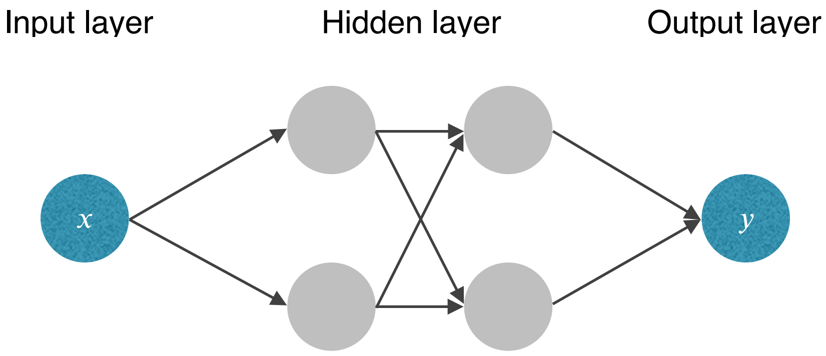
2017年5月21日 星期日

Neural network

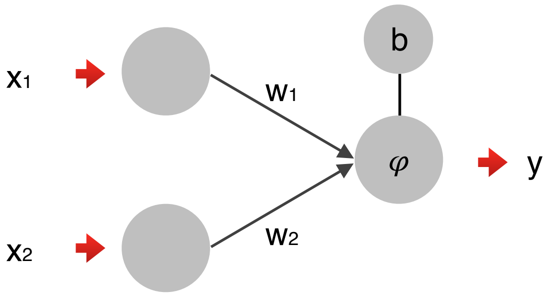
Simple Neural Network

- 結構：



- 過程：

輸入 → 加權(weight) → 神經元上的誤差值(bias) → 代入activation function



$$\Rightarrow \varphi (w_1 x_1 + w_2 x_2 + b) = y$$

- Loss function：

當神經網路結構與activation function已經決定，可調整的參數剩下weights和biases，集合起來形成 $\{ F_\theta \}$。

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{k} ||y_i - F_\theta(x_i)||^2$$

mse : mean(square(error))

- Gradient descent：

$$\begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} - \eta\nabla L \ \text{, where } \nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \frac{\partial L}{\partial b} \end{bmatrix}$$

- 範例：

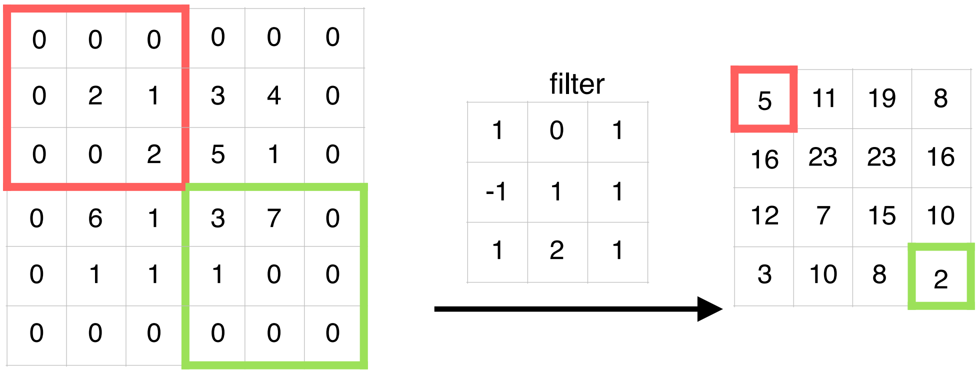
```
model = Sequential()

# input = 28x28, output 500 nodes in first hidden layer with sigmoid function
model.add(Dense(500, input_dim=784))
model.add(Activation(‘sigmoid'))

# input = 500, output 500 nodes in second hidden layer with sigmoid function
model.add(Dense(output_dim=500))
model.add(Activation(‘sigmoid'))

# input = 500, output 10 nodes with sigmoid function
model.add(Dense(output_dim=10))
model.add(Activation(‘softmax'))

model.compile(loss='mse', optimizer=SGD(lr=0.1), metrics=[‘accuracy'])
```

Convolutional Neural Network (CNN)

- 結構：

  Input → convolutional layer → max-pooling layer (→ convolutional layer → max-pooling layer) → output

- Convolutional layer：

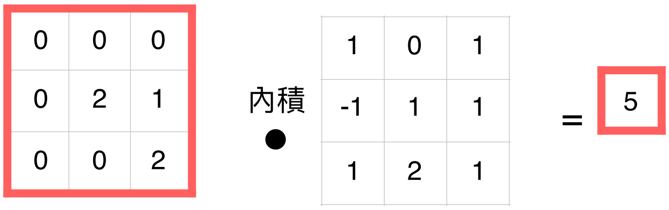


0	0	0	0	0	0
0	2	1	3	4	0
0	0	2	5	1	0
0	6	1	3	7	0
0	1	1	1	0	0
0	0	0	0	0	0

filter

1	0	1
-1	1	1
1	2	1

5	11	19	8
16	23	23	16
12	7	15	10
3	10	8	2

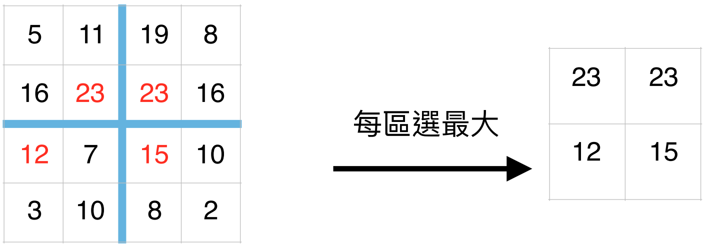


0	0	0
0	2	1
0	0	2

1	0	1
-1	1	1
1	2	1

= 5

- Max-pooling layer：



5	11	19	8
16	23	23	16
12	7	15	10
3	10	8	2

23	23
12	15

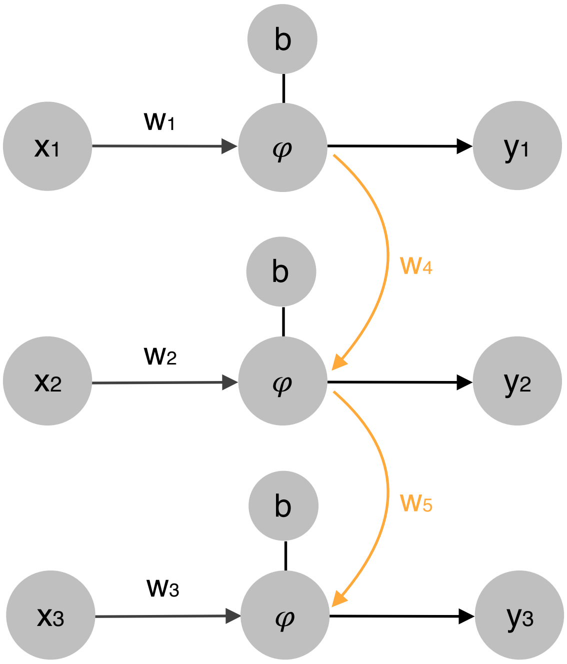
- 範例：

```
model = Sequential()

# input = 28 x 28 x 1, output 28 x 28 x 10 nodes in convolutional layer (kernel size = 3 x 3) with relu function
model.add(Conv2D(10, (3,3), padding='same', input_shape=(28,28,1)))
model.add(Activation('relu'))


# max-pooling layer with pool_size = 2 x 2, output 14 x 14 x 10 nodes
model.add(MaxPooling2D(pool_size=(2,2)))

# input = 14 x 14 x 10, output 14 x 14 x 10 nodes in convolutional layer (kernel size = 3 x 3) with relu function
model.add(Conv2D(10, (3,3), padding='same'))
model.add(Activation(‘relu’))

# max-pooling layer with pool_size = 2 x 2, output 7 x 7 x 10 nodes
model.add(MaxPooling2D(pool_size=(2,2)))

# Flatten the input to 1-dim, output 490 nodes
model.add(Flatten())

# input = 490, output 200 nodes with relu function
model.add(Dense(200))
model.add(Activation('relu'))
```

```
# input = 200, output 10 nodes with softmax function

model.add(Dense(10))

model.add(Activation('softmax'))


model.compile(loss="categorical_crossentropy", optimizer=Adadelta(),
                    metrics=[‘accuracy'])
```

Recurrent Neural Network (RNN)

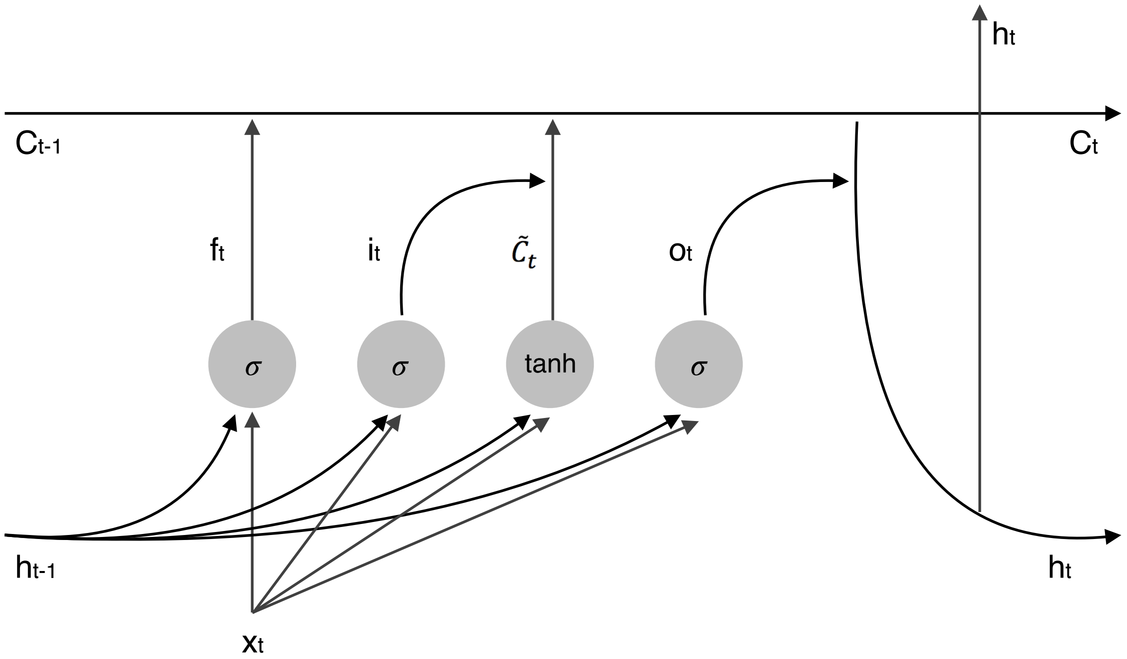
- 有記憶的神經網路
- 結構：

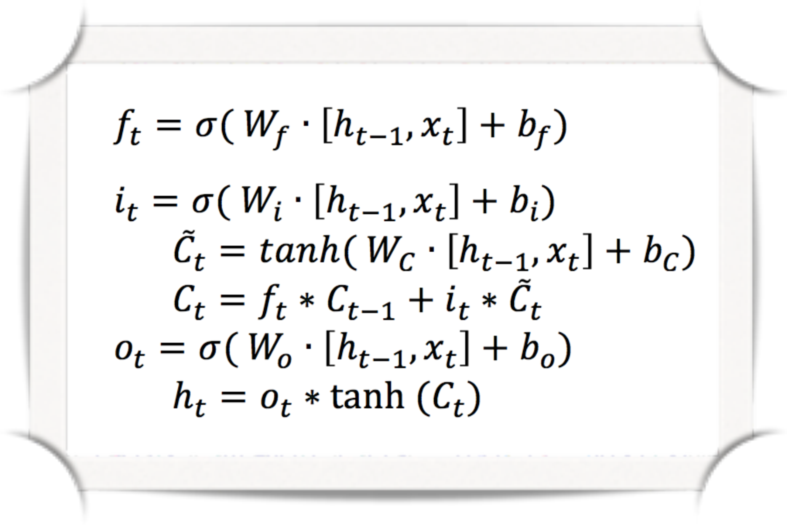


- 範例：

Long Short Term Memory (LSTM)

- 解決長期依賴問題，記住長時間段的訊息
- 控制閥(gate) ：讓信息選擇性通過的機制，由sigmoid神經元組成
- Sigmoid：描述神經元有多少信息被通過，輸出0(全都不通過)或1(全都通過)
- 忘記門(f)＋輸入門(i)＋輸出門(o)
- 神經元狀態(C)貫穿結構
- 結構：





$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh\,(C_t)$$