AI VIETNAM
Free AI Course 2020

Tensorflow (Draft)

Deep Learning Framework

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2020

Outline

- Introduction to Tensorflow
- Tensorflow/Keras
- Model Construction – Linear Regression
- Model Construction – Logistic Regression
- Model Construction – Softmax Regression
- Model Training
- Model Saving and Loading

Introduction

https://en.wikipedia.org/wiki/TensorFlow

Developer(s)	Google Brain Team[1]
Initial release	November 9, 2015; 4 years ago
Stable release	2.2.0[2] / May 6, 2020; 1 month ago
Repository	github.com/tensorflow/tensorflow
Written in	Python, C++, CUDA
Platform	Linux, macOS, Windows, Android, JavaScript[3]
Type	Machine learning library
License	Apache License 2.0
Website	www.tensorflow.org

Installation

```
pip install tensorflow
```

Package Declaration

```
import tensorflow as tf
```

Example 1

```
import tensorflow as tf

print("TensorFlow version: ", tf.__version__)
print("Keras version: ", tf.keras.__version__)
```

```
TensorFlow version:  2.0.0
Keras version:  2.2.4-tf
```

Example 2

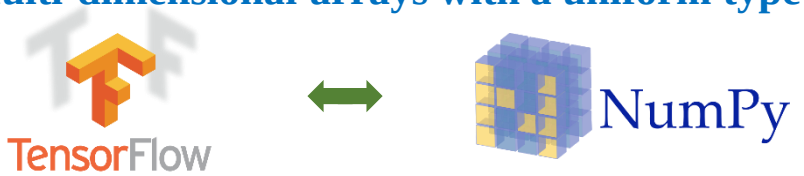
```
import tensorflow as tf

if tf.test.is_gpu_available():
    print('Running on GPU')
else:
    print('Running on CPU')
```

```
Running on CPU
```

Introduction

❖ Tensor

- ❖ ~ ndarray in Numpy
- ❖ Run on both CPU and GPU
- ❖ All tensors are immutable
- ❖ Multi-dimensional arrays with a uniform type




```python
import tensorflow as tf
import numpy as np

# create a ndarray
an_array = np.array([1,3,5,7,11])
print(type(an_array), an_array)

# convert from ndarray to tensor
a_tensor = tf.convert_to_tensor(an_array)
print(a_tensor)

# convert from tensor to ndarray
array_2 = a_tensor.numpy()
print(type(array_2), array_2)
```

```
<class 'numpy.ndarray'> [ 1  3  5  7 11]
tf.Tensor([ 1  3  5  7 11], shape=(5,), dtype=int32)
<class 'numpy.ndarray'> [ 1  3  5  7 11]
```

Important attributes

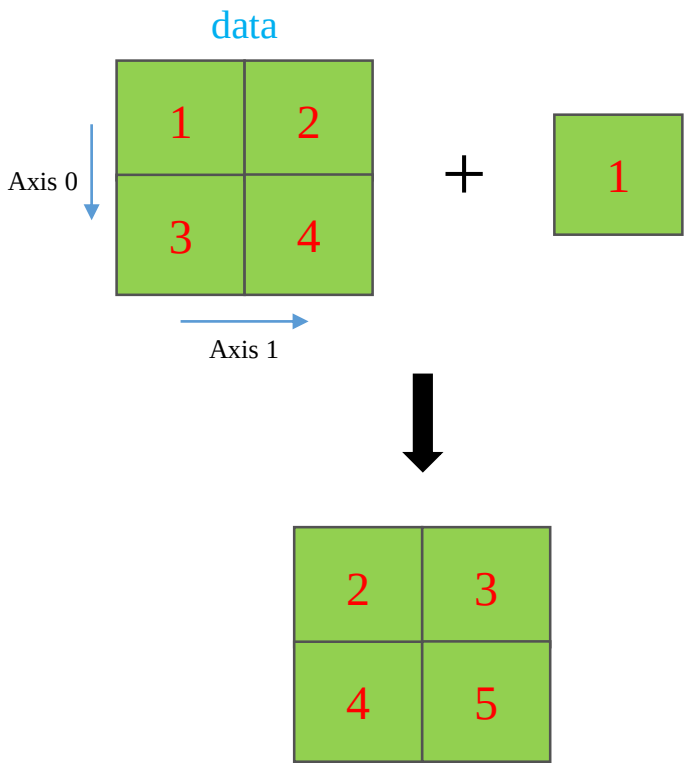
```python
import tensorflow as tf
import numpy as np

# create a 2x3 aray
an_array = np.array([[1,2],
                     [3,4],
                     [5,6]])

# convert to tensor
a_tensor = tf.convert_to_tensor(an_array)

# no. of elements on each axis
print(a_tensor.shape)
# data type
print(a_tensor.dtype)

# no. of axises
print(a_tensor.ndim)
```

```
(3, 2)
<dtype: 'int32'>
2
```

Introduction

❖ Tensor

❖ Broadcasting



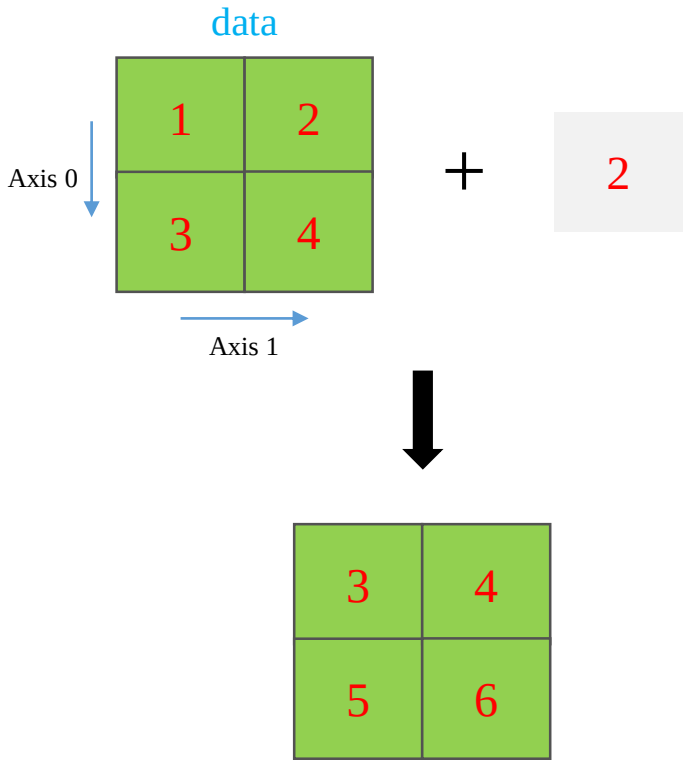

```python
import tensorflow as tf
import numpy as np

# create two tensors
tensor_1 = tf.convert_to_tensor([[1,2],
                                 [3,4]])
tensor_2 = tf.convert_to_tensor([1])

# add two tensors
tensor_3 = tensor_1 + tensor_2

print('tensor_1: \n', tensor_1)
print('tensor_2: \n', tensor_2)
print('tensor_3: \n', tensor_3)
```

```
tensor_1:
 tf.Tensor(
[[1 2]
 [3 4]], shape=(2, 2), dtype=int32)
tensor_2:
 tf.Tensor([1], shape=(1,), dtype=int32)
tensor_3:
 tf.Tensor(
[[2 3]
 [4 5]], shape=(2, 2), dtype=int32)
```

Introduction

❖ Tensor

❖ Broadcasting



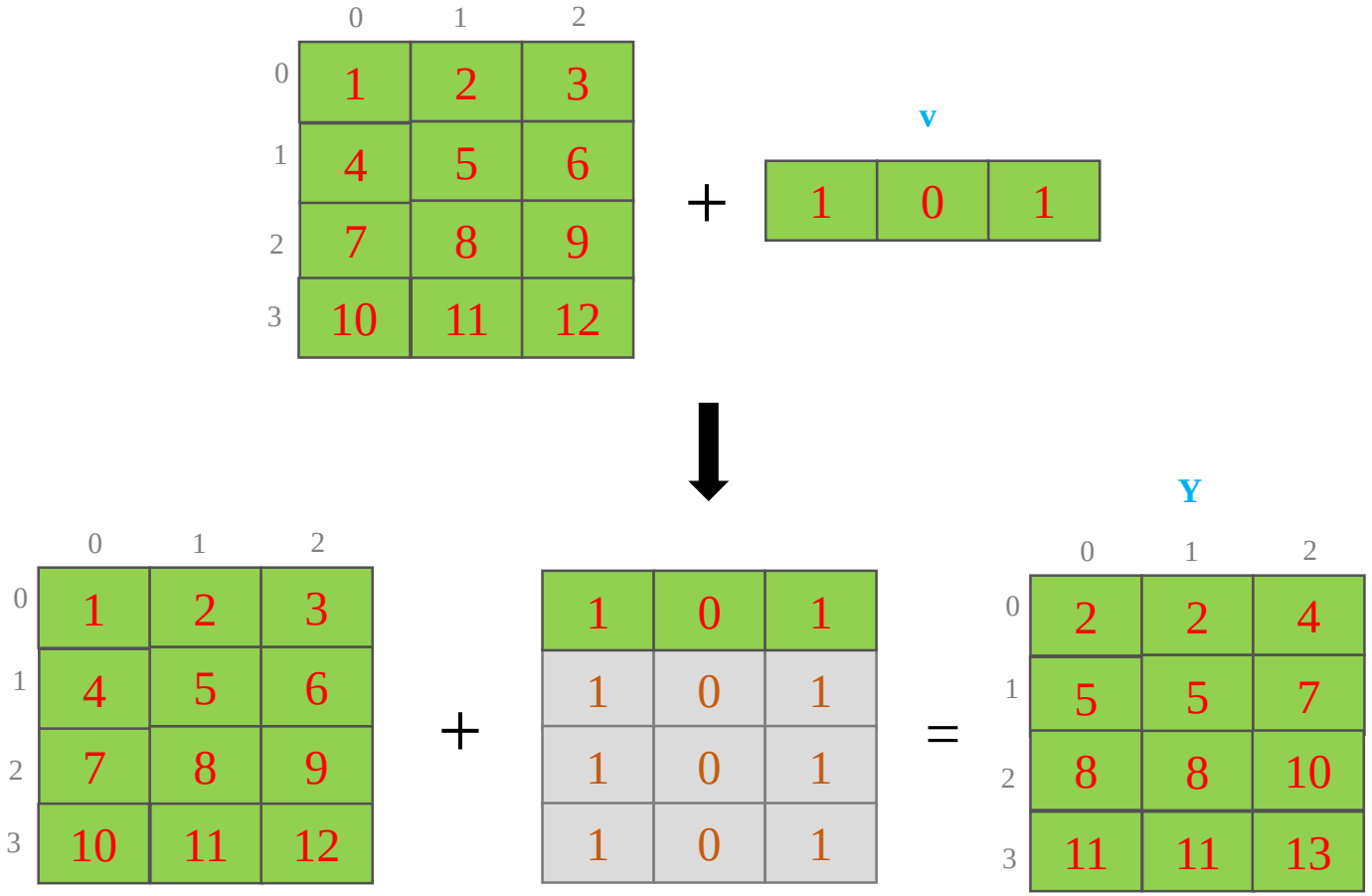

```python
import tensorflow as tf
import numpy as np

# create 2x2 tensor
tensor_1 = tf.convert_to_tensor([[1,2],
                                 [3,4]])

# add a number to a tensor
tensor_2 = tensor_1 + 2

print('tensor_1: \n', tensor_1)
print('tensor_2: \n', tensor_2)
```

```
tensor_1:
 tf.Tensor(
[[1 2]
 [3 4]], shape=(2, 2), dtype=int32)
tensor_2:
 tf.Tensor(
[[3 4]
 [5 6]], shape=(2, 2), dtype=int32)
```

Introduction

❖ Tensor

❖ Broadcasting




```python
import tensorflow as tf
import numpy as np

# create 4x3 tensor
np_data = np.array(range(1, 13)).reshape(4, 3)
tensor_1 = tf.convert_to_tensor(np_data)

# create the second tensor
tensor_2 = tf.convert_to_tensor([1, 0, 1])

# add a number to a tensor
tensor_3 = tensor_1 + tensor_2

print('tensor_1: \n', tensor_1)
print('tensor_2: \n', tensor_2)
print('tensor_3: \n', tensor_3)
```

```
tensor_1:
 tf.Tensor(
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]], shape=(4, 3), dtype=int32)
tensor_2:
 tf.Tensor([1 0 1], shape=(3,), dtype=int32)
tensor_3:
 tf.Tensor(
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]], shape=(4, 3), dtype=int32)
```

Introduction

❖ Tensor

❖ Important functions

Squared Difference

$$sd = (x - y)^2$$

x
1
2
3
4

$(x - y)^2 =$

sd
16
9
4
1

```python
import tensorflow as tf

# create a list
x = [1,2, 3, 4]
y = 5

# compute squared difference
sd = tf.math.squared_difference(x,y)
print(sd)
```

```
tf.Tensor([16  9  4  1], shape=(4,), dtype=int32)
```

```python
import tensorflow as tf

# create a tensor
x = tf.convert_to_tensor([1,2, 3, 4])
y = 5

# compute squared difference
sd = tf.math.squared_difference(x,y)
print(sd)
```

```
tf.Tensor([16  9  4  1], shape=(4,), dtype=int32)
```

```python
import tensorflow as tf
import numpy as np

# create an ndarray
x = np.array([1,2, 3, 4])
y = 5

# compute squared difference
sd = tf.math.squared_difference(x,y)
print(sd)
```

```
tf.Tensor([16  9  4  1], shape=(4,), dtype=int32)
```

Introduction

❖ Tensor

❖ Important functions

random.normal()

```
1  import tensorflow as tf
2
3  rand = tf.random.normal(shape = (3,2), mean=0, stddev=1)
4  print(rand)
```

```
tf.Tensor(
[[ 0.47103247 -0.12765862]
 [-0.26556632 -0.05912822]
 [ 1.0851953  -0.55289406]], shape=(3, 2), dtype=float32)
```

random.uniform()

```
1  import tensorflow as tf
2
3  rand = tf.random.uniform(shape=(3,2), minval=0,
4                           maxval=9, dtype=tf.int32)
5  print(rand)
```

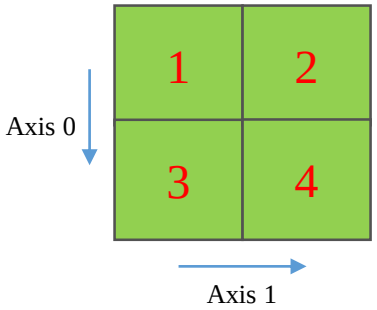
```
tf.Tensor(
[[4 0]
 [1 8]
 [0 2]], shape=(3, 2), dtype=int32)
```

Introduction

❖ Tensor

❖ Important functions

tensor_1



tensor_2

3	4
5	6

tensor_3

1	2
3	4
3	4
5	6

tensor_4

1	2	3	4
3	4	5	6

concat()

```python
import tensorflow as tf

tensor_1 = tf.random.normal(shape=(2,2), mean=0, stddev=1)
tensor_2 = tf.random.normal(shape=(2,2), mean=0, stddev=1)

# concat two tensors along axis_0
tensor_3 = tf.concat([tensor_1, tensor_1], axis=0)

# concat two tensors along axis_1
tensor_4 = tf.concat([tensor_1, tensor_1], axis=1)

print(tensor_1.shape)
print(tensor_2.shape)
print(tensor_3.shape)
print(tensor_4.shape)
```

```
(2, 2)
(2, 2)
(4, 2)
(2, 4)
```

Introduction

❖ Images in files

❖ Important functions

1	2
7	9
8	6

.argmin(axis=0) = | 0 | 0 |

1	2
7	9
8	6

.argmin(axis=1) = | 0 | 0 | 1 |

argmin()

```
1  import tensorflow as tf
2
3  # create a tensor
4  tensor = tf.random.uniform(shape=(3,6), minval=0,
5                             maxval=20, dtype=tf.int32)
6
7  # find the index of the min value
8  min_position_1 = tf.argmin(tensor, axis=0)
9  min_position_2 = tf.argmin(tensor, axis=1)
10
11 print(tensor)
12 print('min_position_1: ', min_position_1)
13 print('min_position_2: ', min_position_2)
```

```
tf.Tensor(
[[ 5  0 13 11  4 10]
 [19  6  7  5 17  6]
 [18  9  9  2  1 11]], shape=(3, 6), dtype=int32)
min_position_1:  tf.Tensor([0 0 1 2 2 1], shape=(6,), dtype=int64)
min_position_2:  tf.Tensor([1 3 4], shape=(3,), dtype=int64)
```

Introduction

❖ Images in files

❖ Important functions

1	2
7	9
8	6

.argmax(axis=0) =

2	1

1	2
7	9
8	6

.argmax(axis=1) =

1	1	0

argmax()

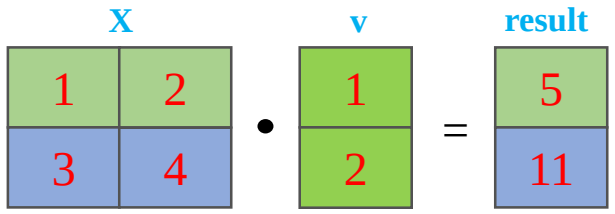
```python
import tensorflow as tf

# create a tensor
tensor = tf.random.uniform(shape=(3,6), minval=0,
                           maxval=20, dtype=tf.int32)

# find the index of the max value
max_position_1 = tf.argmax(tensor, axis=0)
max_position_2 = tf.argmax(tensor, axis=1)

print(tensor)
print('max_position_1: ', max_position_1)
print('max_position_2: ', max_position_2)
```

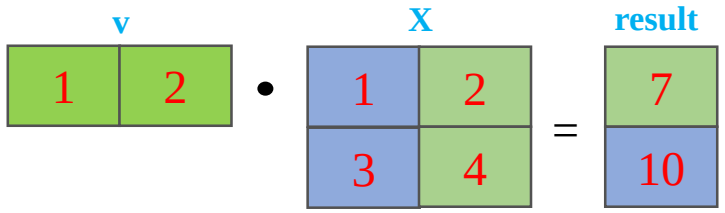
```
tf.Tensor(
[[ 8 10 13  1 12  1]
 [ 8 19  5  3 16 16]
 [ 2 16  7  1  4 17]], shape=(3, 6), dtype=int32)
max_position_1:  tf.Tensor([0 1 0 1 1 2], shape=(6,), dtype=int64)
max_position_2:  tf.Tensor([2 1 5], shape=(3,), dtype=int64)
```

Introduction

❖ Tensor

❖ Important functions





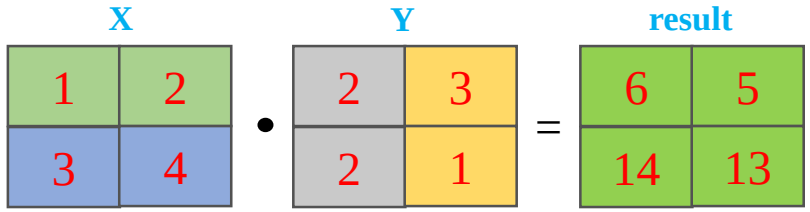

```python
import tensorflow as tf
import numpy as np

matrix = tf.convert_to_tensor([[1, 2], [3, 4]])
vector = tf.convert_to_tensor([1, 2])

print('matrix shape \n', matrix.shape)
print('vector shape \n', vector.shape)

vector = tf.reshape(vector, (2, 1))
print('vector reshape \n', vector.shape)

result1 = tf.matmul(matrix, vector)
print(result1)

result2 = tf.matmul(tf.transpose(vector), matrix)
print(result2)
```

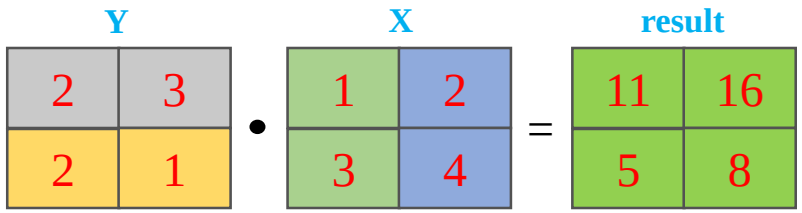
```
matrix shape
 (2, 2)
vector shape
 (2,)
vector reshape
 (2, 1)
tf.Tensor(
[[ 5]
 [11]], shape=(2, 1), dtype=int32)
tf.Tensor([[ 7 10]], shape=(1, 2), dtype=int32)
```

Introduction

❖ Tensor

❖ Important functions





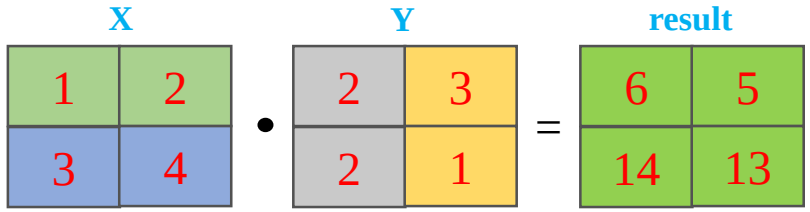

```python
import tensorflow as tf
import numpy as np

matrix1 = tf.convert_to_tensor([[1, 2], [3, 4]])
matrix2 = tf.convert_to_tensor([[2, 3], [2, 1]])

print('matrix1 shape \n', matrix1.shape)
print('matrix2 shape \n', matrix2.shape)

result1 = tf.matmul(matrix1, matrix2)
print(result1)

result2 = tf.matmul(matrix2, matrix1)
print(result2)
```

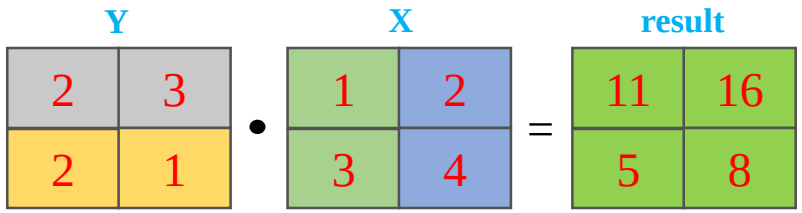
```
matrix1 shape
 (2, 2)
matrix2 shape
 (2, 2)
tf.Tensor(
[[ 6  5]
 [14 13]], shape=(2, 2), dtype=int32)
tf.Tensor(
[[11 16]
 [ 5  8]], shape=(2, 2), dtype=int32)
```

Introduction

❖ Tensor

❖ Important functions






```python
import tensorflow as tf
import numpy as np

matrix1 = tf.convert_to_tensor([[1, 2], [3, 4]])
matrix2 = tf.convert_to_tensor([[2, 3], [2, 1]])

print('matrix1 shape \n', matrix1.shape)
print('matrix2 shape \n', matrix2.shape)

result1 = matrix1@matrix2
print(result1)

result2 = matrix2@matrix1
print(result2)
```

```
matrix1 shape
 (2, 2)
matrix2 shape
 (2, 2)
tf.Tensor(
[[ 6  5]
 [14 13]], shape=(2, 2), dtype=int32)
tf.Tensor(
[[11 16]
 [ 5  8]], shape=(2, 2), dtype=int32)
```

Introduction

- Variable
  - Represents a tensor whose value can be changed

```
# variable

var = tf.Variable([1.0, 2.0])
print(var.numpy())

var.assign([7, 9])
print(var.numpy())
```

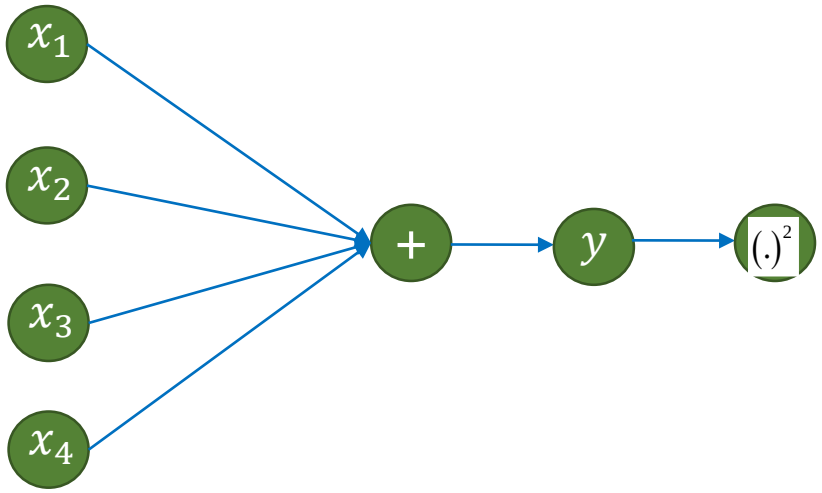
```
[1. 2.]
[7. 9.]
```

```
# variables

var = tf.Variable([1.0, 2.0])

var.assign_add([5, 3])
print(var.numpy())

var.assign_sub([2, 3])
print(var.numpy())
```

```
[6. 5.]
[4. 2.]
```

Introduction

❖ Gradient computation




```
1  import tensorflow as tf
2
3  x = tf.ones((2, 2))
4
5  with tf.GradientTape() as t:
6      t.watch(x)
7      y = tf.reduce_sum(x)
8      z = tf.multiply(y, y)
9
10 # Derivative of z with respect to the original input tensor x
11 dz_dx = t.gradient(z, x)
12 print(dz_dx)
```

```
tf.Tensor(
[[8. 8.]
 [8. 8.]], shape=(2, 2), dtype=float32)
```

Introduction

❖ Gradient computation

```
x = tf.constant(3.0)
with tf.GradientTape(persistent=True) as t:
    t.watch(x)
    y = x * x
    z = y * y

dz_dx = t.gradient(z, x)  # 108.0 (4*x^3 at x = 3)
dy_dx = t.gradient(y, x)  # 6.0
del t  # Drop the reference to the tape

print(dz_dx)
print(dy_dx)
```

```
tf.Tensor(108.0, shape=(), dtype=float32)
tf.Tensor(6.0, shape=(), dtype=float32)
```

Introduction

❖ Gradient computation

```
x = tf.Variable(1.0)  # Create a Tensorflow variable initialized to 1.0

with tf.GradientTape() as t:
    with tf.GradientTape() as t2:
        y = x * x * x
    # Compute the gradient inside the 't' context manager
    # which means the gradient computation is differentiable as well.
    dy_dx = t2.gradient(y, x)
d2y_dx2 = t.gradient(dy_dx, x)

print(dy_dx.numpy())
print(d2y_dx2.numpy())
```

```
3.0
6.0
```

Introduction

❖ Gradient computation

```
# Ví dụ 1
import tensorflow as tf

# tạo biến x có giá trị là 2.0
x = tf.ones((1))*2

with tf.GradientTape() as tape:
    tape.watch(x)

    # xây dựng hàm số
    g_x = -2*x + 5
    h_x =  6*g_x + 3

# Tính đạo hàm cho hàm số h(x) tại x = 2
dh_dx = tape.gradient(h_x, x)
print(dh_dx)
```

```
tf.Tensor([-12.], shape=(1,), dtype=float32)
```

Introduction

❖ Gradient computation

```
1  # Ví dụ 2
2  import tensorflow as tf
3
4  # tạo biến x có giá trị là 1.0
5  x = tf.ones((1))
6
7  with tf.GradientTape() as tape:
8      tape.watch(x)
9
10     # xây dựng hàm số
11     g_x = 3*x*x + 2
12     h_x = tf.exp(g_x)
13
14 # Tính đạo hàm cho hàm số h(x) tại x = 1
15 dh_dx = tape.gradient(h_x, x)
16 print(dh_dx)
```

```
tf.Tensor([890.479], shape=(1,), dtype=float32)
```

Introduction

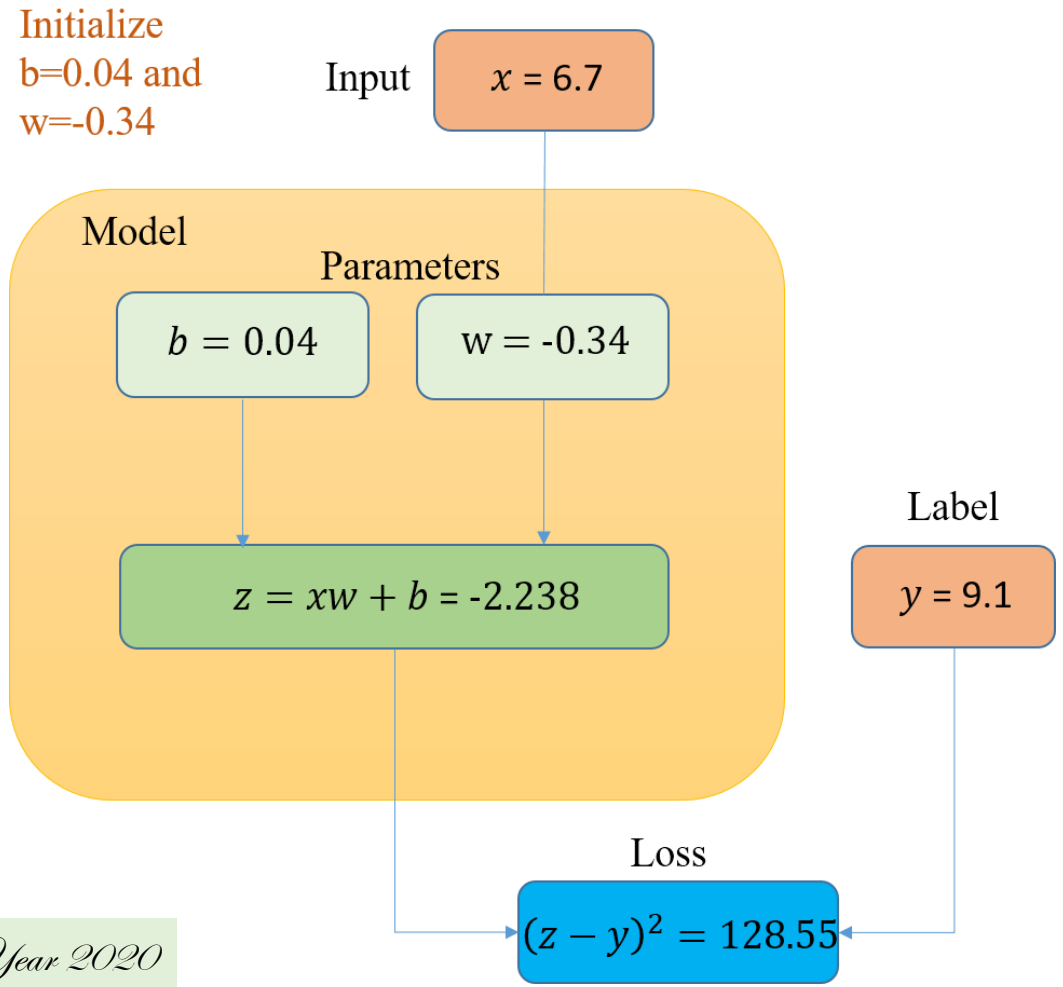
❖ Gradient computation

```python
# Ví dụ 3
import tensorflow as tf

# tạo biến x có giá trị là 1.0
x = tf.ones((1))

with tf.GradientTape() as tape:
    tape.watch(x)

    # xây dựng hàm số
    g_x = tf.math.cos(x*x*tf.exp(x) + 2*x)
    h_x = tf.exp(g_x)*tf.math.sin(tf.math.sqrt(g_x))

# Tính đạo hàm cho hàm số h(x) tại x = 1
dh_dx = tape.gradient(h_x, x)
print(dh_dx)
```

```
tf.Tensor([67.120346], shape=(1,), dtype=float32)
```

❖ Gradient computation

area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

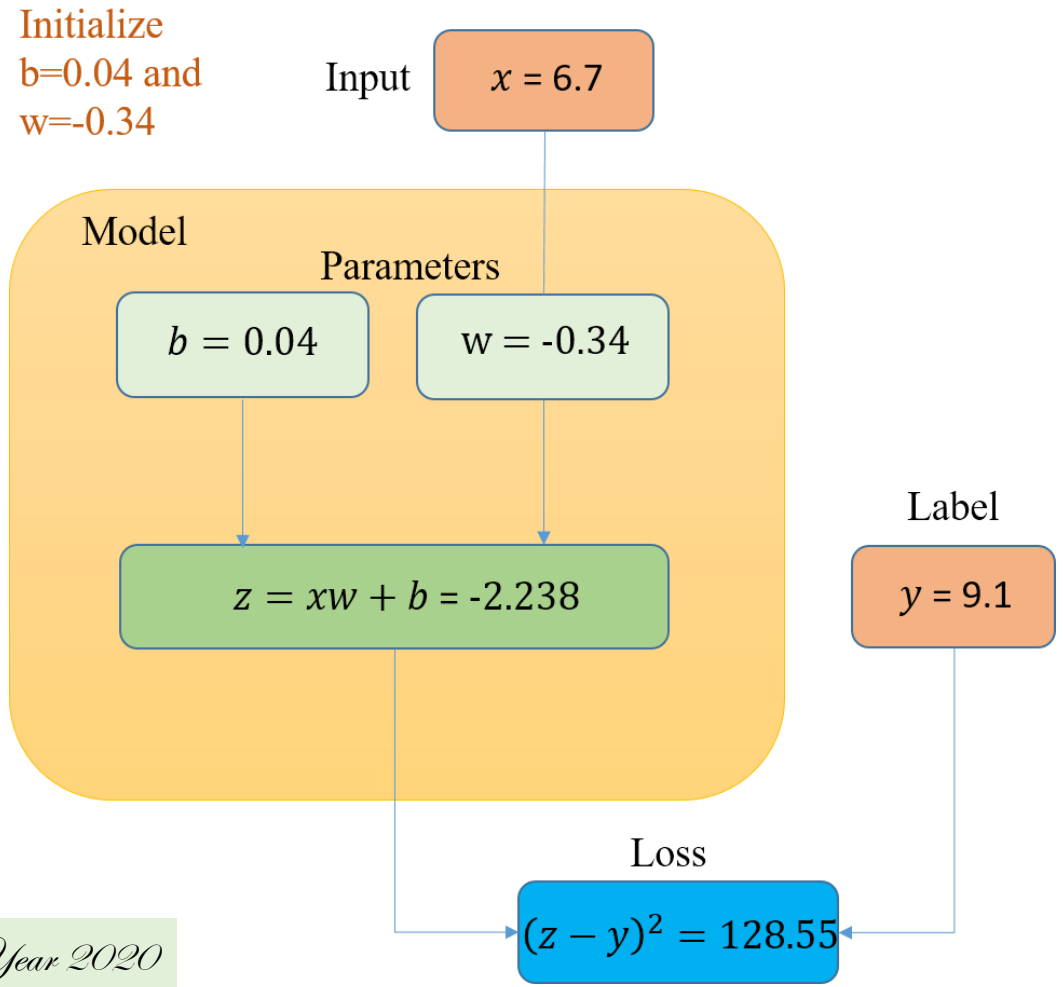



```python
import tensorflow as tf

w = tf.Variable([-0.34])
b = tf.Variable([0.04])

x = [6.7, 4.6, 3.5, 5.5]
x = tf.convert_to_tensor(x)

print('b: \n', b.numpy())
print('w: \n', w.numpy())
print('x: \n', x.numpy())

with tf.GradientTape(persistent=True) as tape:
    y = tf.math.multiply(x, w) + b
    print('y: \n', y.numpy())

    loss = tf.reduce_mean(y**2)
    print('loss: \n', loss.numpy())
```

```
b:
 [0.04]
w:
 [-0.34]
x:
 [6.7 4.6 3.5 5.5]
y:
 [-2.238     -1.524     -1.1500001 -1.83      ]
loss:
 3.000655
```

❖ Gradient computation

area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7




```python
import tensorflow as tf

w = tf.Variable(tf.random.normal((1,)))
b = tf.Variable(tf.zeros(1, dtype=tf.float32))

x = [6.7, 4.6, 3.5, 5.5]
x = tf.convert_to_tensor(x)

print('b: \n', b.numpy())
print('w: \n', w.numpy())
print('x: \n', x.numpy())

with tf.GradientTape(persistent=True) as tape:
    y = tf.math.multiply(x, w) + b
    print('y: \n', y.numpy())

    loss = tf.reduce_mean(y**2)
    print('loss: \n', loss.numpy())
```

```
b:
 [0.]
w:
 [0.03887156]
x:
 [6.7 4.6 3.5 5.5]
y:
 [0.2604395  0.1788092  0.13605048 0.2137936 ]
loss:
 0.04100472
```

Introduction

❖ Gradient computation

	area	price
	6.7	9.1
	4.6	5.9
	3.5	4.6
	5.5	6.7

```python
import tensorflow as tf

theta = tf.Variable([[0.04], [-0.34]])

x = [[1, 6.7],
     [1, 4.6],
     [1, 3.5],
     [1, 5.5]];
x = tf.convert_to_tensor(x)

print('theta: \n', theta.numpy())
print('x: \n', x.numpy())

with tf.GradientTape(persistent=True) as tape:
    y = x@theta
    print('y: \n', y.numpy())

    loss = tf.reduce_mean(y**2)
    print('loss: \n', loss.numpy())
```

```
theta:
 [[ 0.04]
 [-0.34]]
x:
 [[1.  6.7]
 [1.  4.6]
 [1.  3.5]
 [1.  5.5]]
y:
 [[-2.238    ]
 [-1.524    ]
 [-1.1500001]
 [-1.83     ]]
loss:
 3.000655
```

Introduction

❖ Gradient computation

area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

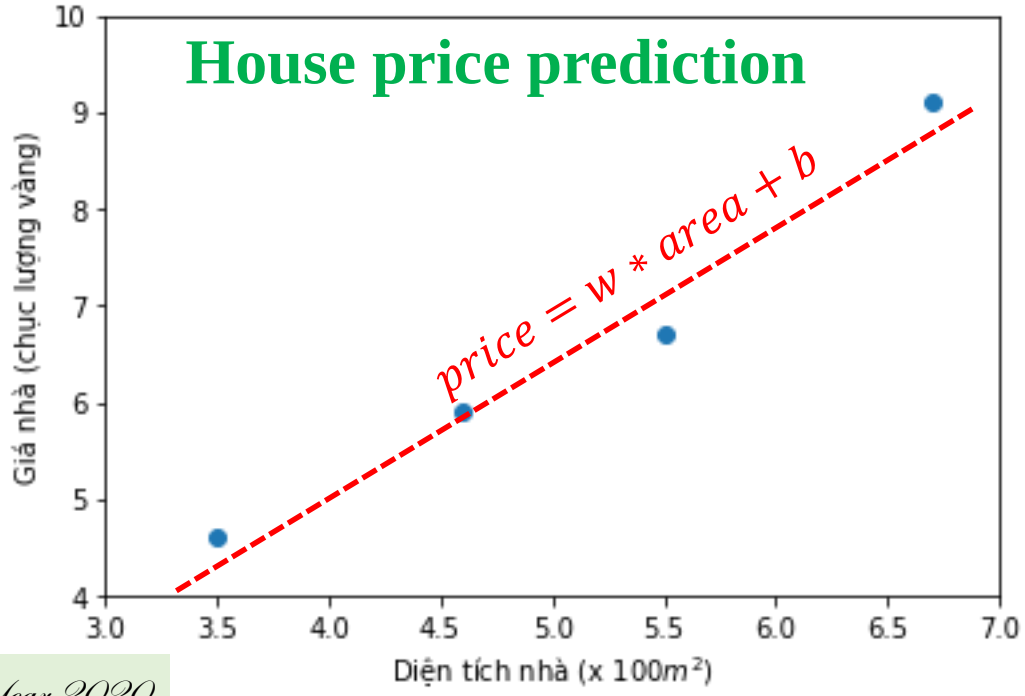
```
1  import tensorflow as tf
2
3  theta = tf.Variable(tf.random.normal((2,1)))
4
5  x = [[1, 6.7],
6       [1, 4.6],
7       [1, 3.5],
8       [1, 5.5]];
9  x = tf.convert_to_tensor(x)
10
11 print('theta: \n', theta.numpy())
12 print('x: \n', x.numpy())
13
14 with tf.GradientTape(persistent=True) as tape:
15     y = x@theta
16     print('y: \n', y.numpy())
17
18     loss = tf.reduce_mean(y**2)
19     print('loss: \n', loss.numpy())
```

```
theta:
 [[0.28044993]
 [1.5637906 ]]
x:
 [[1.  6.7]
 [1.  4.6]
 [1.  3.5]
 [1.  5.5]]
y:
 [[10.757846 ]
 [ 7.4738865]
 [ 5.753717 ]
 [ 8.881298 ]]
loss:
 70.893234
```

Linear Regression

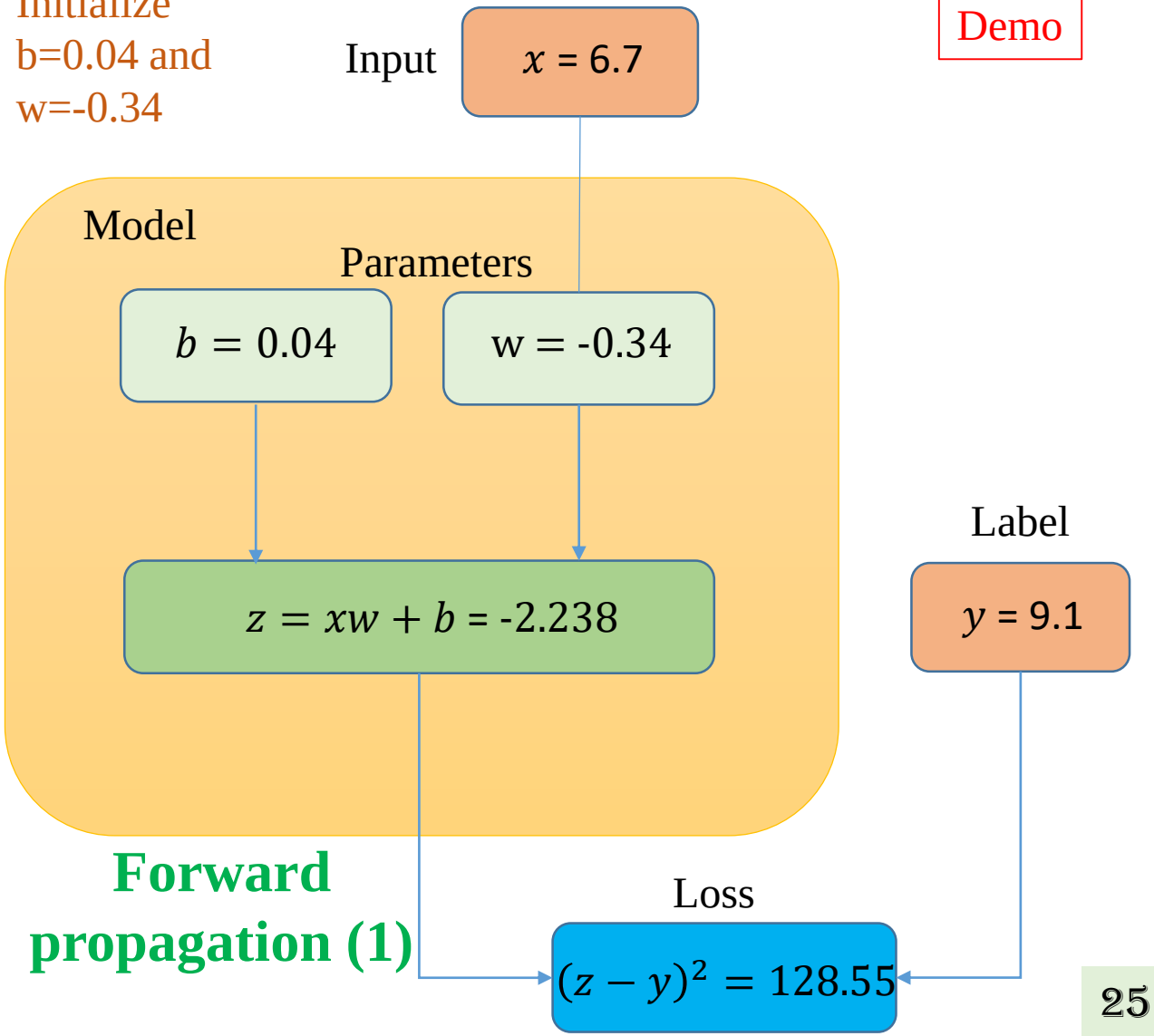
Given sample data

area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7



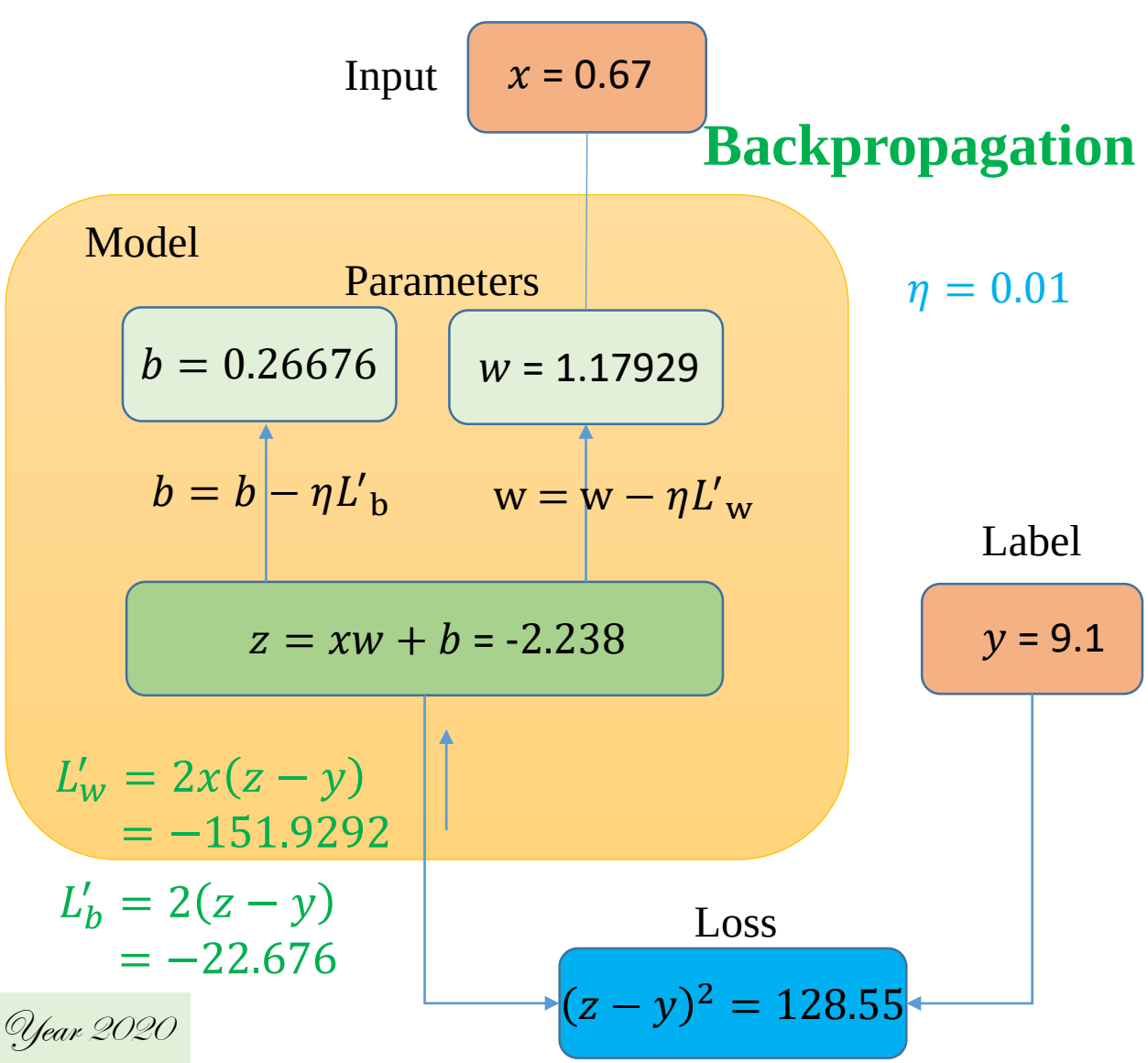
Initialize b=0.04 and w=-0.34

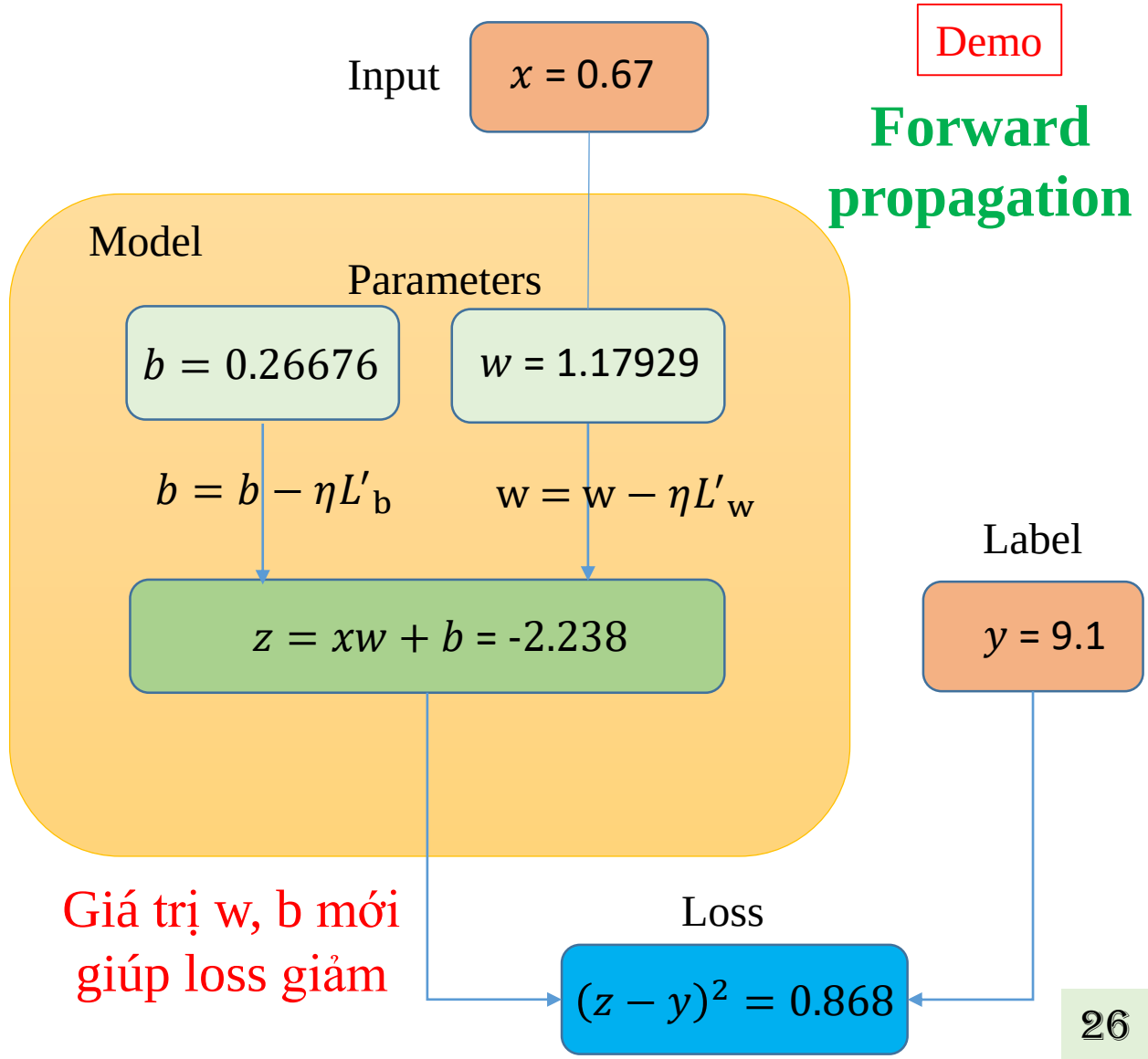
Demo



Forward propagation (1)

Linear Regression





Introduction

❖ Gradient computation

```
theta:
 [[ 0.04]
 [-0.34]]
x:
 [[1.  6.7]]
y:
 [[-2.238]]
loss:
 128.55025

tf.Tensor(
[[ -22.676 ]
 [-151.9292]], shape=(2, 1), dtype=float32)
```

```python
import tensorflow as tf

theta = tf.Variable([[0.04], [-0.34]])

x = [[1, 6.7]];
x = tf.convert_to_tensor(x)

y = [9.1];

print('theta: \n', theta.numpy())
print('x: \n', x.numpy())

with tf.GradientTape(persistent=True) as tape:
    y_hat = x@theta
    print('y: \n', y_hat.numpy())

    loss = tf.reduce_mean((y_hat-y)**2)
    print('loss: \n', loss.numpy())

    dtheta = tape.gradient(loss, theta)
    print(dtheta)
```

Introduction

- **Gradient computation**

Class in Python

❖ Create a class

keyword class

```
1  class SGD:
2      lr = 0.1
3
4  sgd = SGD()
5  print(sgd.lr)
```

```
0.1
```

__init__() function

```
1  class SGD:
2      def __init__(self, lr):
3          self.lr = lr
4
5  sgd = SGD(0.1)
6  print(sgd.lr)
```

```
0.1
```

self: a reference to the current instance of the class

Class in Python

❖ __call__ function

Built-in method

Be called like a function

```python
class SGD:
    def __init__(self, lr):
        self.lr = lr

    def __call__(self, value):
        self.lr = self.lr + value

sgd_instance = SGD(0.1)
sgd_instance(0.3)
print(sgd_instance.lr)
```

```
0.4
```

Outline

- Introduction to Tensorflow
- Tensorflow/Keras
- Model Construction – Linear Regression
- Model Construction – Logistic Regression
- Model Construction – Softmax Regression
- Model Training
- Model Saving and Loading

Keras

- Run on top of TensorFlow
- Integrated into Tensorflow

Package Declaration

```
1  import tensorflow as tf
2  import tensorflow.keras as keras
3
4  print(keras.__version__)
```

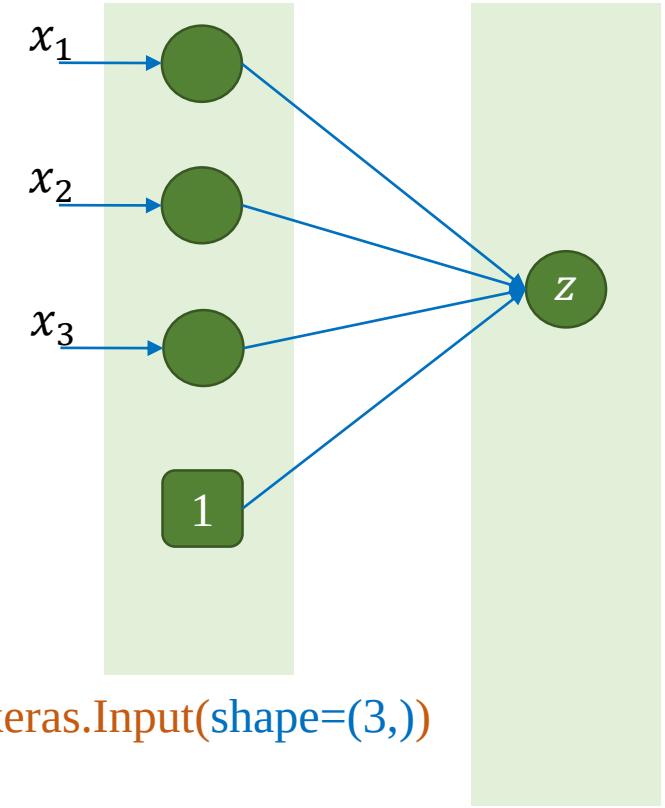
```
2.2.4-tf
```

Original author(s)	François Chollet [fr]
Developer(s)	various
Initial release	27 March 2015; 5 years ago
Stable release	2.3.1[1] / 7 October 2019; 8 months ago
Repository	github.com/keras-team/keras
Written in	Python
Platform	Cross-platform
Type	Neural networks
License	MIT
Website	keras.io

https://en.wikipedia.org/wiki/Keras

Keras

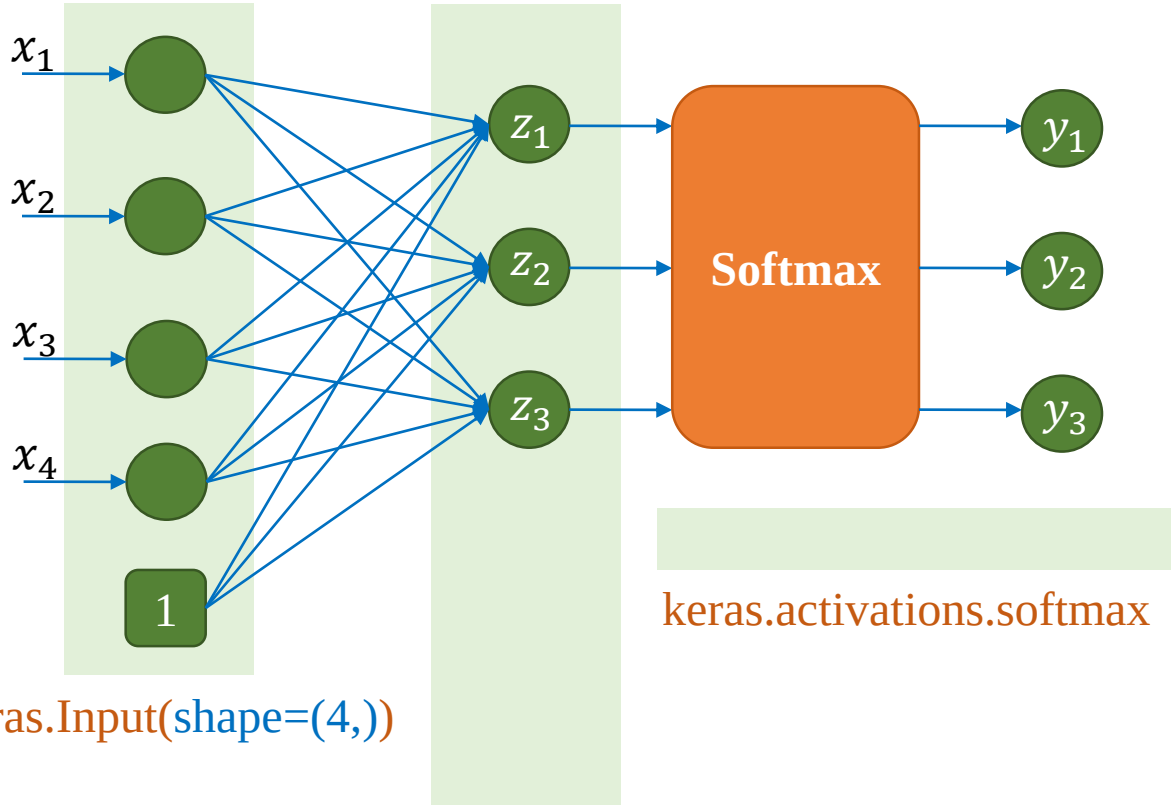
Model



keras.Input(shape=(3,))

keras.layers.Dense(units=1)

Model



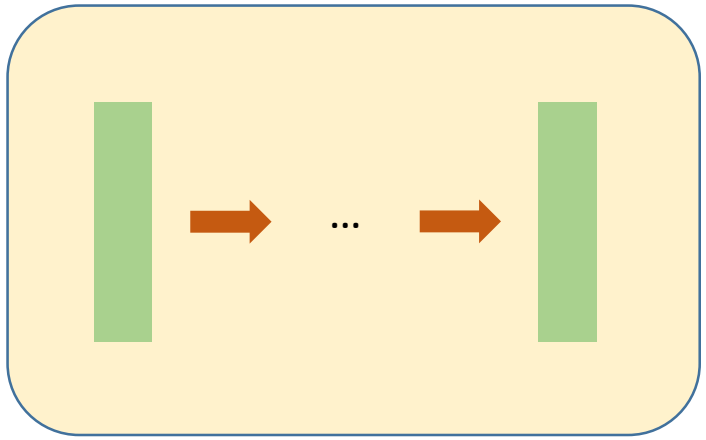
keras.Input(shape=(4,))

keras.activations.softmax

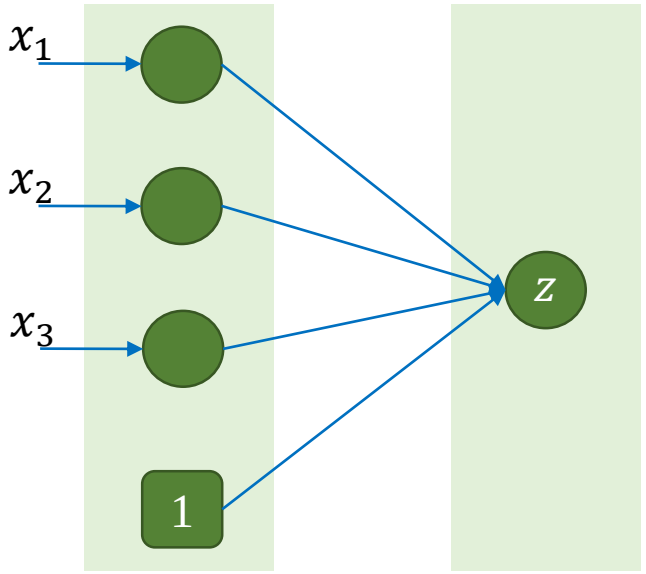
keras.layers.Dense(units=3)

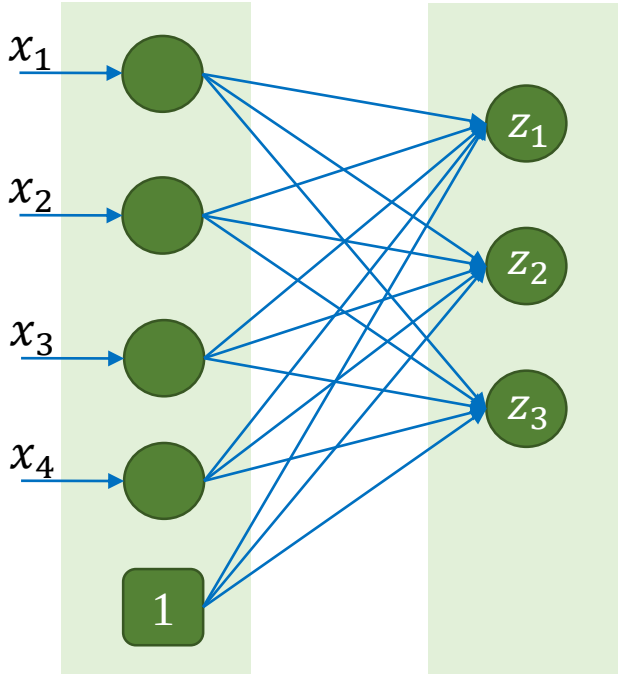
Keras

Model



keras.Sequential()



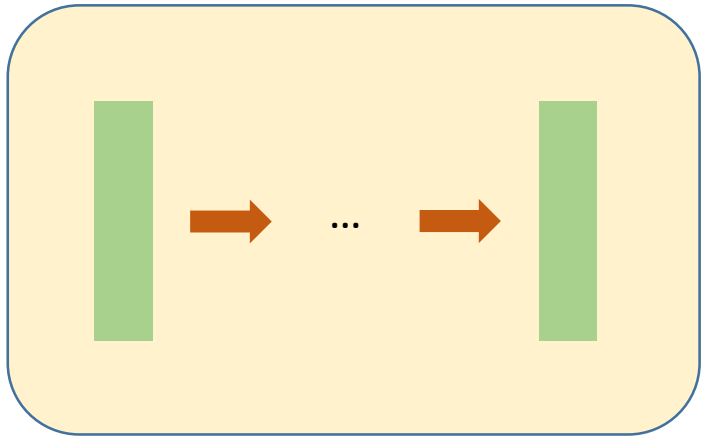



```python
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(3,)))
model.add(keras.layers.Dense(1))
```

```python
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(4,)))
model.add(keras.layers.Dense(3))
```

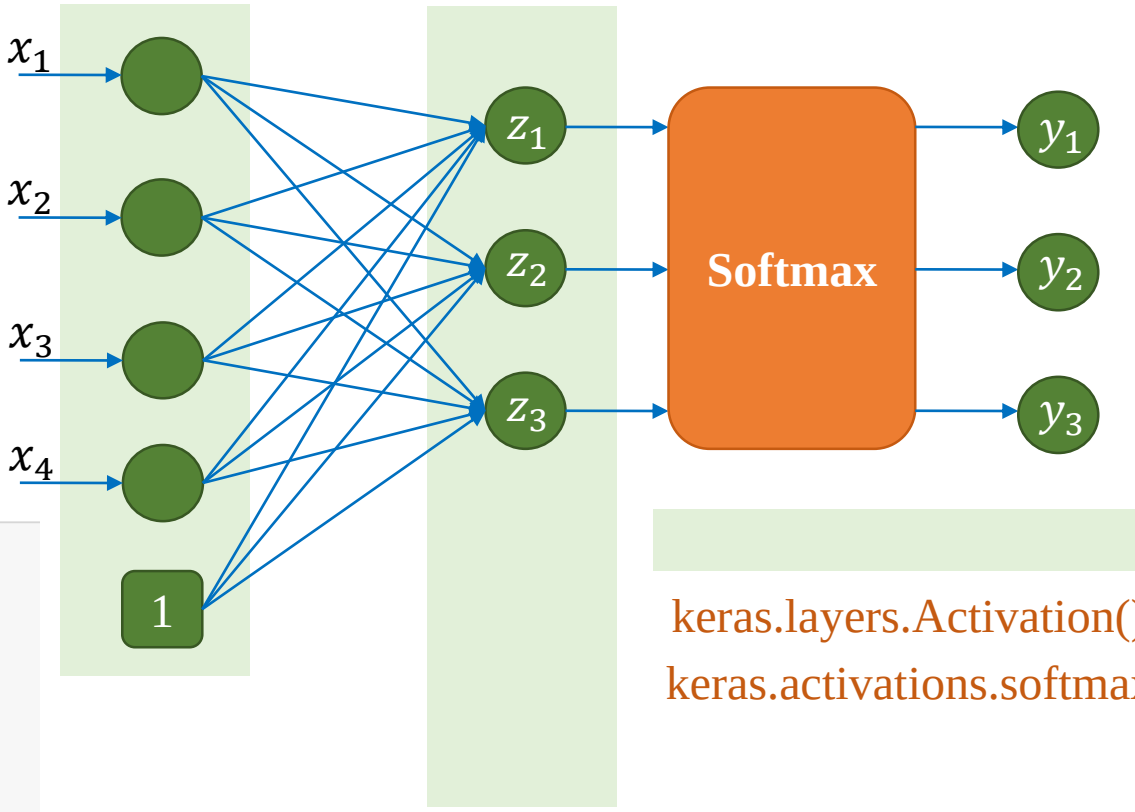
Keras

Model



keras.Sequential()

keras.Input(shape=(4,))



keras.layers.Activation()
keras.activations.softmax

keras.layers.Dense(units=3)

```
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(4,)))
model.add(keras.layers.Dense(3))
model.add(keras.layers.Activation(keras.activations.softmax))
```

Outline

- Introduction to Tensorflow
- Tensorflow/Keras
- Model Construction – Linear Regression
- Model Construction – Logistic Regression
- Model Construction – Softmax Regression
- Model Training
- Model Saving and Loading

Model Construction

❖ Linear regression

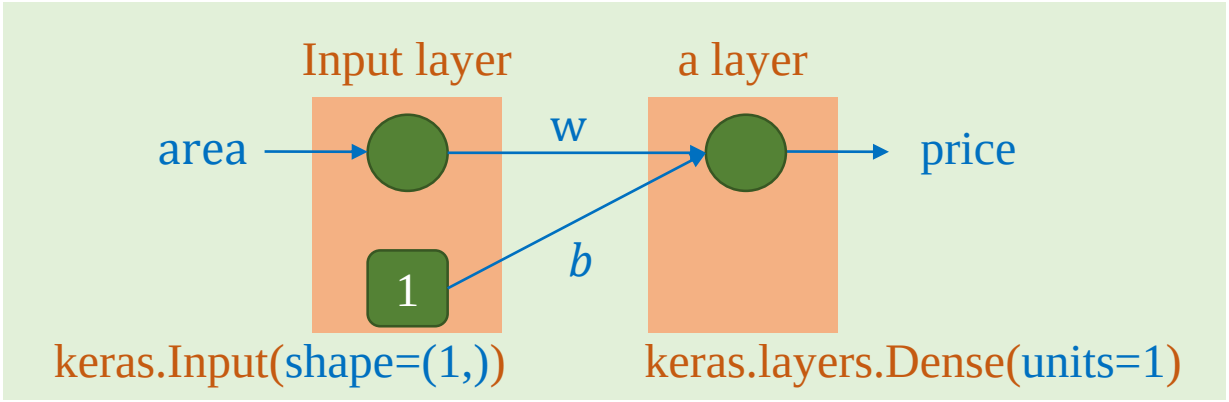
Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

House price data

Model

$$\text{price} = w * area + b$$

$$y = wx + b$$




```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(1,)))
model.add(keras.layers.Dense(1))

model.summary()
```

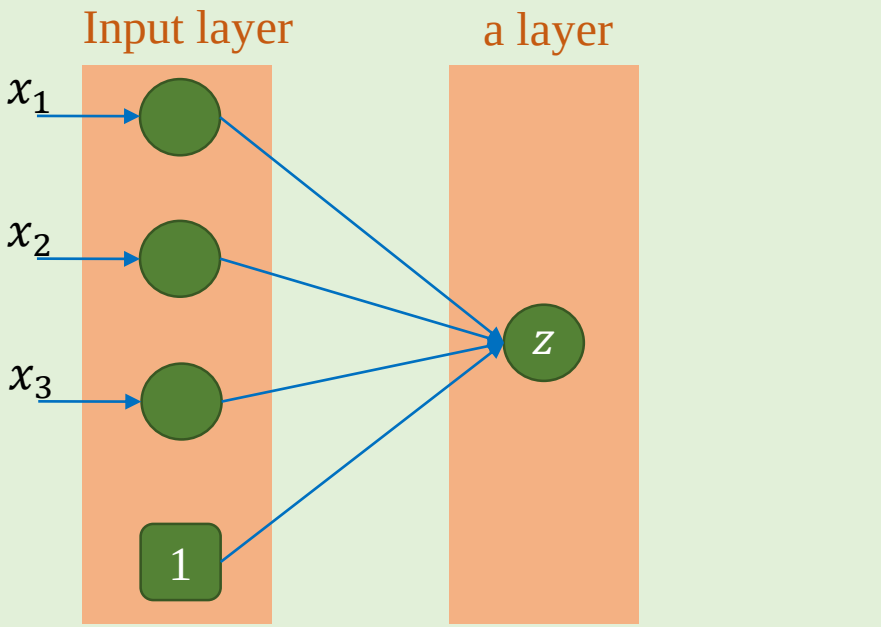
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

Model Construction

❖ Linear regression

Features			Label
TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9

Advertising-based sale data



Model

$$\text{Sale} = w_1 * TV + w_2 * Radio + w_3 * Newspaper + b$$

$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

```
1  import tensorflow as tf
2  import tensorflow.keras as keras
3
4  # create model
5  model = keras.Sequential()
6  model.add(keras.Input(shape=(3,)))
7  model.add(keras.layers.Dense(1))
8
9  model.summary()
```

Model Construction

❖ Linear regression

Boston House Price Data

Features | Label

crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9

Model

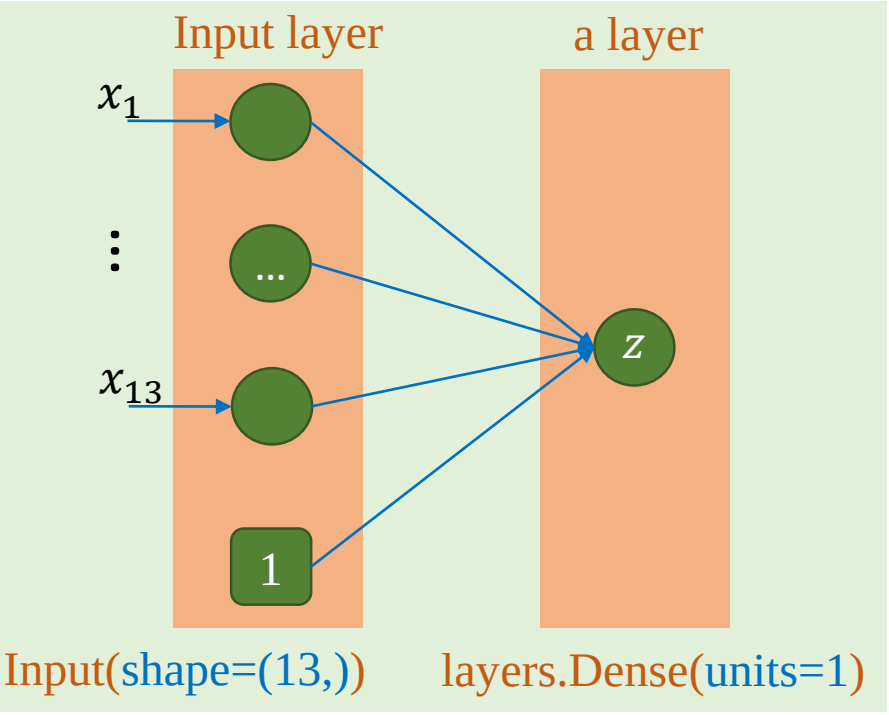
$$\text{medv} = w_1 * x_1 + \cdots + w_{13} * x_{13} + b$$

Model Construction

❖ Linear regression

Model

$$\text{medv} = w_1 * x_1 + \cdots + w_{13} * x_{13} + b$$




```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(13,)))
model.add(keras.layers.Dense(1))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1)                 14
=================================================================
Total params: 14
Trainable params: 14
Non-trainable params: 0
_________________________________________________________________
```

Outline

- Introduction to Tensorflow
- Tensorflow/Keras
- Model Construction – Linear Regression
- Model Construction – Logistic Regression
- Model Construction – Softmax Regression
- Model Training
- Model Saving and Loading

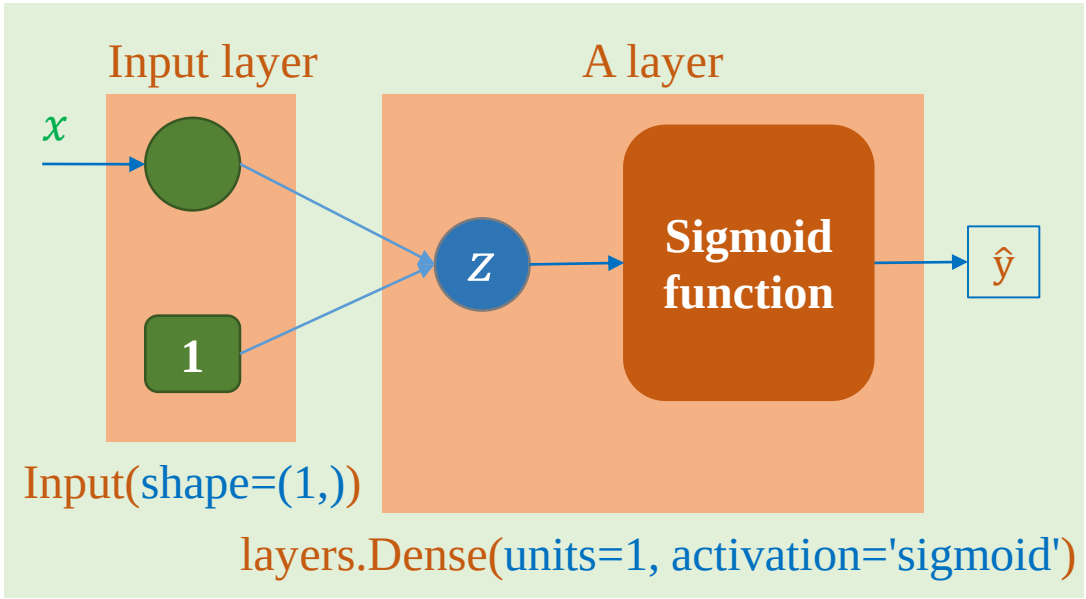
Model Construction

Feature	Label
Petal_Length	Category
1.4	0
1	0
1.5	0
3	1
3.8	1
4.1	1

Model

$$z = wx + b$$

$$\hat{y} = \frac{1}{1 + e^{-z}}$$



❖ Logistic regression

```
1  import tensorflow as tf
2  import tensorflow.keras as keras
3
4  # create model
5  model = keras.Sequential()
6  model.add(keras.Input(shape=(1,)))
7  model.add(keras.layers.Dense(1, activation='sigmoid'))
8
9  model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

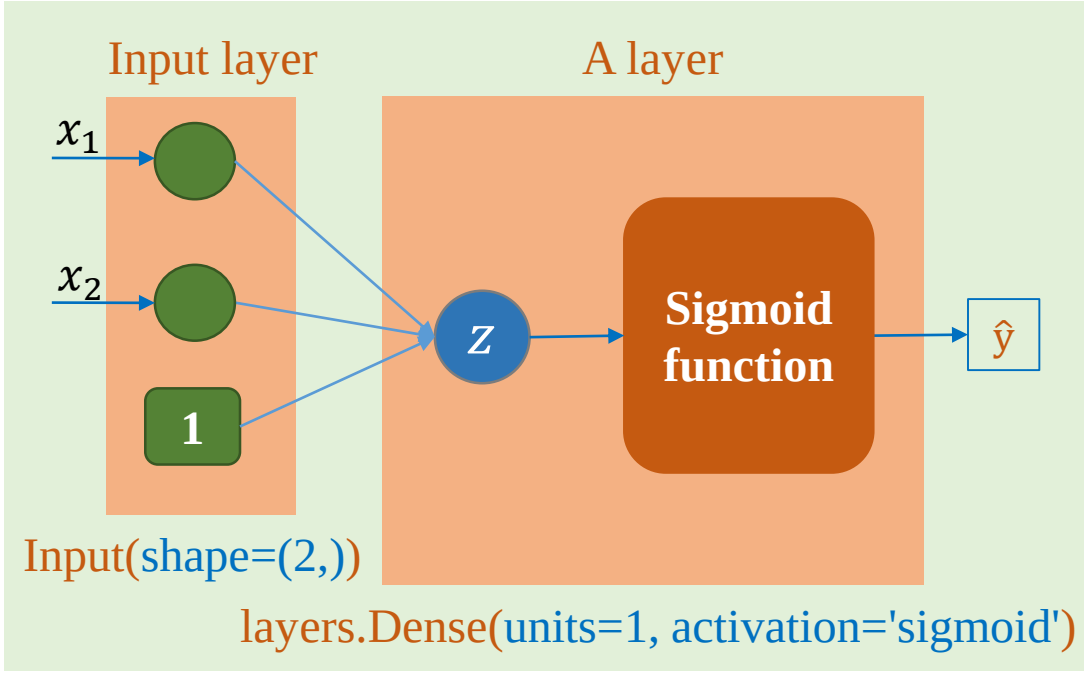
Model Construction

Feature		Label
Petal_Length	Petal_Width	Label
1.5	0.2	0
1.4	0.2	0
1.6	0.2	0
4.7	1.6	1
3.3	1.1	1
4.6	1.3	1

Model

$$z = \theta^T x$$

$$\hat{y} = \frac{1}{1 + e^{-z}}$$



❖ Logistic regression

```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(2,)))
model.add(keras.layers.Dense(1, activation='sigmoid'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 3
=================================================================
Total params: 3
Trainable params: 3
Non-trainable params: 0
_________________________________________________________________
```

AI VIETNAM
Free AI Course

Model Construction

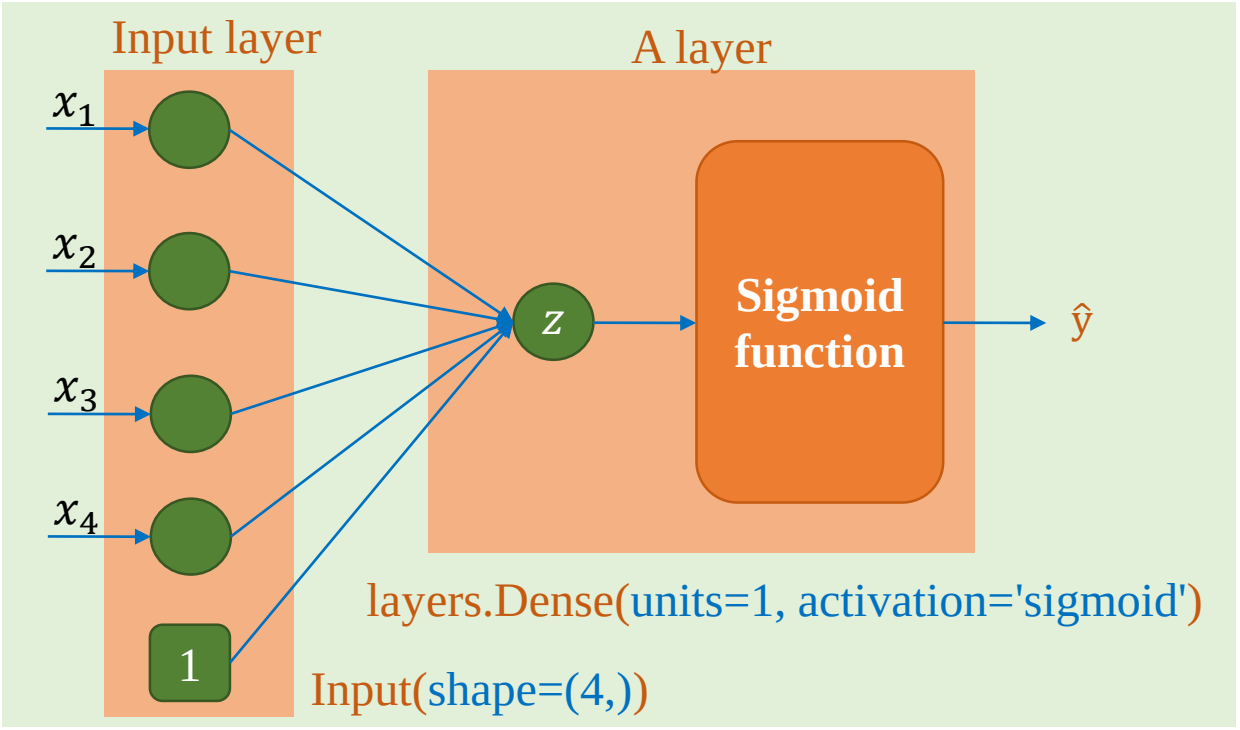
Feature				Label
Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Label
5.2	3.5	1.5	0.2	0
5.2	3.4	1.4	0.2	0
4.7	3.2	1.6	0.2	0
6.3	3.3	4.7	1.6	1
4.9	2.4	3.3	1.1	1
6.6	2.9	4.6	1.3	1

Model

$$z = \theta^T x$$

$$\hat{y} = \frac{1}{1 + e^{-z}}$$

❖ Logistic regression




```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(4,)))
model.add(keras.layers.Dense(1, activation='sigmoid'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 5
=================================================================
Total params: 5
Trainable params: 5
Non-trainable params: 0
_________________________________________________________________
```

Outline

- Introduction to Tensorflow
- Tensorflow/Keras
- Model Construction – Linear Regression
- Model Construction – Logistic Regression
- Model Construction – Softmax Regression
- Model Training
- Model Saving and Loading

AI VIETNAM
Free AI Course

Model Construction

Feature	Label
Petal_Length	Label
1.4	1
1.3	1
1.5	1
4.5	2
4.1	2
4.6	2

Iris Classification Data

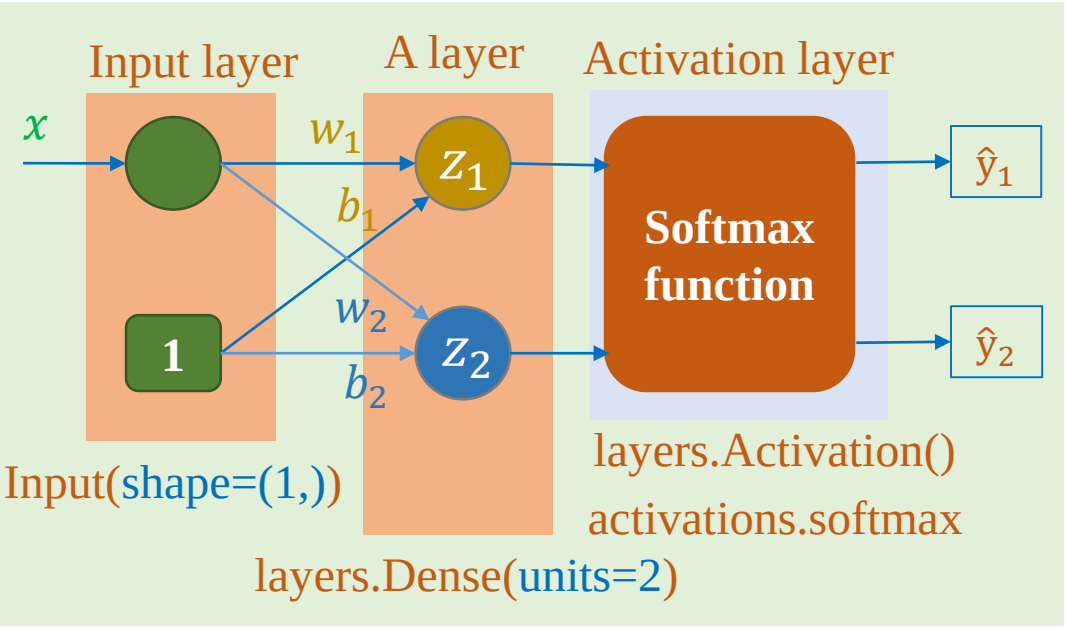
$$z_1 = xw_1 + b_1$$

$$z_2 = xw_2 + b_2$$

$$\hat{y}_1 = \frac{e^{z_1}}{\sum_{j=1}^{2} e^{z_j}}$$

$$\hat{y}_2 = \frac{e^{z_1}}{\sum_{j=1}^{2} e^{z_j}}$$

❖ Softmax regression




```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(1,)))
model.add(keras.layers.Dense(2))
model.add(keras.layers.Activation(keras.activations.softmax))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 2)                 4
_________________________________________________________________
activation (Activation)      (None, 2)                 0
=================================================================
Total params: 4
Trainable params: 4
Non-trainable params: 0
_________________________________________________________________
```

Model Construction

Feature **Label**

Petal_Length	Label
1.4	1
1.3	1
1.5	1
4.5	2
4.1	2
4.6	2

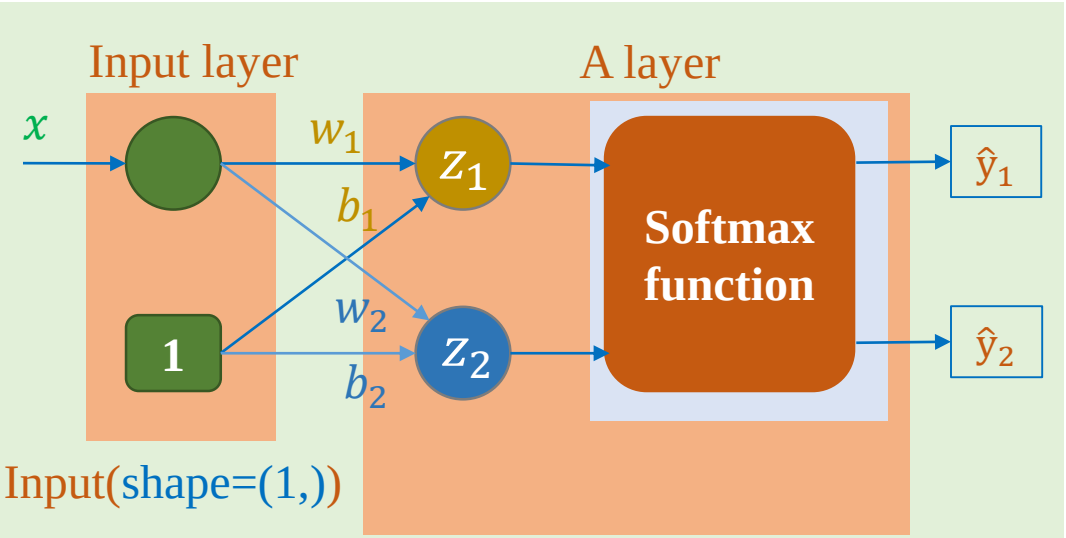
Iris Classification Data

$$z_1 = xw_1 + b_1$$

$$z_2 = xw_2 + b_2$$

$$\hat{y}_1 = \frac{e^{z_1}}{\sum_{j=1}^{2} e^{z_j}}$$

$$\hat{y}_2 = \frac{e^{z_1}}{\sum_{j=1}^{2} e^{z_j}}$$



❖ Softmax regression

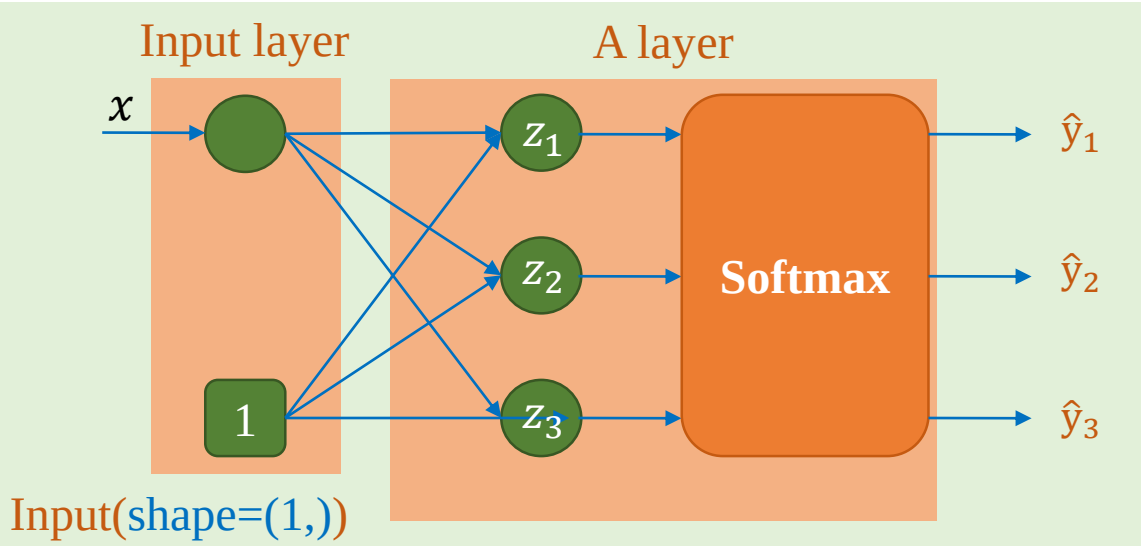
```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(1,)))
model.add(keras.layers.Dense(2, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 2)                 4
=================================================================
Total params: 4
Trainable params: 4
Non-trainable params: 0
_________________________________________________________________
```

Model Construction

❖ Softmax regression

Petal_Length	Label
1.4	1
1.3	1
1.5	1
4.5	2
4.1	2
4.6	2
5.2	3
5.6	3
5.9	3



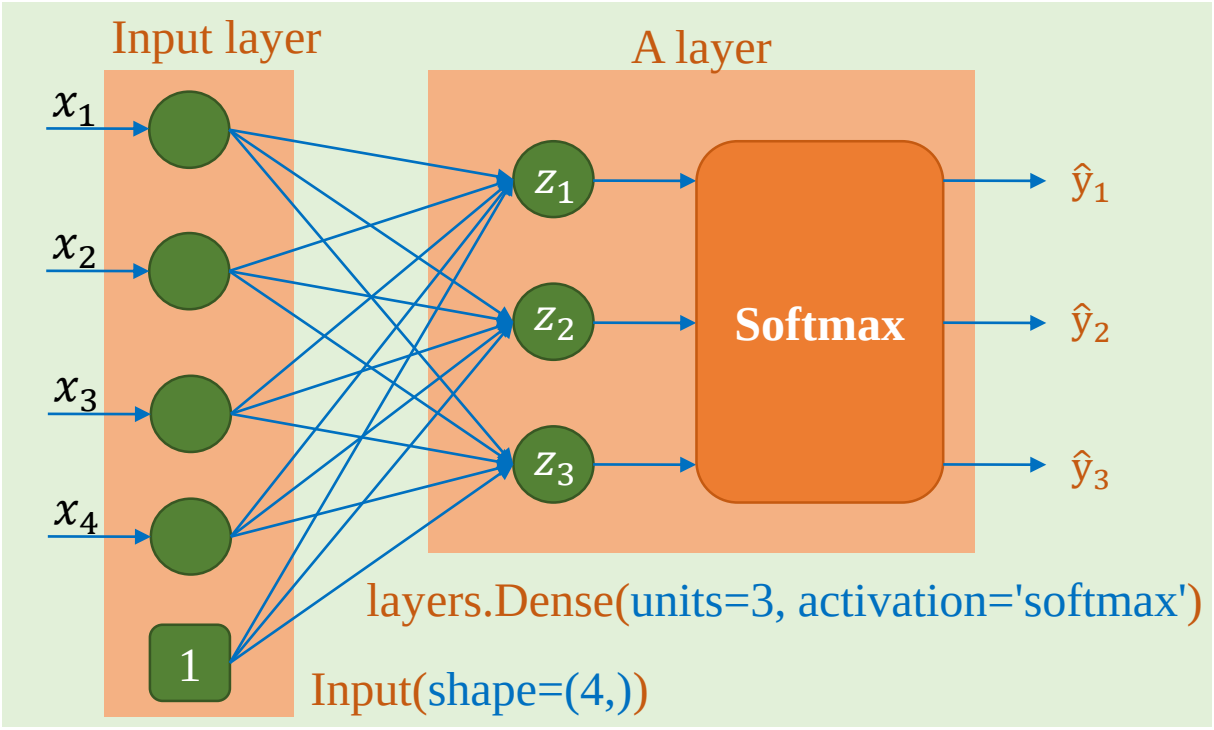

```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(1,)))
model.add(keras.layers.Dense(3, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 3)                 6
=================================================================
Total params: 6
Trainable params: 6
Non-trainable params: 0
_________________________________________________________________
```

AI VIETNAM
Free AI Course

Model Construction

Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Label
5.2	3.5	1.5	0.2	1
5.2	3.4	1.4	0.2	1
4.7	3.2	1.6	0.2	1
6.3	3.3	4.7	1.6	2
4.9	2.4	3.3	1.1	2
6.6	2.9	4.6	1.3	2
6.4	2.8	5.6	2.2	3
6.3	2.8	5.1	1.5	3
6.1	2.6	5.6	1.4	3



❖ Softmax regression

Forward computation

$$z = \theta^T x \qquad \hat{y} = \frac{e^z}{\sum_{i=1}^{k} e^{z_i}}$$

```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = keras.Sequential()
model.add(keras.Input(shape=(4,)))
model.add(keras.layers.Dense(3, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 3)                 15
=================================================================
Total params: 15
Trainable params: 15
Non-trainable params: 0
_________________________________________________________________
```

Outline

- Introduction to Tensorflow
- Tensorflow/Keras
- Model Construction – Linear Regression
- Model Construction – Logistic Regression
- Model Construction – Softmax Regression
- Model Training
- Model Saving and Loading

Training

❖ Logistic regression

→ Tính output $\hat{y}$

$$z = \boldsymbol{\theta}^T \boldsymbol{x}$$
$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

→ Tính loss (binary cross-entropy)

$$L(\boldsymbol{\theta}) = \left(-\mathrm{y}^{\mathrm{T}}\log\hat{\mathrm{y}} - (1-\mathrm{y})^{\mathrm{T}}\log(1-\hat{\mathrm{y}})\right)$$

→ Tính đạo hàm

$$L'_{\boldsymbol{\theta}} = \mathrm{x}^{\mathrm{T}}(\hat{\mathrm{y}} - y)$$

→ Cập nhật tham số (Stochastic gradient descent)

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta L'_{\boldsymbol{\theta}}$$

Computed automatically

Declare optimizer and loss function

```
model.compile(optimizer='sgd',
              loss='binary_crossentropy')
```

Start training

```
model.fit(x-data, y-data, batch-size, epochs)
```

If batch-size=1 → Stochastic training

If batch-size=N → Batch training

If 1<batch-size<N → Mini-batch training

Training

❖ Softmax regression

→ Tính output $\hat{y}$

$$z = \theta^T x \qquad \hat{y} = \frac{e^z}{\sum_{i=1}^{k} e^{z_i}}$$

→ Tính loss (cross-entropy)

$$L(\theta) = -\sum_{i=1}^{k} \delta(i, y) \log \hat{y}_i$$

→ Tính đạo hàm

$$\frac{\partial L}{\partial \theta_i} = x\left(\hat{y}_i - \delta(i, y)\right)$$

→ Cập nhật tham số (Stochastic gradient descent)

$$\theta = \theta - \eta L'_\theta$$

Computed automatically

Declare optimizer and loss function

```
model.compile(optimizer='sgd',
              loss='categorical_crossentropy')
```

Start training

```
model.fit(x-data, y-data, batch-size, epochs)
```

If batch-size=1 → Stochastic training

If batch-size=m → Batch training

If 1<batch-size<m → Mini-batch training

AI VIETNAM
Free AI Course

Training

❖ Linear regression

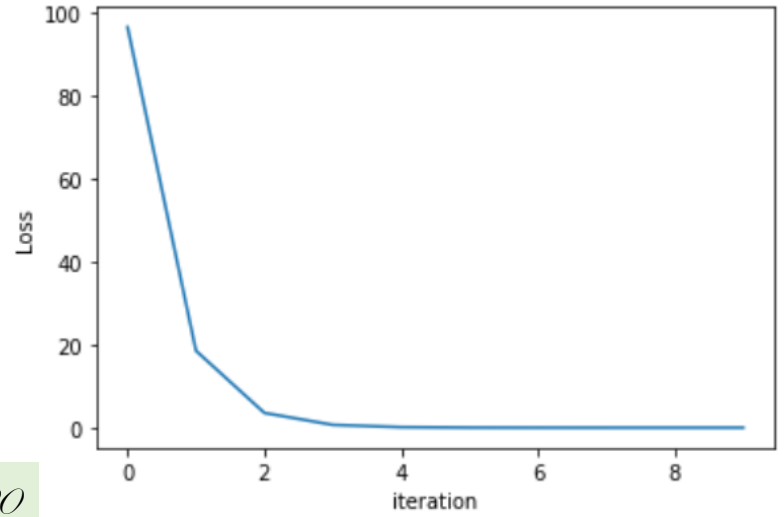
Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

House price data

Model

$$\text{price} = w * area + b$$

$$y = wx + b$$




```python
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras

batch_size = 4
epochs = 10

# Data Preparation
data = np.genfromtxt('data.csv', delimiter=',')
X = data[:,0:1]
y = data[:,1:]

# create model
model = tf.keras.Sequential(
    [tf.keras.layers.Dense(units=1, input_shape=[1])])

# declare optimization method and loss function
opt = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=opt, loss='mse')

# training
history = model.fit(X, y, batch_size, epochs)
```

```
Train on 4 samples
Epoch 1/10
4/4 [==============================] - 0s 28ms/sample - loss: 96.6016
Epoch 2/10
4/4 [==============================] - 0s 247us/sample - loss: 18.6414
Epoch 3/10
4/4 [==============================] - 0s 499us/sample - loss: 3.6692
Epoch 4/10
4/4 [==============================] - 0s 245us/sample - loss: 0.7937
Epoch 5/10
4/4 [==============================] - 0s 499us/sample - loss: 0.2415
Epoch 6/10
4/4 [==============================] - 0s 499us/sample - loss: 0.1353
```

Training

❖ Logistic regression

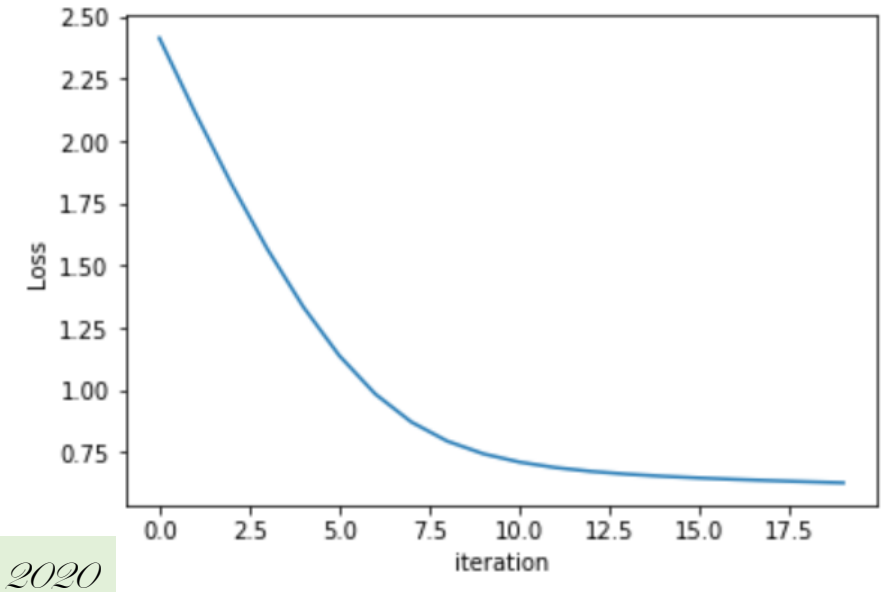
Feature **Label**

Petal_Length	Category
1.4	0
1	0
1.5	0
3	1
3.8	1
4.1	1

Model

$$z = wx + b$$

$$\hat{y} = \frac{1}{1 + e^{-z}}$$




```python
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras

batch_size = 6
epochs = 20

# Data Preparation
data = np.genfromtxt('iris_1D.csv', delimiter=',', skip_header=1)
X = data[:,0:1]
y = data[:,1]

# create model
model = tf.keras.Sequential(
    [tf.keras.layers.Dense(units=1, activation='sigmoid', input_shape=[1])])

# declare optimization method and loss function
opt = keras.optimizers.SGD(learning_rate=0.1)
model.compile(optimizer=opt, loss='binary_crossentropy')

# training
history = model.fit(X, y, batch_size, epochs)
```

```
Train on 6 samples
Epoch 1/20
6/6 [==============================] - 0s 20ms/sample - loss: 2.4125
Epoch 2/20
6/6 [==============================] - 0s 339us/sample - loss: 2.1118
Epoch 3/20
6/6 [==============================] - 0s 326us/sample - loss: 1.8278
Epoch 4/20
6/6 [==============================] - 0s 161us/sample - loss: 1.5661
Epoch 5/20
6/6 [==============================] - 0s 333us/sample - loss: 1.3337
Epoch 6/20
6/6 [==============================] - 0s 166us/sample - loss: 1.1381
Epoch 7/20
6/6 [==============================] - 0s 160us/sample - loss: 0.9842
```

Training

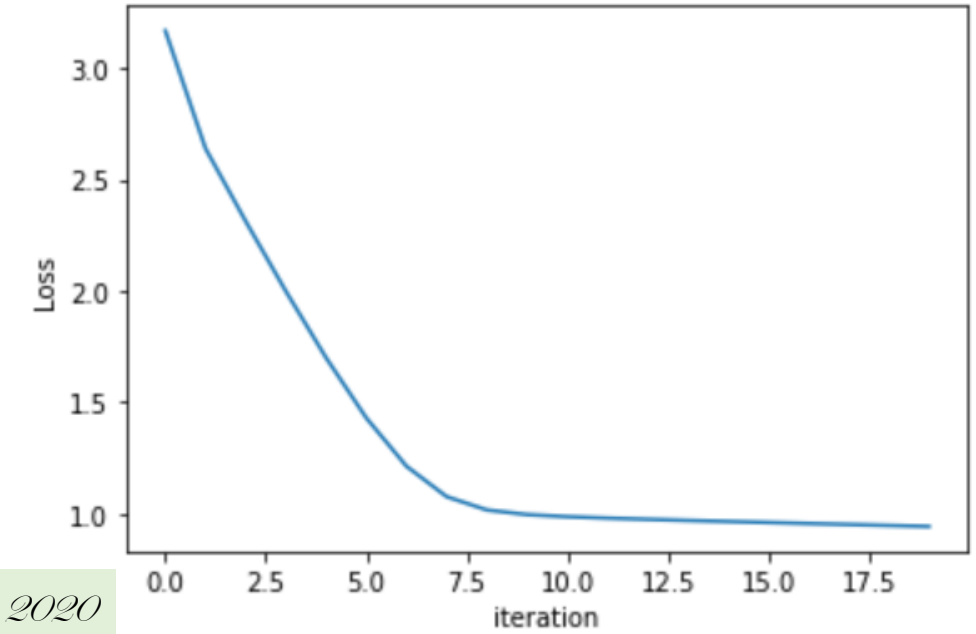
Petal_Length	Label
1.4	1
1.3	1
1.5	1
4.5	2
4.1	2
4.6	2
5.2	3
5.6	3
5.9	3

❖ Softmax regression

Model

$$z = \theta^T x$$

$$\hat{y} = \frac{e^z}{\sum_{i=1}^{k} e^{z_i}}$$




```python
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras

batch_size = 9
epochs = 20

# Data Preparation
data = np.genfromtxt('iris_1D_3c.csv', delimiter=',', skip_header=1)
X = data[:,0:1]
y = data[:,1]

# create model
model = tf.keras.Sequential(
    [tf.keras.layers.Dense(units=3, activation='softmax', input_shape=[1])])

# declare optimization method and loss function
opt = keras.optimizers.SGD(learning_rate=0.1)
model.compile(optimizer=opt, loss='sparse_categorical_crossentropy')

# training
history = model.fit(X, y, batch_size, epochs)
```

```
Train on 9 samples
Epoch 1/20
9/9 [==============================] - 0s 19ms/sample - loss: 3.1676
Epoch 2/20
9/9 [==============================] - 0s 221us/sample - loss: 2.6389
Epoch 3/20
9/9 [==============================] - 0s 110us/sample - loss: 2.3121
Epoch 4/20
9/9 [==============================] - 0s 221us/sample - loss: 1.9978
Epoch 5/20
9/9 [==============================] - 0s 111us/sample - loss: 1.6993
Epoch 6/20
9/9 [==============================] - 0s 220us/sample - loss: 1.4296
Epoch 7/20
9/9 [==============================] - 0s 222us/sample - loss: 1.2127
Epoch 8/20
9/9 [==============================] - 0s 221us/sample - loss: 1.0761
Epoch 9/20
9/9 [==============================] - 0s 222us/sample - loss: 1.0165
Epoch 10/20
9/9 [==============================] - 0s 222us/sample - loss: 0.9956
```

Outline

- Introduction to Tensorflow
- Tensorflow/Keras
- Model Construction – Linear Regression
- Model Construction – Logistic Regression
- Model Construction – Softmax Regression
- Model Training
- Model Saving and Loading

Model Saving and Loading

Model Saving

```python
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras

batch_size = 4
epochs = 10

# Data Preparation
data = np.genfromtxt('data.csv', delimiter=',')
X = data[:,0:1]
y = data[:,1:]

# create model
model = tf.keras.Sequential(
    [tf.keras.layers.Dense(units=1, input_shape=[1])])

# declare optimization method and loss function
opt = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=opt, loss='mse')

# training
history = model.fit(X, y, batch_size, epochs)

# save model
checkpoint_path = "my_model/model.ckpt"
model.save_weights(checkpoint_path)
```

Testing

```python
# testing
X_testing = [[5.0]]
y_hat = model.predict(X_testing)
print(y_hat)
```

```
[[6.51236]]
```

Model Loading

```python
import tensorflow as tf
import tensorflow.keras as keras

# create model
model = tf.keras.Sequential(
    [tf.keras.layers.Dense(units=1, input_shape=[1])])

# load model
model.load_weights('my_model/model.ckpt')

X_testing = [[5.0]]
y_hat = model.predict(X_testing)
print(y_hat)
```

```
[[6.5058403]]
```

Model Saving and Loading

Model Saving

```python
import numpy as np
import tensorflow as tf
import tensorflow.keras as keras

batch_size = 4
epochs = 10

# Data Preparation
data = np.genfromtxt('data.csv', delimiter=',')
X = data[:,0:1]
y = data[:,1:]

# create model
model = tf.keras.Sequential(
    [tf.keras.layers.Dense(units=1, input_shape=[1])])

# declare optimization method and loss function
opt = keras.optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=opt, loss='mse')

# training
history = model.fit(X, y, batch_size, epochs)

# save entire model
model.save('my_model/model.h5')
```

Model Loading

```python
import tensorflow as tf
import tensorflow.keras as keras

# load model
model = tf.keras.models.load_model('my_model/model.h5')

# testing
X_testing = [[5.0]]
y_hat = model.predict(X_testing)
print(y_hat)
```

```
[[6.5115185]]
```

Tensorflow

❖ Demo

```
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] ::
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>> for epoch in range(n_epochs):
...     sum_of_losses = 0
...     gradients = np.zeros((2,1))
...
...     for index in range(4):
...         xi = X_b[index:index+1]
...         yi = y[index:index+1]
```

Reference

Tensor

https://www.tensorflow.org/guide/tensor

TensorFlow 2 quickstart for beginners

https://www.tensorflow.org/tutorials/quickstart/beginner

Save and load models

https://www.tensorflow.org/tutorials/keras/save_and_load

Gradient tape

https://www.tensorflow.org/guide/autodiff


Thank you!