Introduction to TensorFlow

Prerequisites for Building Models with TensorFlow

郭耀仁

大綱

- 機器學習基礎
- Loss Functions
- 梯度遞減
- Name Scopes
- Optimizers
- Summaries for TensorBoard
- Lazy Loading

機器學習基礎

常見入門途徑有兩種

- 從理論入門
- 從工具（Scikit-Learn、TensorFlow、PyTorch...等）入門

從理論入門

- Andrew Ng: Machine Learning (https://www.coursera.org/learn/machine-learning)
- 林軒田： 機器學習基石 (https://www.coursera.org/learn/ntumlone-mathematicalfoundations/)
- Deep Learning (https://www.deeplearningbook.org/)
- Introduction to Statistical Learning (http://www-bcf.usc.edu/~gareth/ISL/)

從工具入門

- Scikit-Learn (https://scikit-learn.org/stable/)
- Google Machine Learning Crash Course (https://developers.google.com/machine-learning/crash-course/ml-intro)
- fast.ai (https://www.fast.ai/)

Scikit-Learn 是學習門檻較低的選項

先關注 Scikit-Learn 的這三個模組

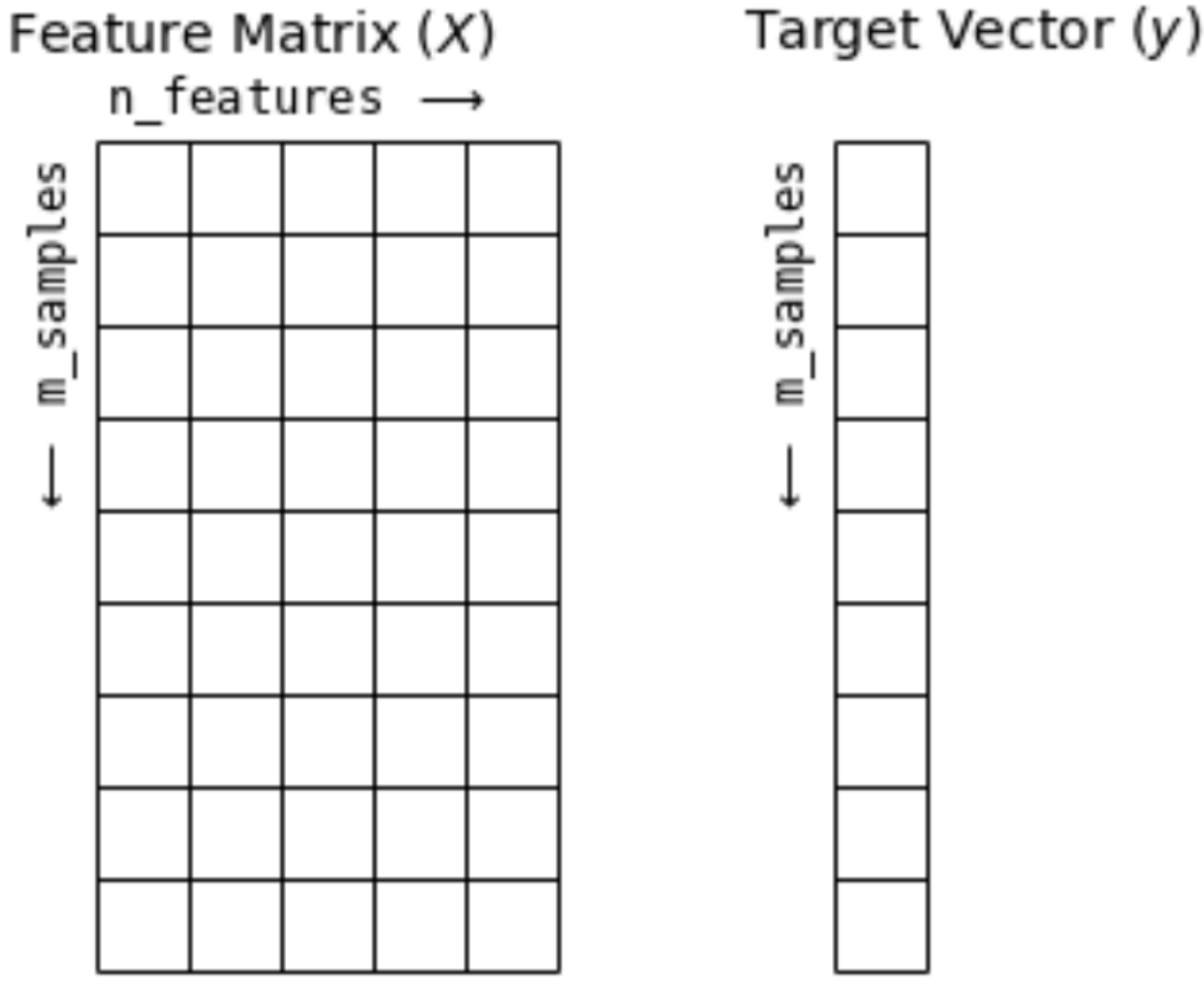
- Preprocessing
- Supervised Learning
    - Classification
    - Regression
- Model Selection

瞭解資料的長相

- Feature Matrix: 常用 X 代表
- Target Vector: 常用 y 代表

通常我們讀入的資料整合 X 與 y

- 讀入資料外觀 (m, n+1)：m 個觀測值、n+1 個特徵
    - y：(m, 1)
    - X：(m, n)


Feature Matrix (X)
n_features ⟶
⟵ m_samples
Target Vector (y)
⟵ m_samples

Loss Functions

有時也被稱為成本函數（Cost Functions）

- 每個學習模型在訓練過程的目標都相同：**想辦法找到一組讓 Loss Function 最小化的係數**
    - 線性迴歸：讓 SSE 最低
    - 羅吉斯迴歸：讓誤分類數最低
    - 決策樹：讓 cross-entropy 最低
    - ...etc.

梯度遞減

定義好 Loss Function 以後呢?

是找出真實、能讓 Loss Function 為 0 的$f(x)$ 函數嗎?

$$f(x)$$

定義好 Loss Function 以後

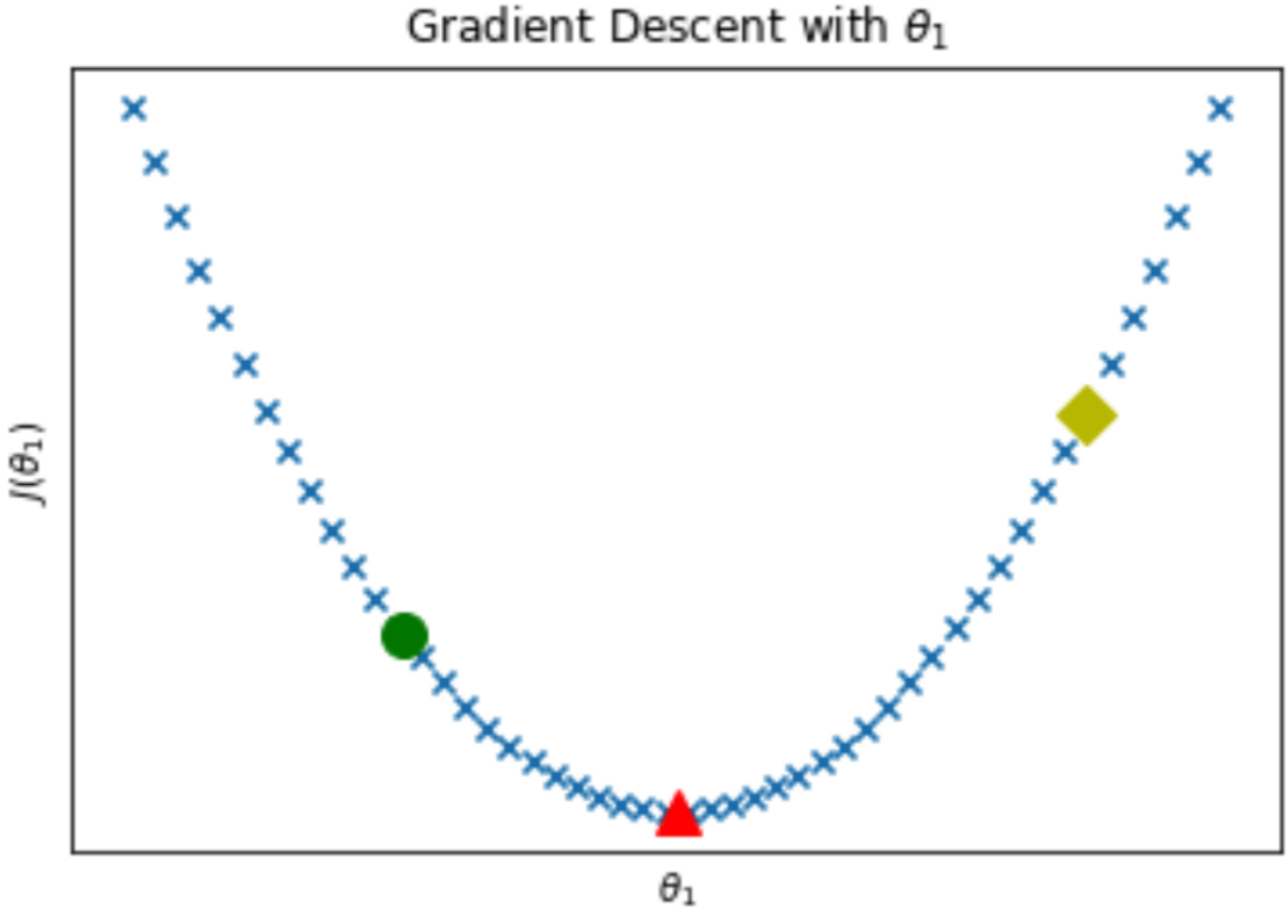
其實是找出能夠讓 Loss Function 很小、長得跟 $f(x)$ 函數很像很像的 $h(x)$ 函數！

$$h(x) = XW + b$$

那又該如何著手?

千里之行，始於足下。

但是要有技巧地往對的方向移動



計算 Loss Function 的斜率（梯度）

$$\Delta W = \frac{\partial L}{\partial W}$$

找到正確的方向，然後用適當的速度前進

$$W := W - \alpha \Delta W$$

Name Scopes

一個 TensorFlow 計算圖形（Graph）中會有

- 變數張量
- Placeholder 張量
- 運算節點

當模型變得複雜，TensorBoard 就會顯得凌亂

- 使用 `tf.name_scope()` 來解決
- 會在每個張量的命名前面加入 Name Space 的名稱

In [1]:

```
import tensorflow as tf

with tf.name_scope("constants"):
  lucky_number_const = tf.constant(24)
  stupid_number_const = tf.constant(87)

with tf.name_scope("variables"):
  lucky_number_var = tf.Variable(24)
  stupid_number_var = tf.Variable(87)

with tf.name_scope("placeholders"):
  X = tf.placeholder(tf.float32, (5, 1))
  y = tf.placeholder(tf.float32, (5,))

print(lucky_number_const)
print(stupid_number_const)
print(lucky_number_var)
print(stupid_number_var)
print(X)
print(y)
```

```
Tensor("constants/Const:0", shape=(), dtype=int32)
Tensor("constants/Const_1:0", shape=(), dtype=int32)
<tf.Variable 'variables/Variable:0' shape=() dtype=int32_ref>
<tf.Variable 'variables/Variable_1:0' shape=() dtype=int32_ref>
Tensor("placeholders/Placeholder:0", shape=(5, 1), dtype=float32)
Tensor("placeholders/Placeholder_1:0", shape=(5,), dtype=float32)
```

Optimizers

到目前訓練模型還缺什麼?

- Placeholders: 餵入資料點
- Loss functions
- 梯度遞減

我們不知道該怎麼樣讓 TensorFlow 開始做梯度遞減

透過 `tf.train.XXX()` 啟動

- `tf.train.GradientDescentOptimizer()`
- `tf.train.AdamOptimizer()`
- …etc.

```
# pseudo code
learning_rate = .001
with tf.name_scope("optimizer"):
  train_op = tf.train.AdamOptimizer(learning_rate).minimize(l) # l as in some los
s function
```

Summaries for TensorBoard

先前我們僅使用 graph 頁籤

- Graph 頁籤：觀察計算圖
- Scalar 頁籤：觀察 loss function 是否隨著訓練次數增加而遞減

以 tf.summary 將 Scalar 與 Graph 整合至 TensorBoard

```
with tf.name_scope("summaries"):
  tf.summary.scalar("loss", l)
  merged = tf.summary.merge_all()

train_writer = tf.summary.FileWriter('/graphs/tmp', tf.get_default_graph())
```

Lazy Loading

何謂 Lazy Loading

- Normal Loading: 在執行 Session 之前將運算（節點）定義完畢
- Lazy Loading: 在 Session 中將運算（節點）輸入

In [2]:
```
tf.reset_default_graph()
```

In [3]:

```
# Normal Loading
x = tf.Variable(10, name='x')
y = tf.Variable(20, name='y')
z = tf.add(x, y) # check point!

writer = tf.summary.FileWriter('./graphs/normal_loading', tf.get_default_graph())
with tf.Session() as sess:
  sess.run(x.initializer)
  sess.run(y.initializer)
  for _ in range(10):
    sess.run(z)
writer.close()
```

In [4]:
```
tf.get_default_graph().as_graph_def()
```

Out[4]:
```
node {
  name: "x/initial_value"
  op: "Const"
  attr {
    key: "dtype"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "value"
    value {
      tensor {
        dtype: DT_INT32
        tensor_shape {
        }
        int_val: 10
      }
    }
  }
}
node {
  name: "x"
  op: "VariableV2"
  attr {
    key: "container"
    value {
      s: ""
    }
  }
  attr {
    key: "dtype"
    value {
      type: DT_INT32
```

```
    }
  }
  attr {
    key: "shape"
    value {
      shape {
      }
    }
  }
  attr {
    key: "shared_name"
    value {
      s: ""
    }
  }
}
node {
  name: "x/Assign"
  op: "Assign"
  input: "x"
  input: "x/initial_value"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "_class"
    value {
      list {
        s: "loc:@x"
      }
    }
  }
  attr {
    key: "use_locking"
    value {
```

```
      b: true
    }
  }
  attr {
    key: "validate_shape"
    value {
      b: true
    }
  }
}
node {
  name: "x/read"
  op: "Identity"
  input: "x"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "_class"
    value {
      list {
        s: "loc:@x"
      }
    }
  }
}
node {
  name: "y/initial_value"
  op: "Const"
  attr {
    key: "dtype"
    value {
      type: DT_INT32
    }
  }
```

```
  attr {
    key: "value"
    value {
      tensor {
        dtype: DT_INT32
        tensor_shape {
        }
        int_val: 20
      }
    }
  }
}
node {
  name: "y"
  op: "VariableV2"
  attr {
    key: "container"
    value {
      s: ""
    }
  }
  attr {
    key: "dtype"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "shape"
    value {
      shape {
      }
    }
  }
  attr {
    key: "shared_name"
    value {
      s: ""
```

```
    }
  }
}
node {
  name: "y/Assign"
  op: "Assign"
  input: "y"
  input: "y/initial_value"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "_class"
    value {
      list {
        s: "loc:@y"
      }
    }
  }
  attr {
    key: "use_locking"
    value {
      b: true
    }
  }
  attr {
    key: "validate_shape"
    value {
      b: true
    }
  }
}
node {
  name: "y/read"
  op: "Identity"
```

```
  input: "y"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "_class"
    value {
      list {
        s: "loc:@y"
      }
    }
  }
}
node {
  name: "Add"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
versions {
  producer: 27
}
```

In [5]:
```
tf.reset_default_graph()
```

In [6]:

```
# Lazy Loading
x = tf.Variable(10, name='x')
y = tf.Variable(20, name='y')

writer = tf.summary.FileWriter('./graphs/lazy_loading', tf.get_default_graph())
with tf.Session() as sess:
  sess.run(x.initializer)
  sess.run(y.initializer)
  for _ in range(10):
    sess.run(tf.add(x, y)) # check point!
writer.close()
```

In [7]:
```
tf.get_default_graph().as_graph_def()
```

Out[7]:
```
node {
  name: "x/initial_value"
  op: "Const"
  attr {
    key: "dtype"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "value"
    value {
      tensor {
        dtype: DT_INT32
        tensor_shape {
        }
        int_val: 10
      }
    }
  }
}
node {
  name: "x"
  op: "VariableV2"
  attr {
    key: "container"
    value {
      s: ""
    }
  }
  attr {
    key: "dtype"
    value {
      type: DT_INT32
```

```
    }
  }
  attr {
    key: "shape"
    value {
      shape {
      }
    }
  }
  attr {
    key: "shared_name"
    value {
      s: ""
    }
  }
}
node {
  name: "x/Assign"
  op: "Assign"
  input: "x"
  input: "x/initial_value"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "_class"
    value {
      list {
        s: "loc:@x"
      }
    }
  }
  attr {
    key: "use_locking"
    value {
```

```
      b: true
    }
  }
  attr {
    key: "validate_shape"
    value {
      b: true
    }
  }
}
node {
  name: "x/read"
  op: "Identity"
  input: "x"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "_class"
    value {
      list {
        s: "loc:@x"
      }
    }
  }
}
node {
  name: "y/initial_value"
  op: "Const"
  attr {
    key: "dtype"
    value {
      type: DT_INT32
    }
  }
```

```
  attr {
    key: "value"
    value {
      tensor {
        dtype: DT_INT32
        tensor_shape {
        }
        int_val: 20
      }
    }
  }
}
node {
  name: "y"
  op: "VariableV2"
  attr {
    key: "container"
    value {
      s: ""
    }
  }
  attr {
    key: "dtype"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "shape"
    value {
      shape {
      }
    }
  }
  attr {
    key: "shared_name"
    value {
      s: ""
```

```
    }
  }
}
node {
  name: "y/Assign"
  op: "Assign"
  input: "y"
  input: "y/initial_value"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "_class"
    value {
      list {
        s: "loc:@y"
      }
    }
  }
  attr {
    key: "use_locking"
    value {
      b: true
    }
  }
  attr {
    key: "validate_shape"
    value {
      b: true
    }
  }
}
node {
  name: "y/read"
  op: "Identity"
```

```
  input: "y"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
  attr {
    key: "_class"
    value {
      list {
        s: "loc:@y"
      }
    }
  }
}
node {
  name: "Add"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
node {
  name: "Add_1"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
```

```
  }
}
node {
  name: "Add_2"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
node {
  name: "Add_3"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
node {
  name: "Add_4"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
```

```
node {
  name: "Add_5"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
node {
  name: "Add_6"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
node {
  name: "Add_7"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
node {
  name: "Add_8"
```

```
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
node {
  name: "Add_9"
  op: "Add"
  input: "x/read"
  input: "y/read"
  attr {
    key: "T"
    value {
      type: DT_INT32
    }
  }
}
versions {
  producer: 27
}
```