Introduction to TensorFlow 2.0

Agenda

- Keras API
  - Sequential model
  - Functional API
- Dataset API
- Low-level API

Example: quadratic regression

```
import numpy as np
n = 100
x = np.random.uniform(-2, 2, n)
y = 3 + 2*x + x**2 + np.random.normal(0,1,n)
X = np.column_stack((x,x**2))
```

```
import tensorflow as tf
print(tf.__version__)
```

```
2.3.0
```

Sequential model

- The easiest way to use TensorFlow 2.x

- Method 1:
  model = tf.keras.Sequential(name='')
  model.add(layer.xxx(..., input_shape=...))
  ...

- Method 2:
  model = tf.keras.Sequential([
  ...])

```
from tensorflow.keras import layers
from tensorflow.keras import regularizers
n, p = X.shape
model = tf.keras.Sequential(name='quadraticreg')
model.add(layers.Dense(1, activation='linear',
          kernel_regularizer=regularizers.l2(0.001),
          input_shape=(p,)))
model.summary()
```

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

Compile the model

- Configures the model for training: model.compile(optimizer, loss, metric, ...)

```
model.compile(optimizer='sgd',
              loss=tf.keras.losses.MeanSquaredError())
```

Training

- Trains the model for a fixed number of epochs by model.fit:
  model.fit(x, y, batch_size, epoch, ...)

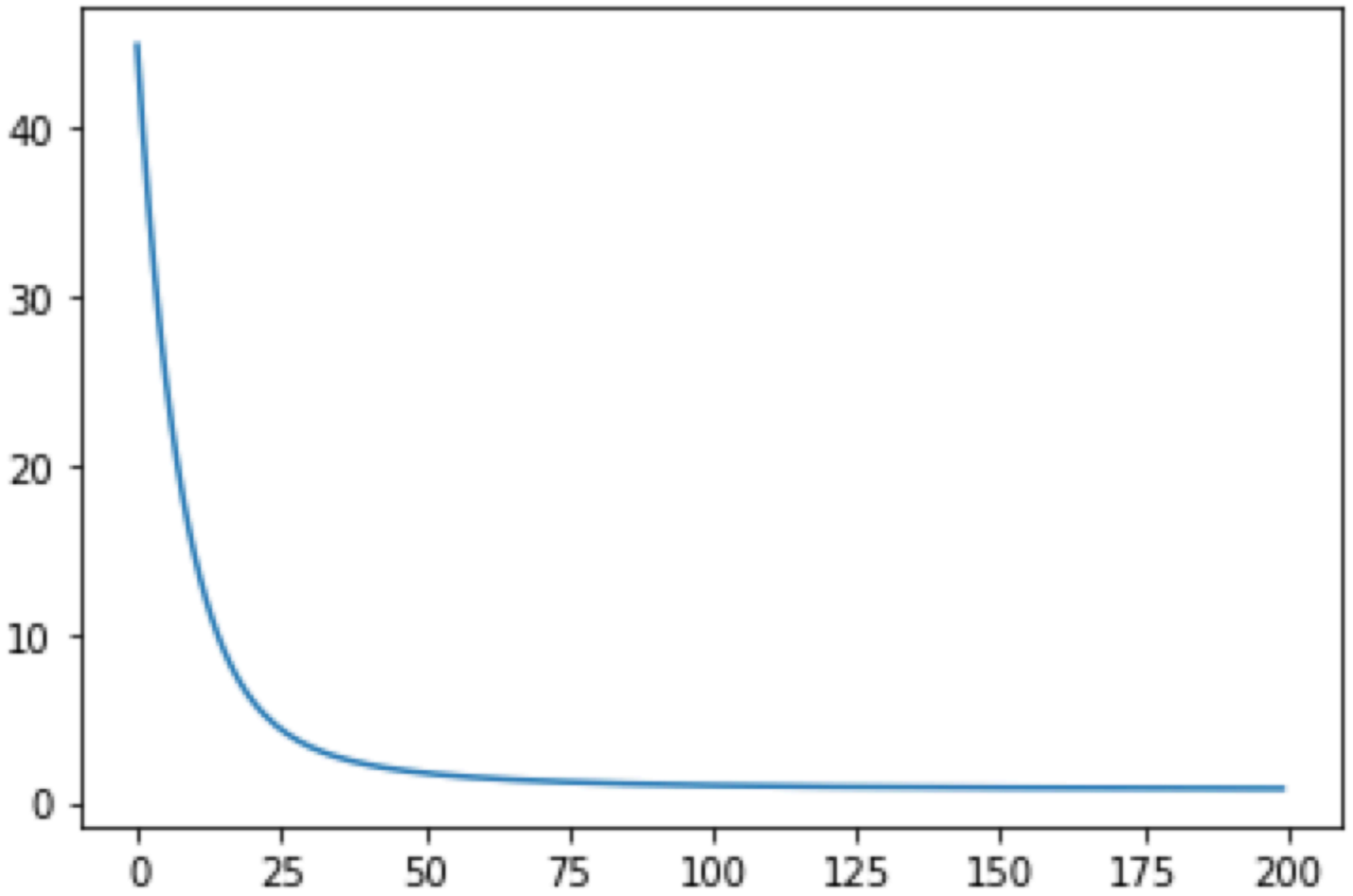
```
history = model.fit(X, y, batch_size=n, epochs=200, verbose=0)
```

```
print(history.history.keys())
```

```
dict_keys(['loss'])
```

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
```

```
[<matplotlib.lines.Line2D at 0x1426f9d00>]
```



Evaluation

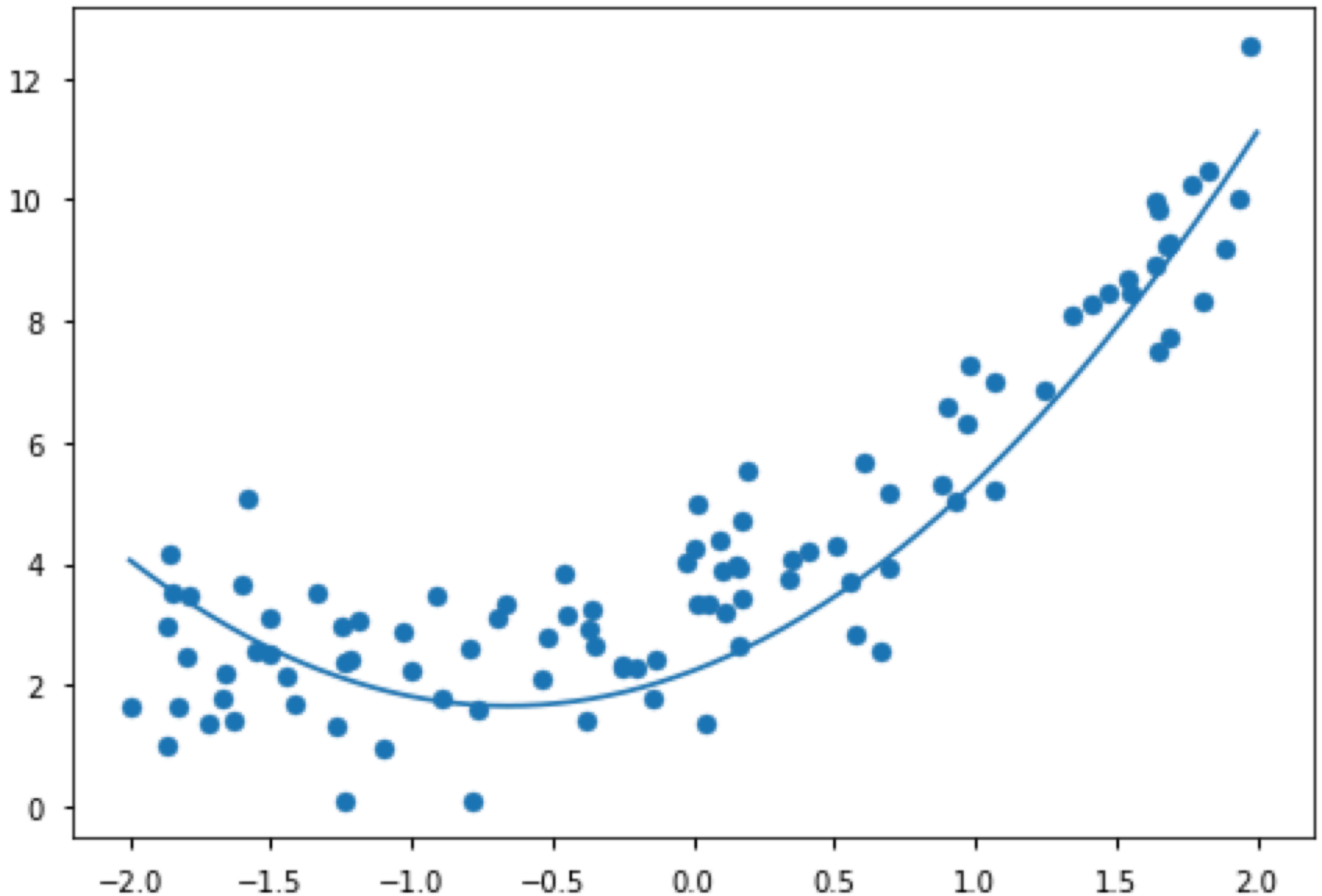
- Returns the loss value & metrics values for the model:
  model.evaluate(x, y, ...)

Predictions

- Make predictions by model.predict(): model.predict(x, ...)

```
xeval = np.linspace(-2,2,401)
Xeval = np.column_stack((xeval,xeval**2))
yeval = model.predict(Xeval)
```

```
import matplotlib.pyplot as plt
plt.style.use('seaborn-notebook')
plt.scatter(x, y)
plt.plot(xeval, yeval)
plt.show()
```



References

- 輕鬆學會 Google TensorFlow 2.0 人工智慧深度學習實作開發 (GitHub)

- Official documentation for tf.keras

Homework: binary logistic regression by tf.keras

重做上次作業，但改用tf.keras自行實做binary logistic regression(可自行嘗試各種不同網路結構)。Hints:

1. output layer不需要加activation function
2. loss function使用tf.keras.losses中的 BinaryCrossentropy，並將from_logits設為true