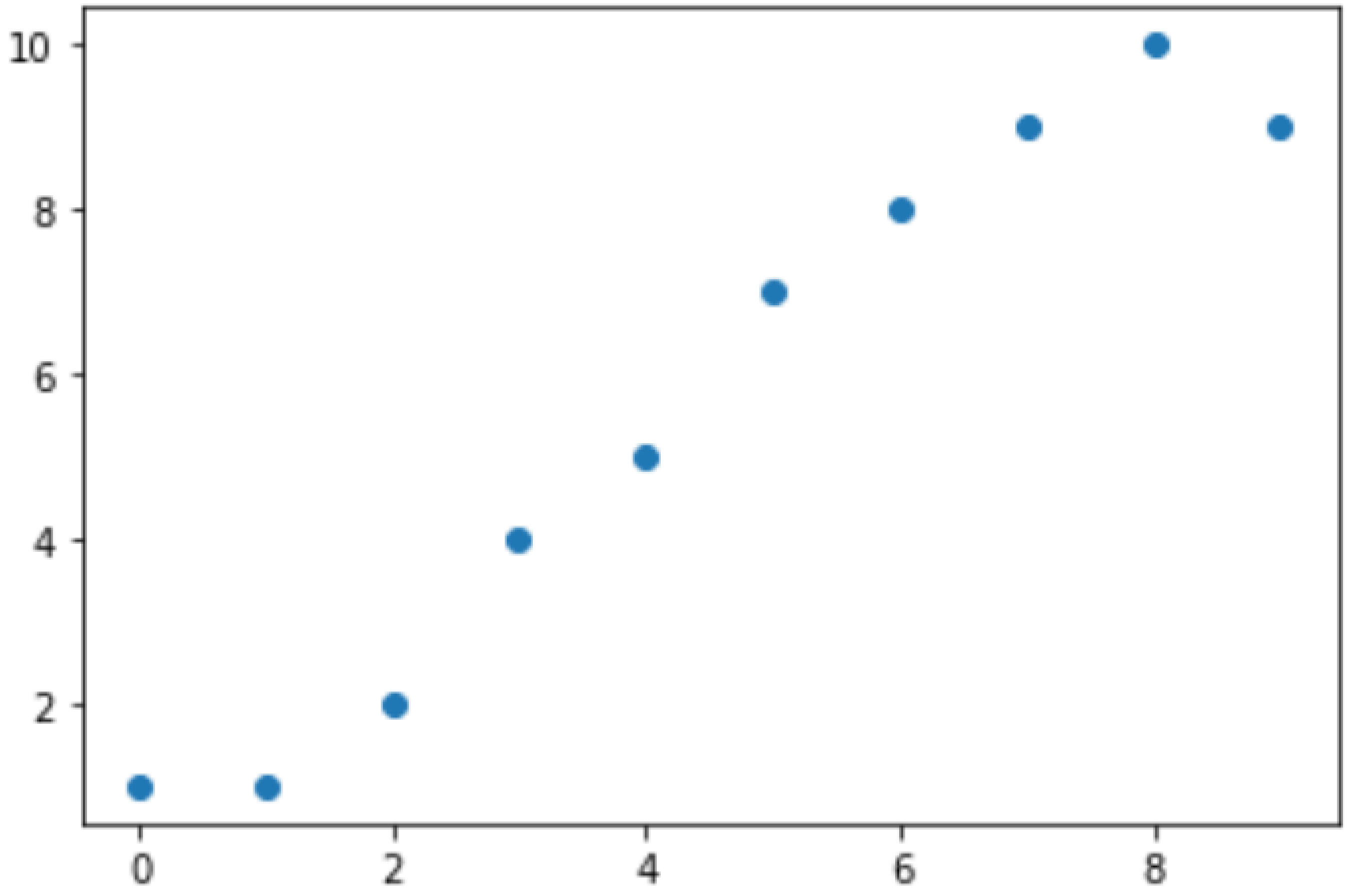
Linear regression : single variable

x	y
0	1
1	1
2	2
3	4
4	5
5	7
6	8
7	9
8	10
9	9

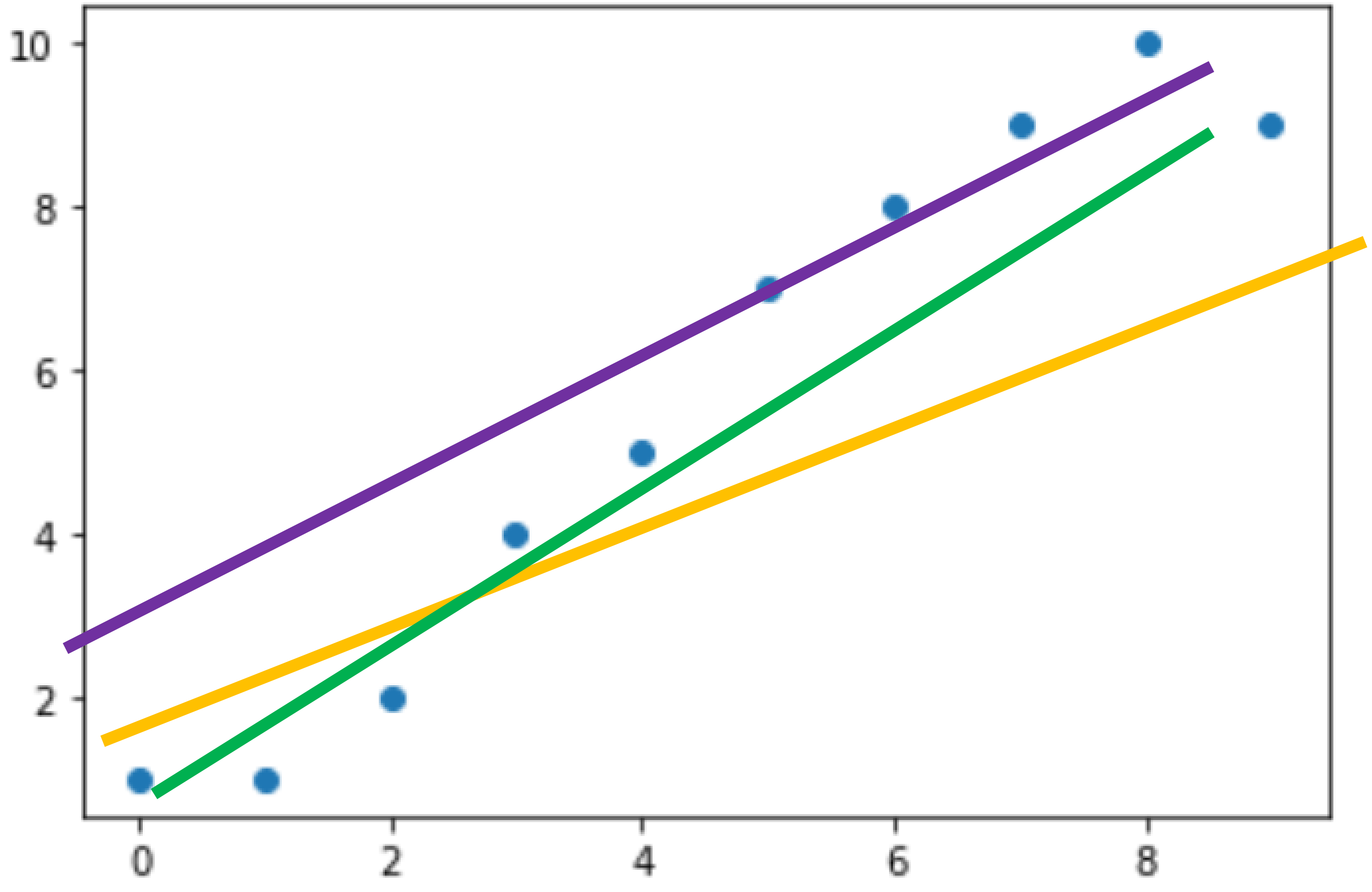


Data preparation: Input (x, y)

x	y
0	1
1	1
2	2
3	4
4	5
5	7
6	8
7	9
8	10
9	9

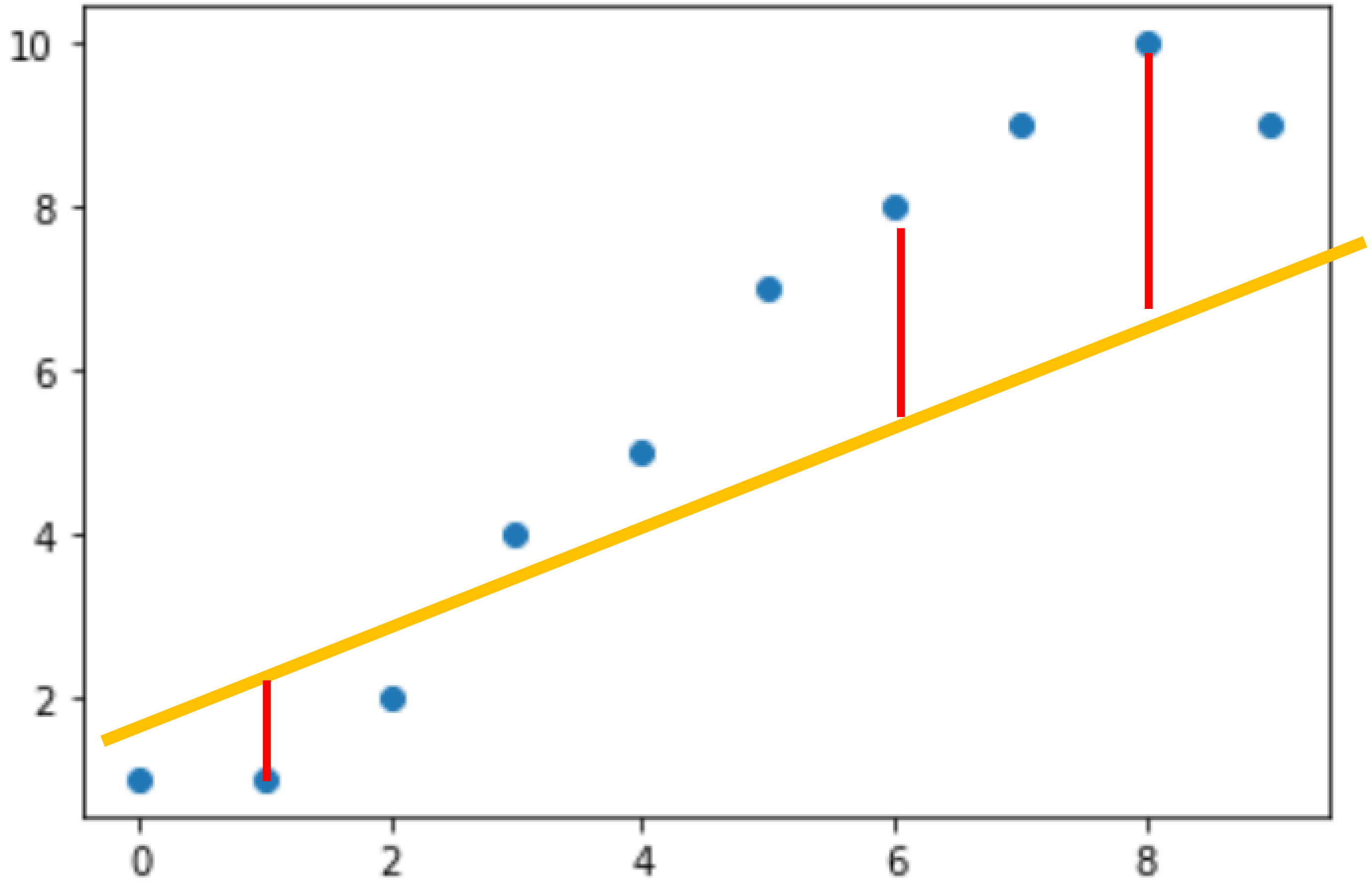


Model (Hypothesis)



$$H(x) = Wx + b$$

Which model is better?



How well fit the line to data?

$$\underbrace{H(x)}_{\text{Predicted}} - \underbrace{y}_{\text{True}}$$

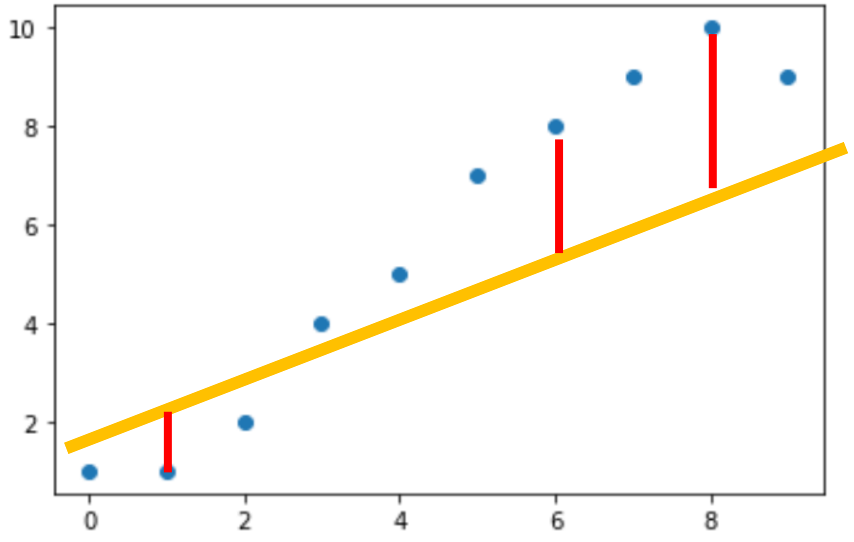
Cost function

<u>Model</u>

$$H(x) = Wx + b$$

<u>Mean Square Error</u>

$$cost = \frac{1}{m}\sum_{i=1}^{m}(H(x_i) - y_i)^2$$



m is the number of data.

Now we can see the <u>cost function</u> as a function of W and b.

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}\left((Wx_i + b) - y_i\right)^2$$

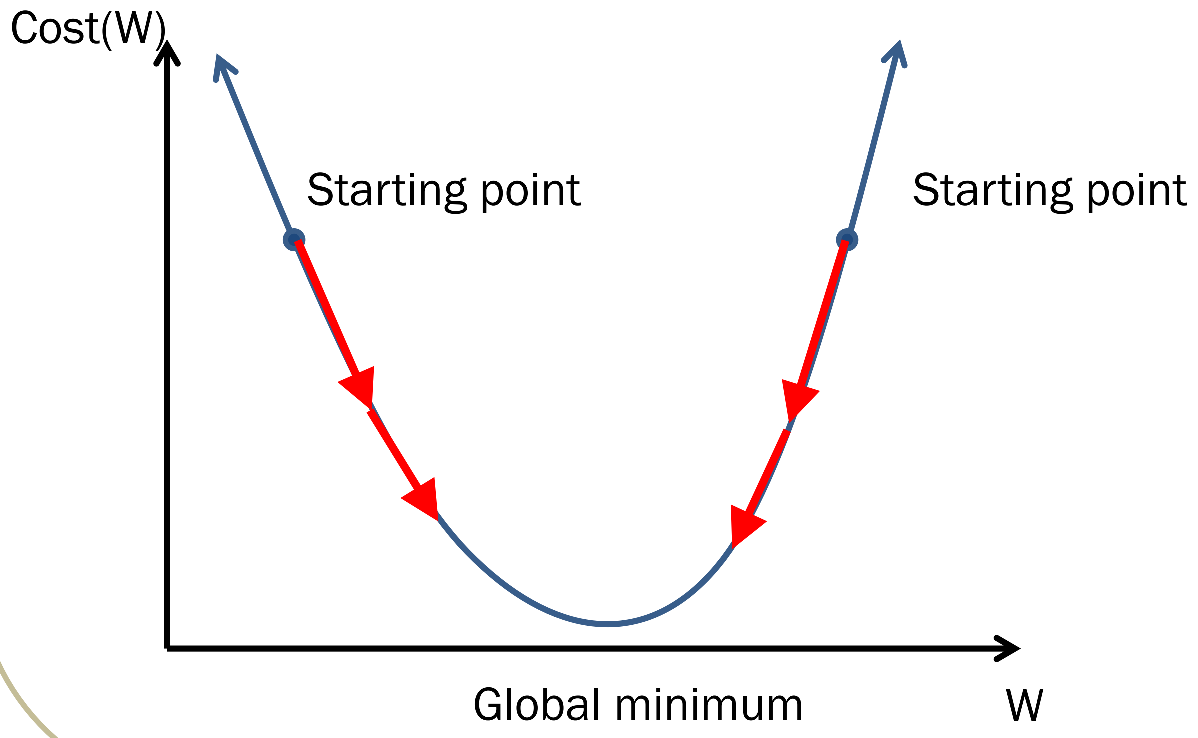
Cost function: what we want?

We want to minimize the cost!

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}\left((Wx_i + b) - y_i\right)^2$$

Gradient decent algorithm

Let's consider a simple case with W only.



$$cost(W) = \frac{1}{2m}\sum_{i=1}^{m}(Wx_i - y_i)^2$$

$$\frac{d}{dW}cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx_i - y_i)x_i$$

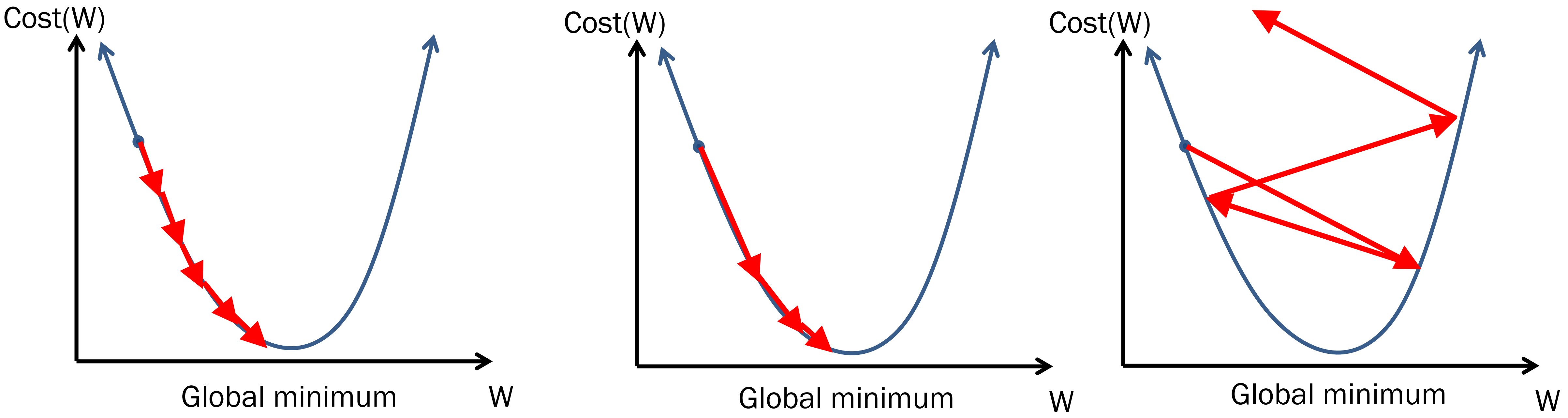
$$W := W - \alpha\frac{d}{dW}cost(W)$$

Learning rate

Learning rate

$$W := W - \alpha \frac{d}{dW} cost(W)$$

Learning rate



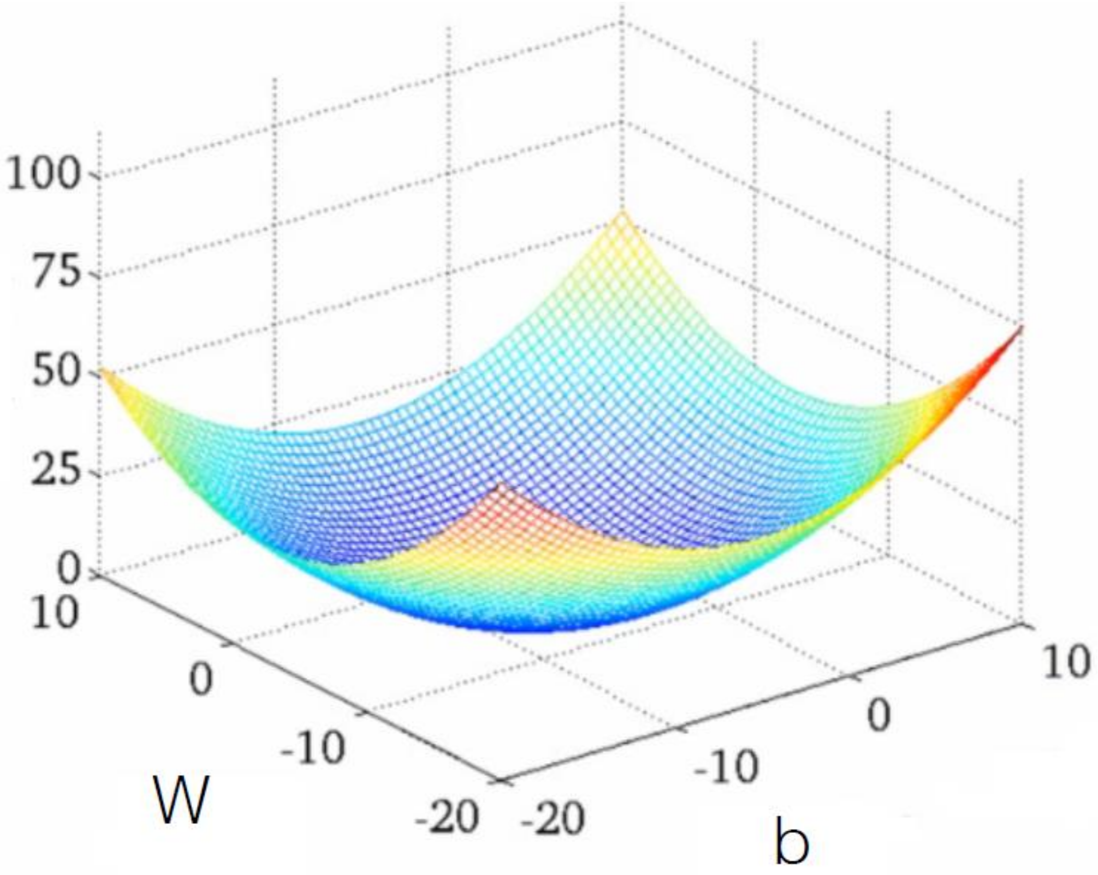
A small learning rate requires many steps

The optimal learning rate (possibly adaptive value)

Too large learning rate causes divergence

Cost (loss) function

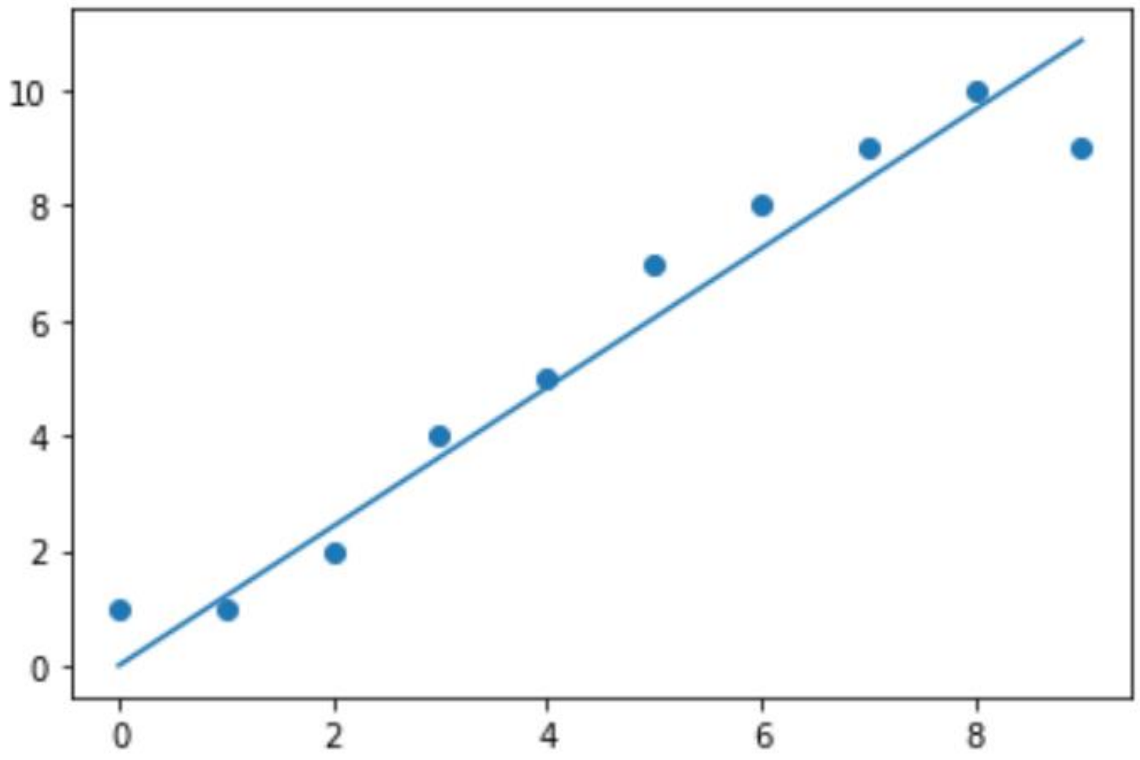
$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}\left((Wx_i + b) - y_i\right)^2$$



http://www.holehouse.org/mlclass

Lab 2A: Linear regression (single variable)

x	y
0	1
1	1
2	2
3	4
4	5
5	7
6	8
7	9
8	10
9	9

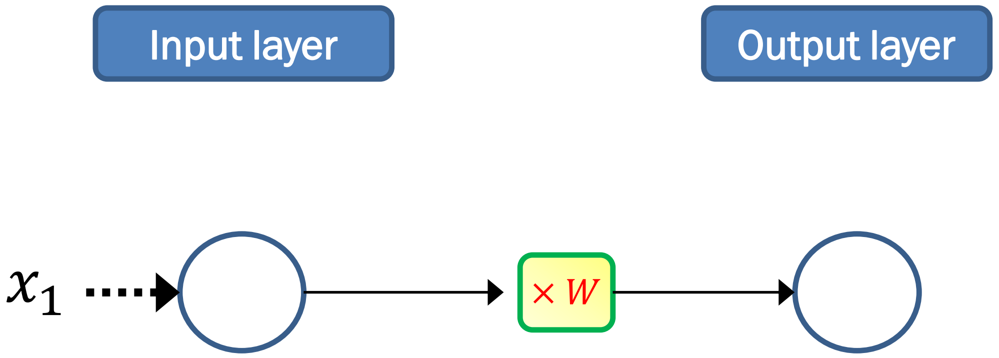


Layout

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_9 \\ x_{10} \end{bmatrix}$$

Input features = 1
Output features = 1

# of feature = 1



$$H(x_i) = Wx_i$$

Let's consider a simple case with W only.

Coding modules

Data Preparation

Model define

Cost function + optimizer

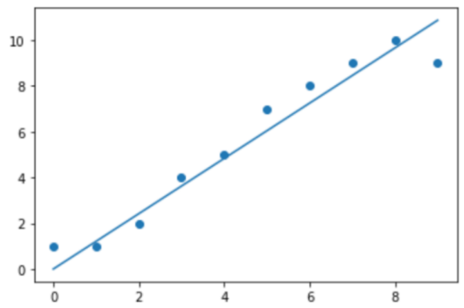
Model Test

x	y
0	1
1	1
2	2
3	4
4	5
5	7
6	8
7	9
8	10
9	9

$$H(x) = Wx + b$$

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(H(x_i) - y_i)^2$$

$$W := W - \alpha \frac{d}{dW} cost(W)$$



Lab 2A: Linear regression - vanilla

Github:

https://github.com/isaacye/SS2020_ML_Day2

What you may want to try :

1. Check the model define
2. Check the result by changing the starting point
3. Check the result by changing learning rate

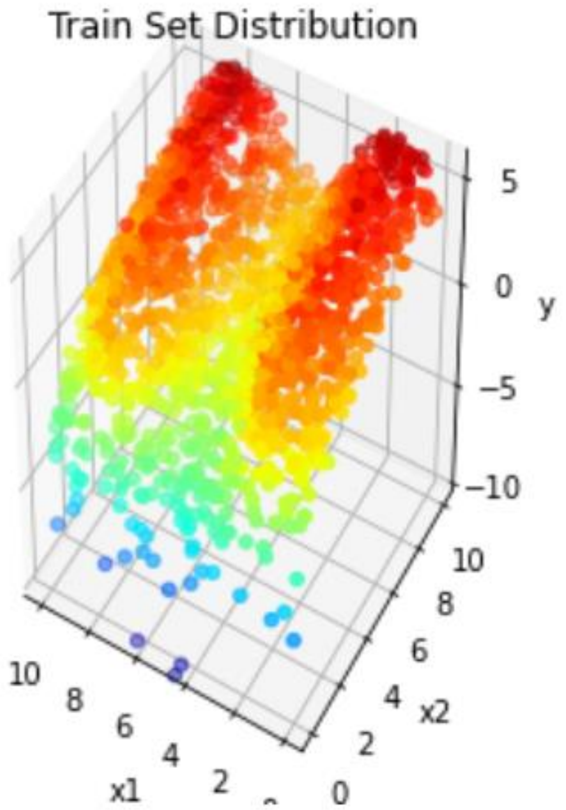
Break room

Linear regression: multivariable

Data preparation: Input (x_1,x_2, y)

x_1	x_2	y
3.91870851	2.32626914	0.73817558
2.59194437	6.00656071	4.3940048
6.46991632	3.57514815	0.61488728
:	:	:
4.56486433	2.14296641	3.95964088
1.29483514	1.67730041	3.48018992

```
num_data = 2400
x1 = np.random.rand(num_data) *10
x2 = np.random.rand(num_data) *10
e = np.random.normal(0, 0.5, num_data)
X= np.array([x1,x2]).T  # T for transpose from (2, 2400) to (2400, 2)
y=2*np.sin(x1) + np.log(0.5*x2**2)+e
```



Model (Hypothesis)

$$H(x_1, x_2) = w_1 x_1 + w_2 x_2 + b$$

For the data with n number of features, it is can be written as

$$H(x_1, x_2, x_3, \ldots, x_n) = w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_n x_n + b$$

Expression in matrix

$$H(x_{i1}, x_{i2}) = w_1 x_{i1} + w_2 x_{i2} + b$$

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = w_1 x_1 + w_2 x_2 + b$$

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = \begin{bmatrix} w_1 x_{11} + w_2 x_{12} + b \\ w_1 x_{21} + w_2 x_{22} + b \\ \vdots \\ w_1 x_{n1} + w_2 x_{n2} + b \end{bmatrix}$$

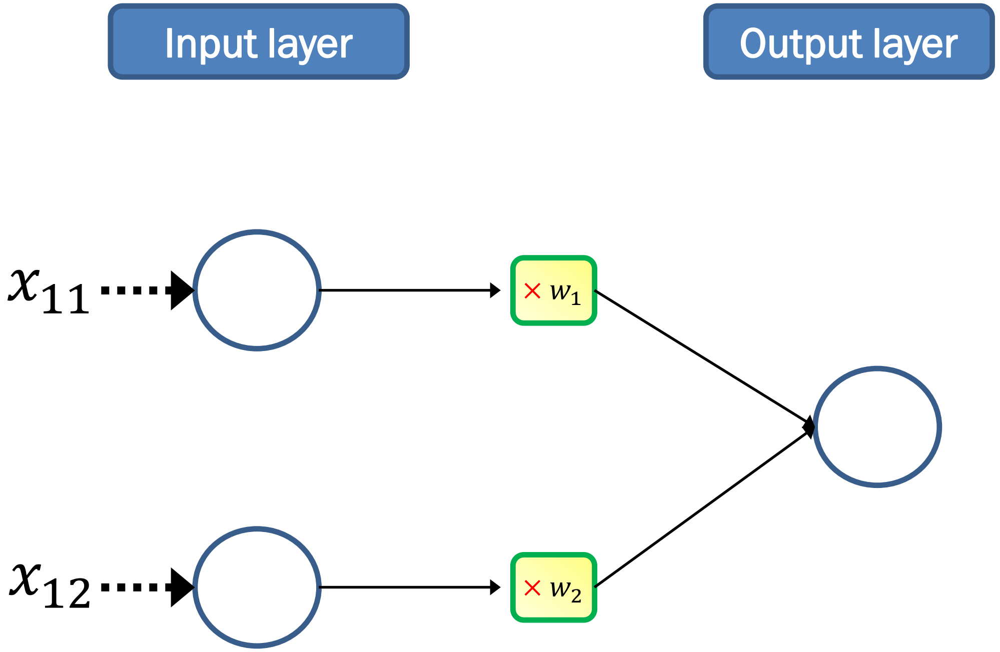
$$H(X) = XW + b$$

Layout

Input features = 2
Output features = 1

# of feature = 1

$$\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{bmatrix}$$



$$H(x_{i1}, x_{i2}) = w_1 x_{i1} + w_2 x_{i2}$$

Let's consider a simple case with W only.

Cost function

$$H(X) = XW + b$$

$$cost = \frac{1}{m}\sum_{i=1}^{m}(H(x_{i1}, x_{i2}) - y_i)^2$$

We want to minimize the cost as well!

Coding modules

Data Preparation

Model define

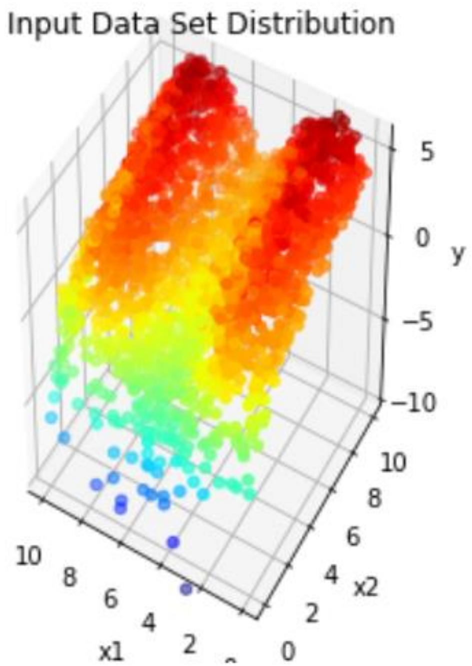
Cost function + optimizer

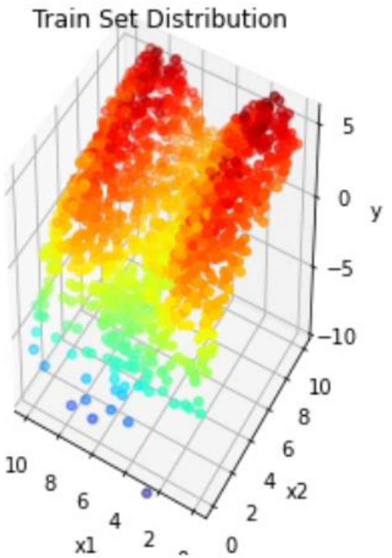
Model Test

Data preparation

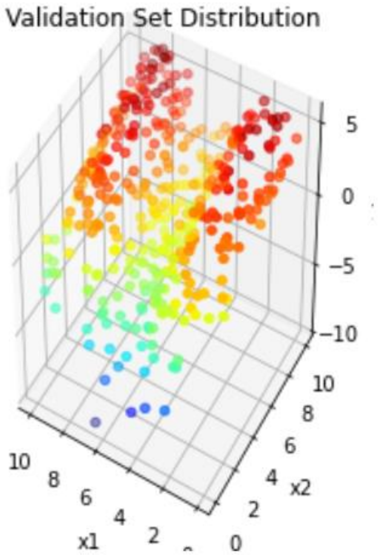
Data Preparation

x_1	x_2	y
3.91870851	2.32626914	0.73817558
2.59194437	6.00656071	4.3940048
6.46991632	3.57514815	0.61488728
:	:	:
4.56486433	2.14296641	3.95964088
1.29483514	1.67730041	3.48018992

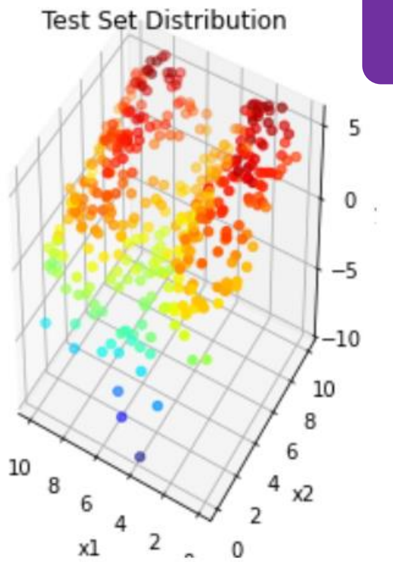




Train set



Validation set



Testing set

In the code

```
train_X, train_y = X[:1600, :], y[:1600]
val_X, val_y = X[1600:2000, :], y[1600:2000]
test_X, test_y = X[2000:, :], y[2000:]
```

Model define

Model define

In the code

```
import torch
import torch.nn as nn

class LinearModel(nn.Module):
  def __init__(self):
    super(LinearModel, self).__init__()
    self.linear = nn.Linear(in_features=2, out_features=1, bias=True)

  def forward(self, x):
    return self.linear(x)
```

Linear model in PyTorch

Linear

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: $(N, *, H_{in})$ where $*$ means any number of additional dimensions and $H_{in} = \text{in\_features}$
- Output: $(N, *, H_{out})$ where all but the last dimension are the same shape as the input and $H_{out} = \text{out\_features}$.

Cost (loss) function + Optimizer

Cost function + optimizer

Loss function

In the code

```
reg_loss = nn.MSELoss()
```

MSELoss

CLASS `torch.nn.MSELoss(size_average=None, reduce=None, reduction='mean')` [SOURCE]

Creates a criterion that measures the mean squared error (squared L2 norm) between each element in the input x and target y.

Optimizer

In the code

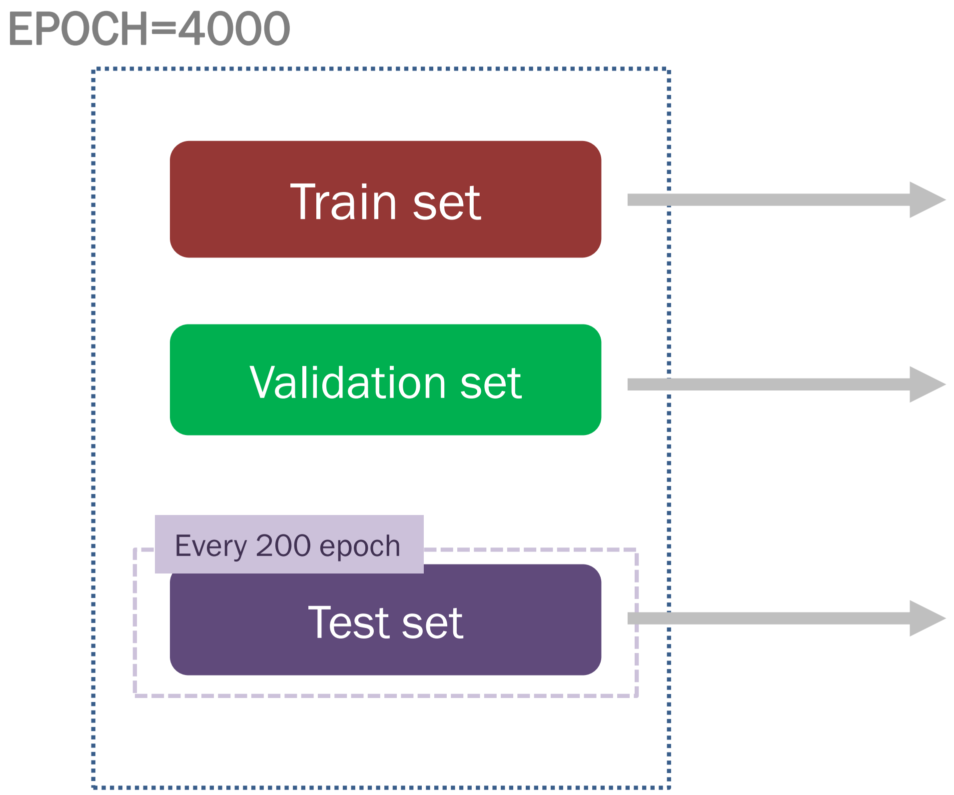
```
lr = 0.005
optimizer = optim.SGD(model.parameters(), lr =lr)
```

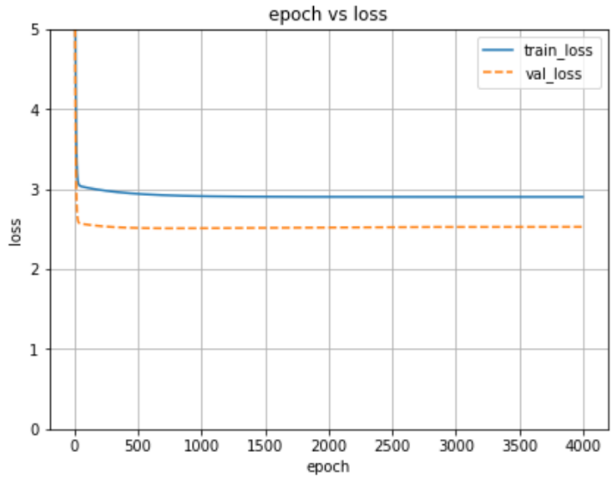
CLASS `torch.optim.SGD(params, lr=<required parameter>, momentum=0, dampening=0, weight_decay=0, nesterov=False)` [SOURCE]

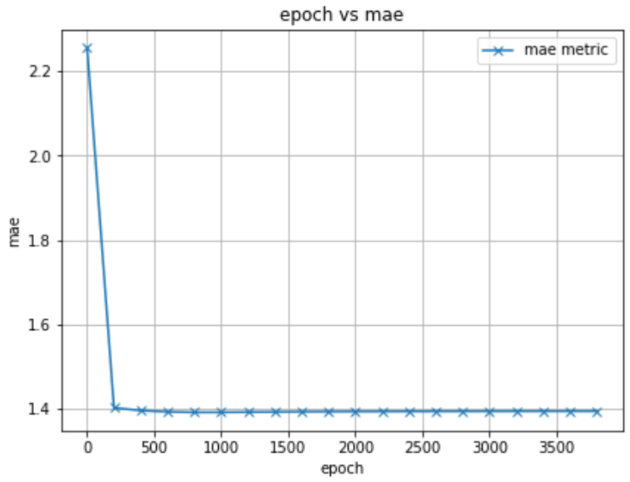
Implements stochastic gradient descent (optionally with momentum).

Model test

Model Test







Lab 2B: Linear regression – Linear model

1. Check the model define (linear model)

2. Check the result by varying learning rate

3. Check the result w/ w/o bias

4. Check the result with different number of Epoch

Break room

Lab 2B: Linear regression – Linear model

Github:

https://github.com/isaacye/SS2020_ML_Day2

What you may want to try :

1. Check the model define
2. Check the result by changing learning rate
3. Increase size of data and re-run it

Break room

Session break:

Please come back by 2 PM