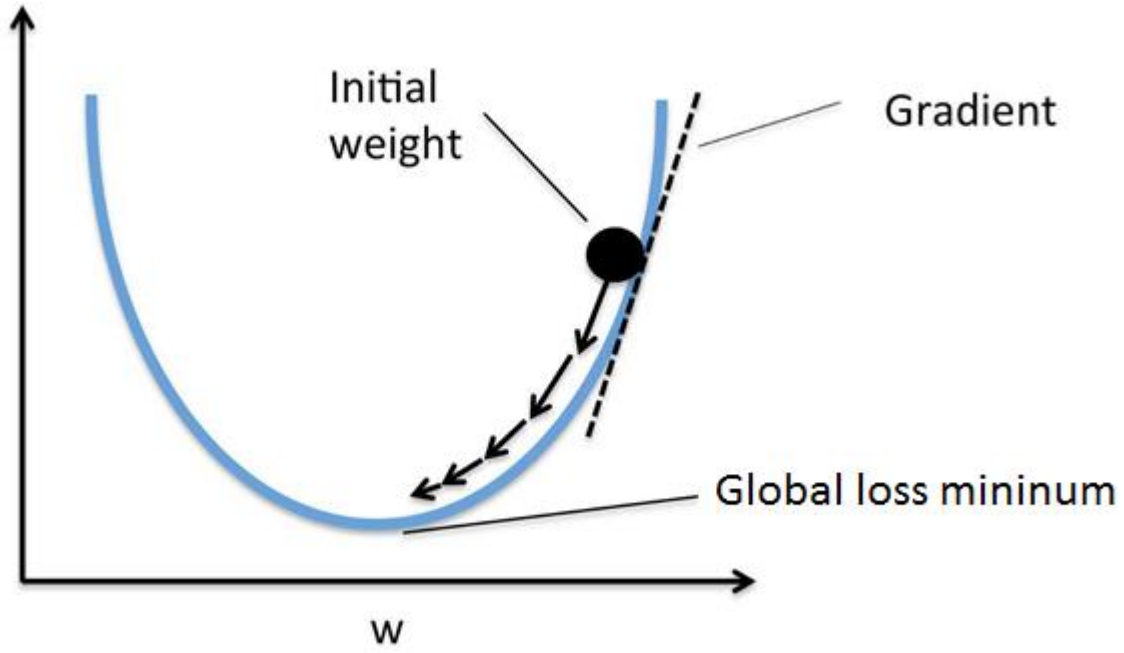
機器學習於材料資訊的應用
Machine Learning on Material Informatics

陳南佑(NAN-YOW CHEN)

楊安正(AN-CHENG YANG)

Gradient descent

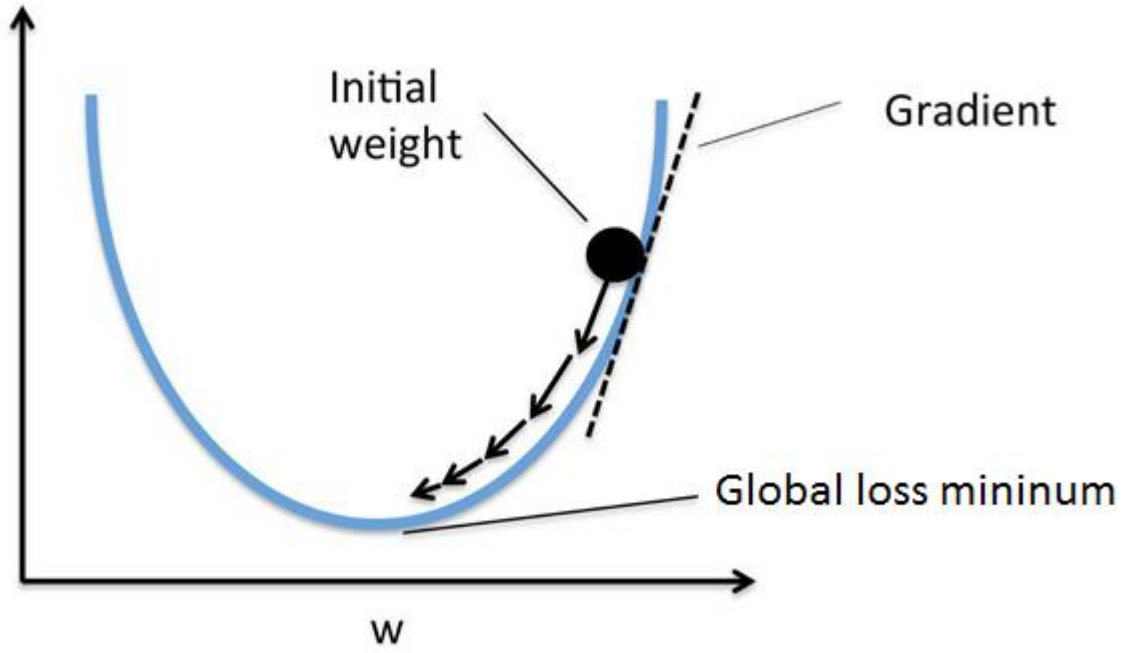


The general rule of updating weights:
SGD (Stochastic gradient descent)

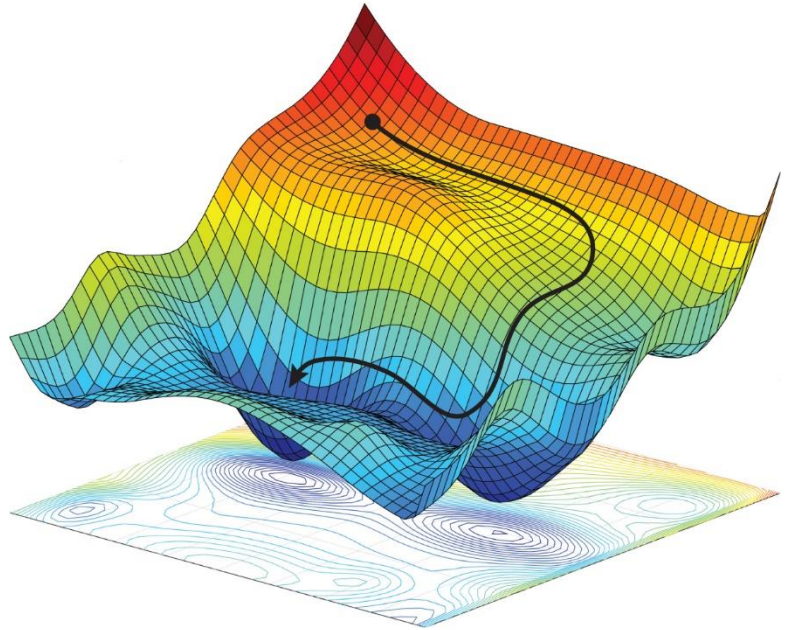
$$w' = w - \eta \times \frac{\partial Loss}{\partial w}$$

$$\eta = learning\ rate$$

Gradient descent



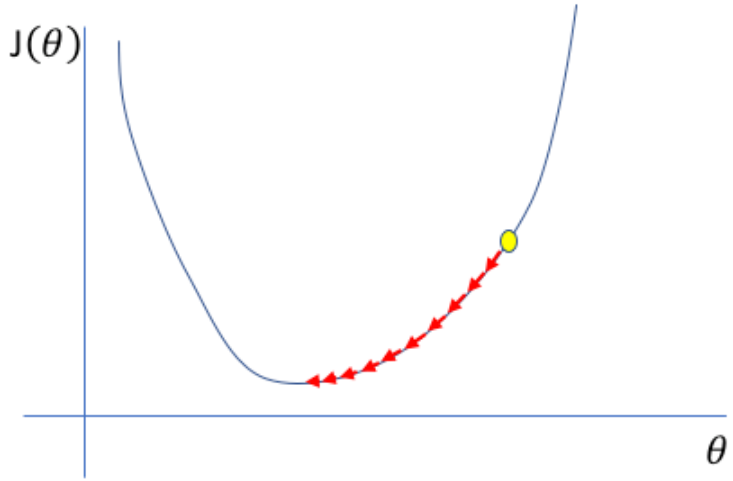
The general rule of updating weights:
SGD (Stochastic gradient descent)



http://www.its.caltech.edu/~nazizanr/papers/SMD.html

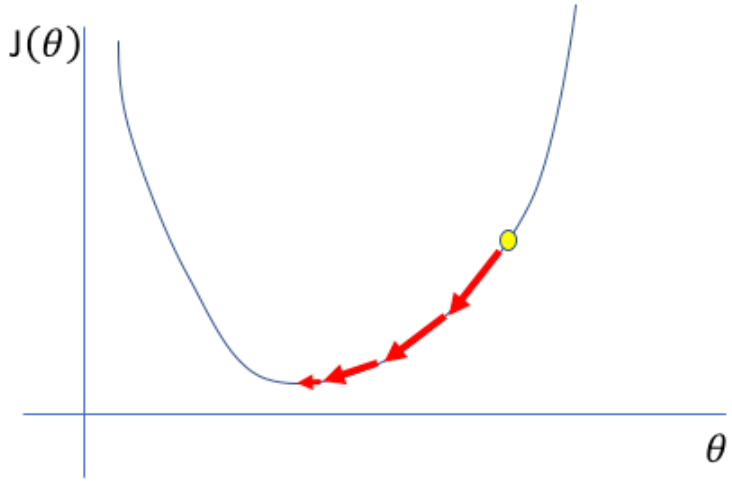
Learning rate

Too low



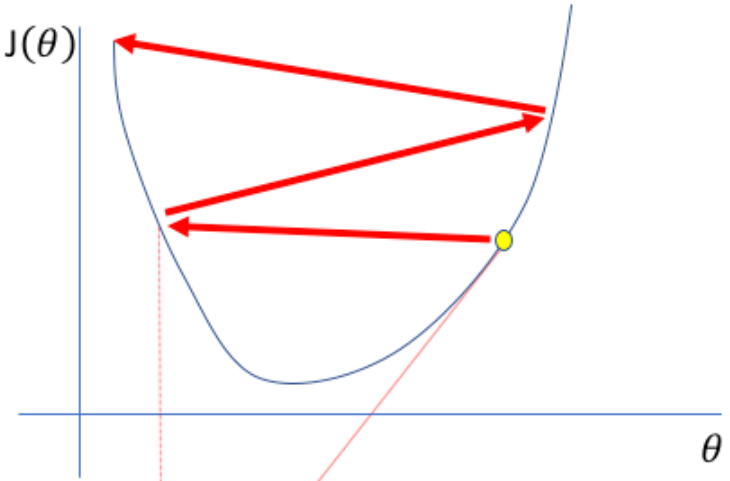
A small learning rate requires many updates before reaching the minimum point

Just right



The optimal learning rate swiftly reaches the minimum point

Too high



Too large of a learning rate causes drastic updates which lead to divergent behaviors

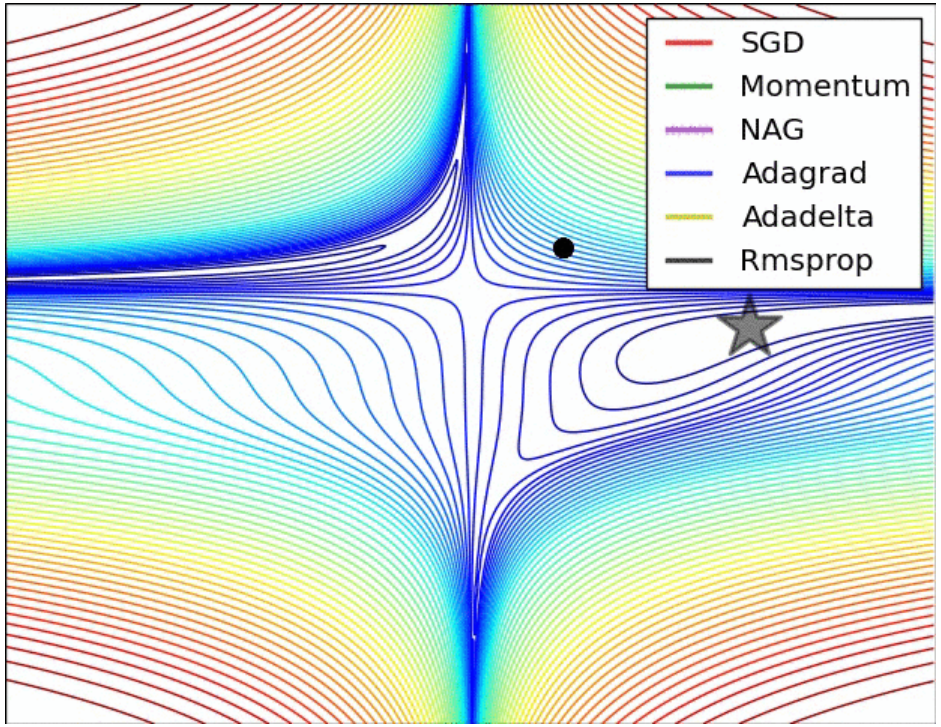
https://www.jeremyjordan.me/nn-learning-rate/

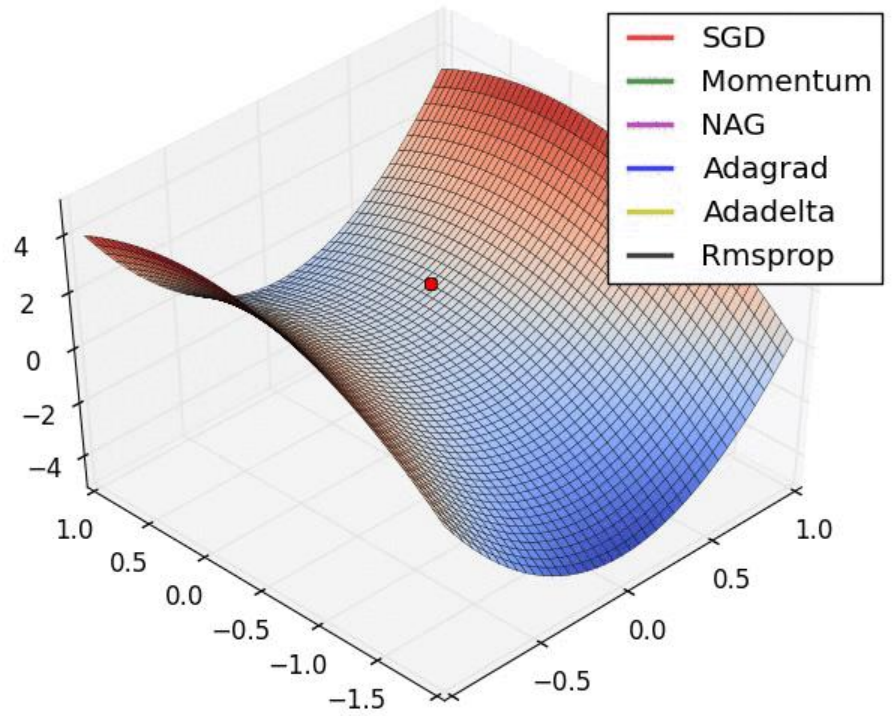
Optimizers

- Momentum: Sutton, R. S. (1986). Two problems with backpropagation and other steepest-descent learning procedures for networks. Proc. 8th Annual Conf. Cognitive Science Society
  - 在同方向的維度上學習速度會變快，方向改變的時候學習速度會變慢。
- NAG: Nesterov Accelerated Gradient, Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, pp. 543– 547
  - Momentum的變形。
- Adagrad: http://jmlr.org/papers/v12/duchi11a.html
  - 前期梯度較小的時候，放大學習率；後期梯度較大的時候，約束學習率。
- Adadelta: https://arxiv.org/abs/1212.5701
  - Adagrad的變形。
- RMSprop: proposed by Geoff Hinton
  - Adadelta的特例。
- Adam: Adaptive Moment Estimation, https://arxiv.org/abs/1412.6980
  - Momentum & Adagrad的結合。
- AdaMax, Nadam, ……

https://arxiv.org/pdf/1609.04747.pdf

Optimizers





http://ruder.io/optimizing-gradient-descent/index.html#adagrad

Classification - MNIST

Classification - MNIST database

- The MNIST database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples.
- It is a subset of a larger set available from NIST (National Institute of Standards and Technology). The digits have been size-normalized and centered in a fixed-size image.
- It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

http://yann.lecun.com/exdb/mnist/

Classification - MNIST database

- MNIST 手寫圖檔資料，總共有四的壓縮檔，分別為訓練用（training set）與測試用（test set）的影像與標示（label）檔：

```
# training set images (9912422 bytes)
wget http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
# training set labels (28881 bytes)
wget http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
# test set images (1648877 bytes)
wget http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
# test set labels (4542 bytes)
wget http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
```

https://blog.gtwang.org/programming/tensorflow-softmax-regression-hand-written-digits-recognizer-tutorial/

Classification - MNIST database

- MNIST 的原始圖檔資料是 20x20 像素的手寫圖，計算重心之後，放在 28x28 的圖檔中心（重心對準新圖檔中心），而 training set 標示檔的檔案格式如下：

Offset	資料類型	資料值	說明
0000	32 bits integer	0x00000801(2049)	magic number (MSB first)
0004	32 bits integer	60000	資料筆數
0008	unsigned byte	0 ~ 9	label 資料
0009	unsigned byte	0 ~ 9	label 資料
...	unsigned byte	0 ~ 9	label 資料

標示資料的值皆為 0 到 9 的數值。

https://blog.gtwang.org/programming/tensorflow-softmax-regression-hand-written-digits-recognizer-tutorial/

Classification - MNIST database

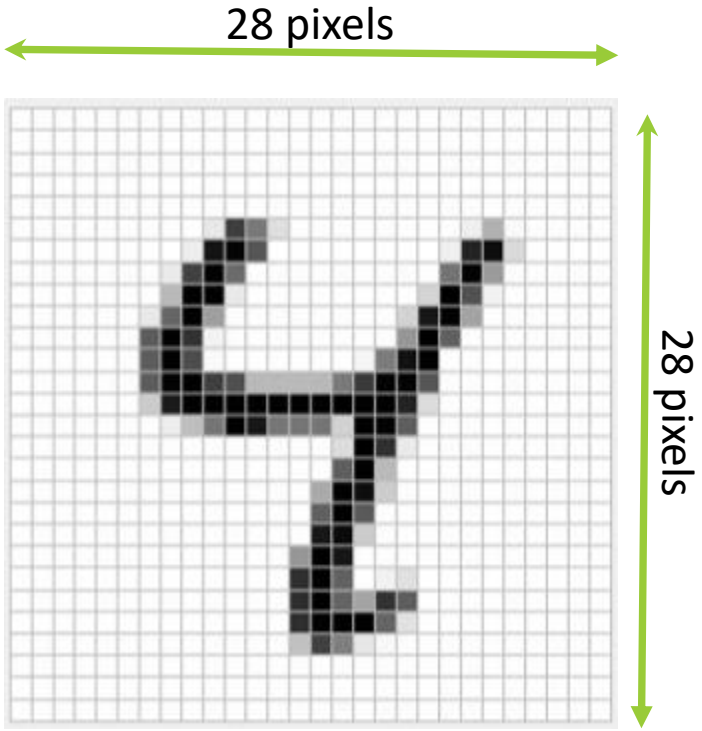
- Training set 圖檔的檔案格式如下：

Offset	資料類型	資料值	說明
0000	32 bits integer	0x00000803(2051)	magic number
0004	32 bits integer	60000	資料筆數
0008	32 bits integer	28	rows 數目
00012	32 bits integer	28	columns 數目
0016	unsigned byte		像素資料
0017	unsigned byte		像素資料
...	unsigned byte		像素資料

https://blog.gtwang.org/programming/tensorflow-softmax-regression-hand-written-digits-recognizer-tutorial/

Classification - MNIST database

- 像素資料是採用row-wise的方式儲存，其值為 0 到 255，0 代表背景（白色），255 代表前景（黑色）。



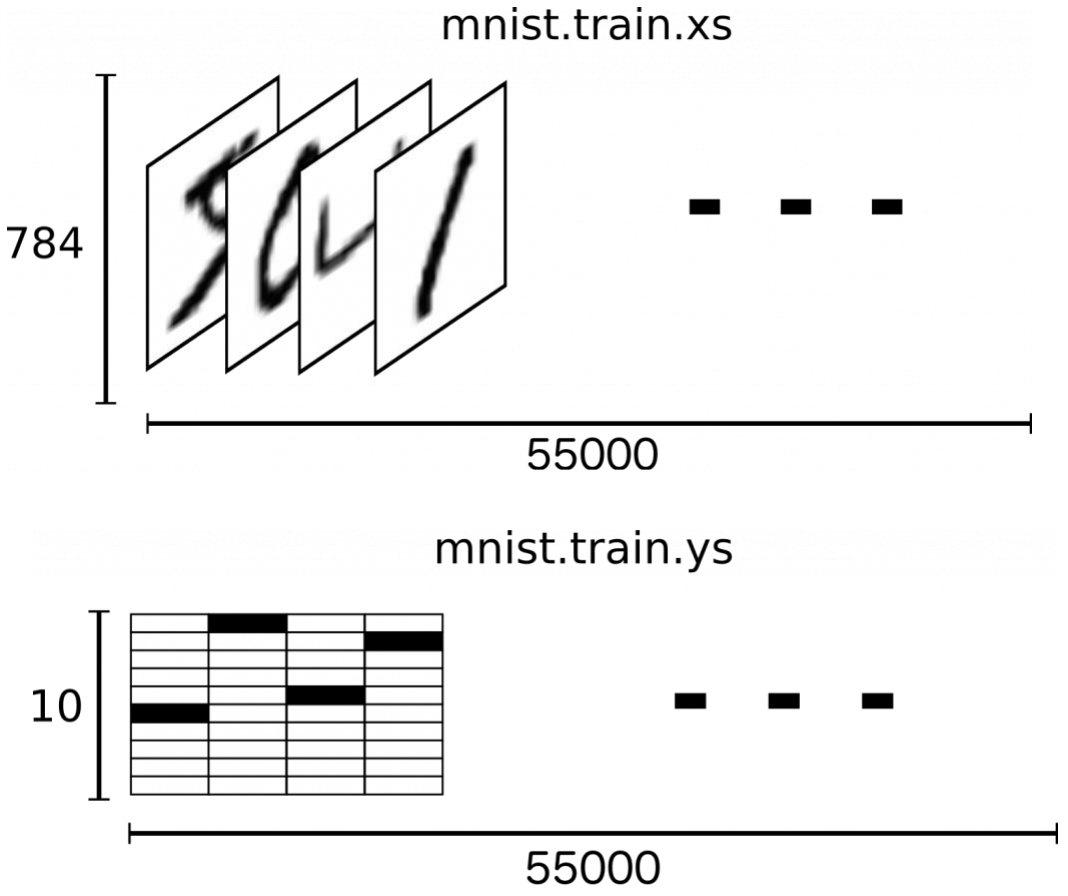
- MNIST資料讀取進來後，會分成三部份：
  - mnist.train：訓練用資料55,000筆。
  - mnist.validation：驗證用資料5,000筆。
  - mnist.test：測試用資料10,000筆。

- 每部份資料都有影像（如mnist.train.images）與標示（如 mnist.train.labels），我們將影像當作輸入的 x，而標示則當作輸出的 y。

Classification - MNIST database

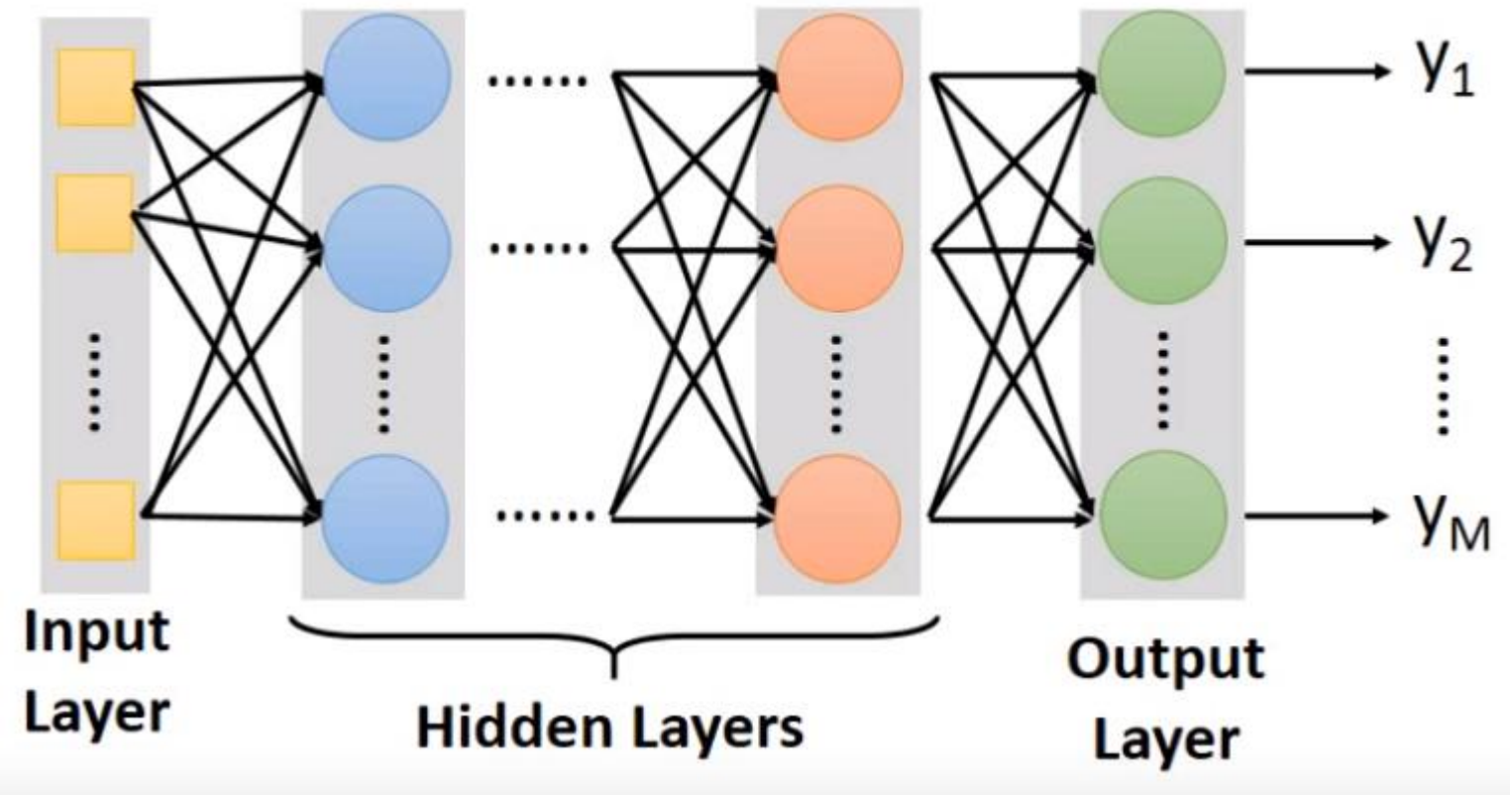
- 這裡我們將每一個影像都視為長度為28x28=784的向量，因此 mnist.train.images 就是一個shape為[55000, 784]的tensor，其中每個數值都是一個介於 0 與 1 的浮點數。

- 在標示的資料（mnist.train.label）上，我們使用長度為 10 的向量來表示 0 到 9 的數字，這種向量之中只有一個元素是 1，其餘都是 0，當第 n 個向量元素是 1 時就代表第 n 個數字，例如 [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] 就代表 3。



https://blog.gtwang.org/programming/tensorflow-softmax-regression-hand-written-digits-recognizer-tutorial/

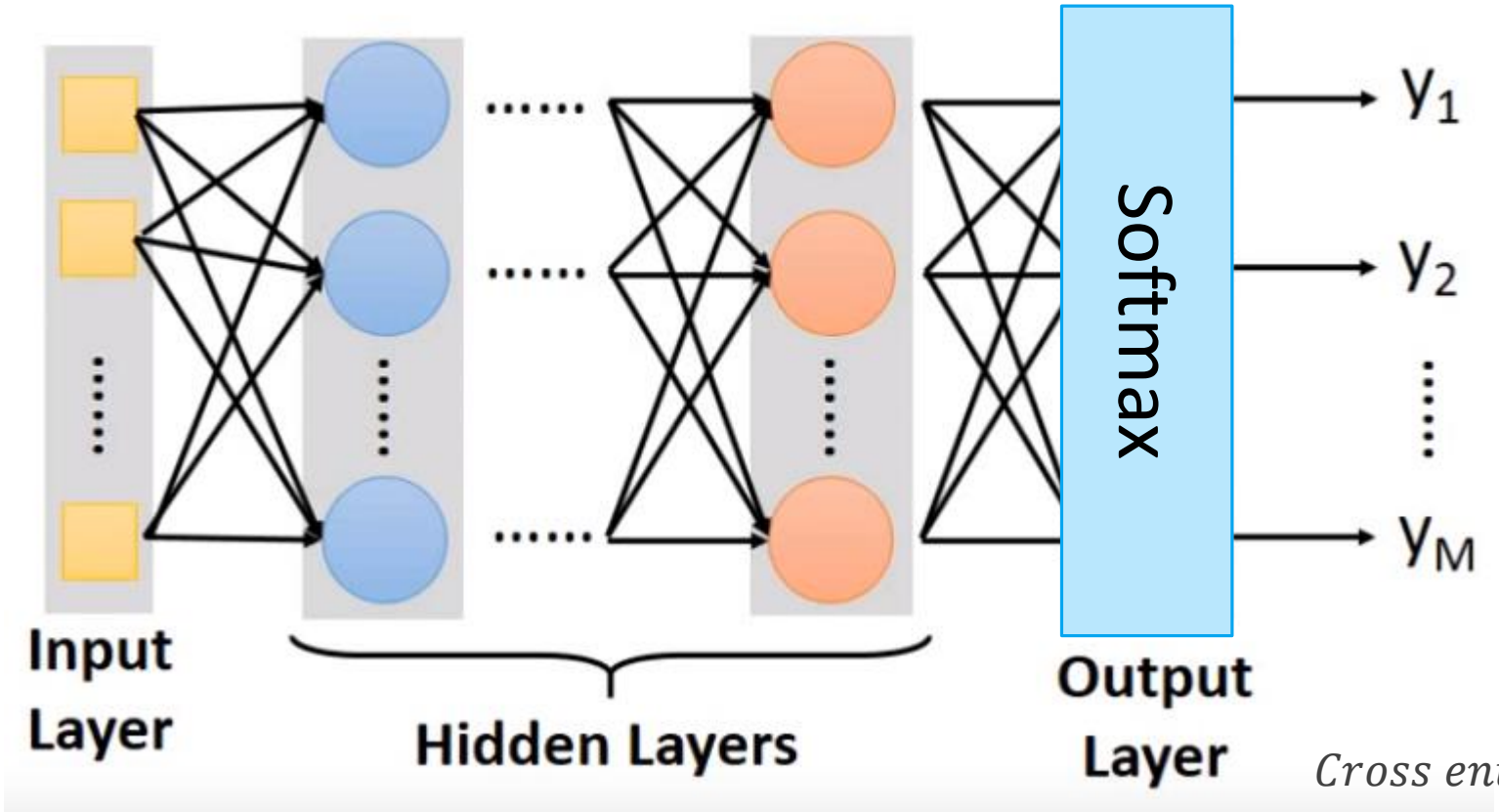
Classification



?

https://www.youtube.com/watch?v=Dr-WRlEFefw

Classification



$\widehat{y_1}$

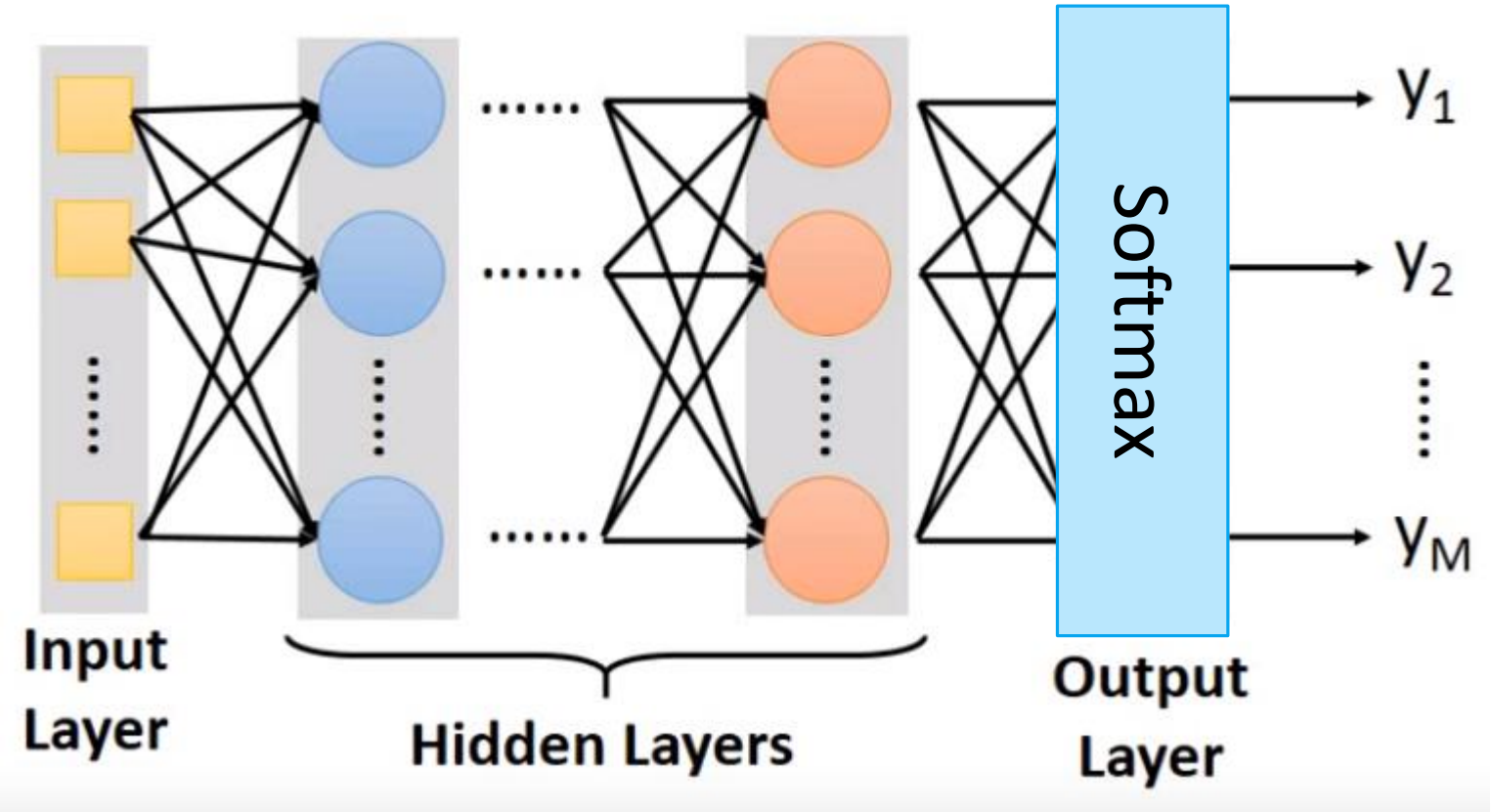
$\widehat{y_2}$

$\widehat{y_M}$

$$Cross\ entropy = \sum_{c=1}^{C}\sum_{i=1}^{n} -y_{c,i} log_2(p_{c,i})$$

https://www.youtube.com/watch?v=Dr-WRlEFefw

Classification - Softmax



$$\text{Softmax}(y)_i = \frac{e^{y_i}}{\sum_{j=1}^{M} e^{y_j}}$$

Where *i*=1, …, M

https://www.youtube.com/watch?v=Dr-WRlEFefw

Classification - MNIST database

- 定義模型：
  - 定義輸入的資料x：

```
x = tf.placeholder(tf.float32, [None, 784])
```

這裡的x就是多張MNIST的影像資料，每一張影像就是一個784維的向量，此處的None代表該維度的長度可以是任意值。

  - 定義權重與偏差值：

```
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
```

由於W要乘上784維的向量（輸入影像），然後產生10維的表徵特性向量（代表一個數字），所以W的shape會是[784, 10]，而b只是單純加在10維的表徵特性向量上的常數而已，所以shape為[10]。

https://blog.gtwang.org/programming/tensorflow-softmax-regression-hand-written-digits-recognizer-tutorial/

Classification - MNIST database

- 定義模型：
  - 建立主要的模型：

```
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

此處我們使用tf.matmul處理矩陣乘法，再用tf.nn.softmax處理softmax函數。

- 訓練模型：
  - 我們使用 cross-entropy 的方式來衡量模型的表現。實作時，要增加一個placeholder來輸入真實的分佈（正確答案）：

```
y_ = tf.placeholder(tf.float32, [None, 10])
```

```
cross_entropy = tf.reduce_mean(
  tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y))
```

https://blog.gtwang.org/programming/tensorflow-softmax-regression-hand-written-digits-recognizer-tutorial/

Classification - MNIST database

- 訓練模型：
  - 接著使用backpropagation演算法尋找最佳解：

```
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

這裡我們使用 gradient descent 演算法，learning rate 取 0.5。

  - 建立一個 InteractiveSession，並初始化 variables：

```
sess = tf.InteractiveSession()
tf.global_variables_initializer().run()
```

  - 進行 1000 次的訓練，在每一次的迭代中，我們從訓練用資料中隨機取出 100 筆資料來使用（稱為一個 batch）：

```
for _ in range(1000):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

Classification - MNIST database

- 驗證模型：
  - tf.argmax 可將向量中最大值的索引取出來，我們可以利用這個函數來檢查模型預測值與實際值是否相符合：

```
correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
```

  - 再將這些布林值轉換為浮點數，計算整體模型的準確度：

```
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

  - 最後一步就是實際執行驗證：

```
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

```
0.9074
```

  - 準確率約在90%左右，這個值尚未達到收斂，如果多增加訓練步驟，可以增加到 92% 以上。

Classification - MNIST references

- 3Blue1Brown
  - https://www.youtube.com/watch?v=aircAruvnKk
  - https://www.youtube.com/watch?v=IHZwWFHWa-w
  - https://www.youtube.com/watch?v=Ilg3gGewQ5U

- Backward propagation
  - https://www.youtube.com/watch?v=tIeHLnjs5U8

Next class

- Hierarchical clustering
- K-mean
- Modularity
- ……