心理與神經資訊學
(Psychoinformatics & Neuroinformatics)

課號: Psy5261

識別碼: 227U9340

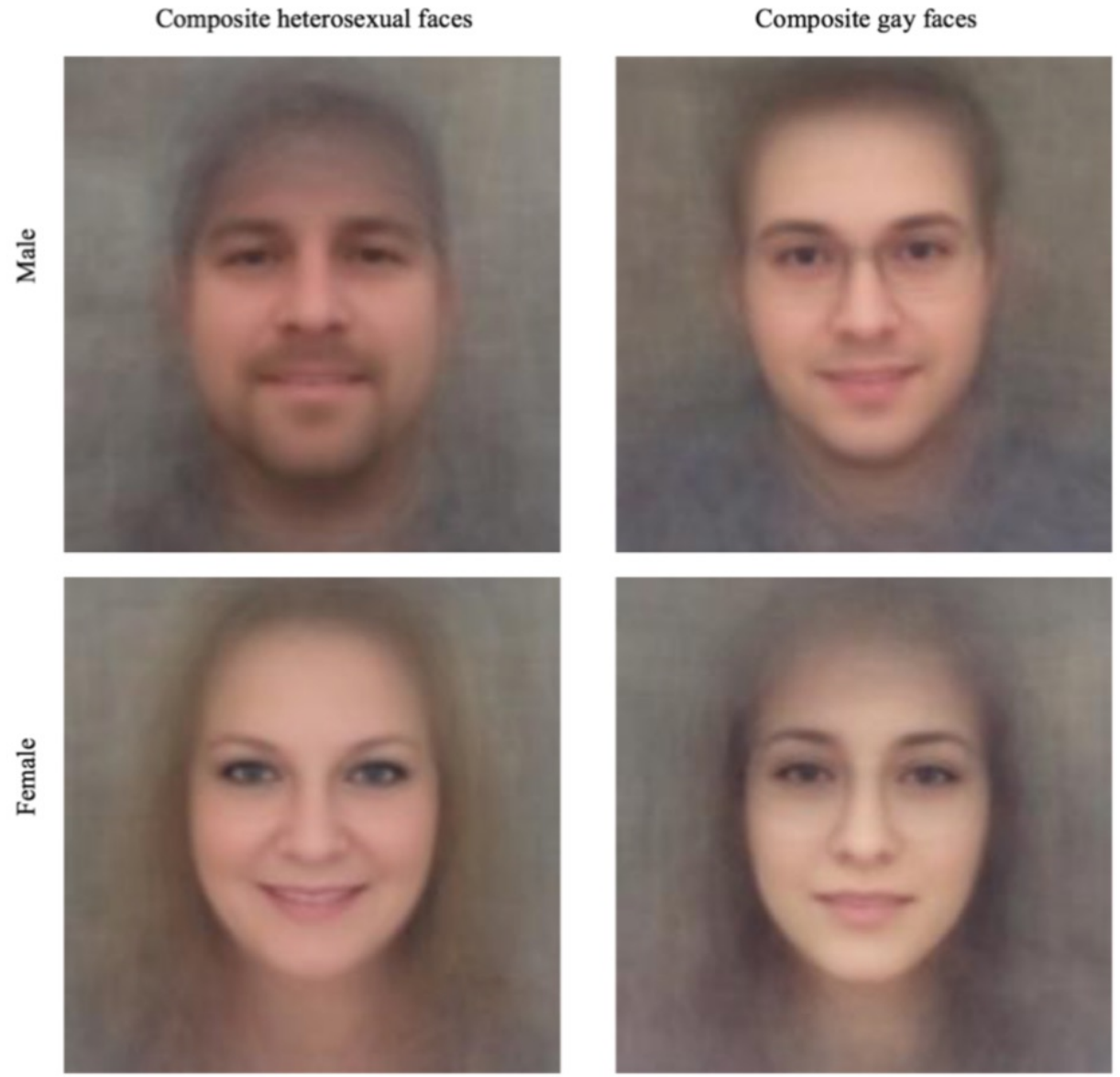
教室:彷彿在雲端

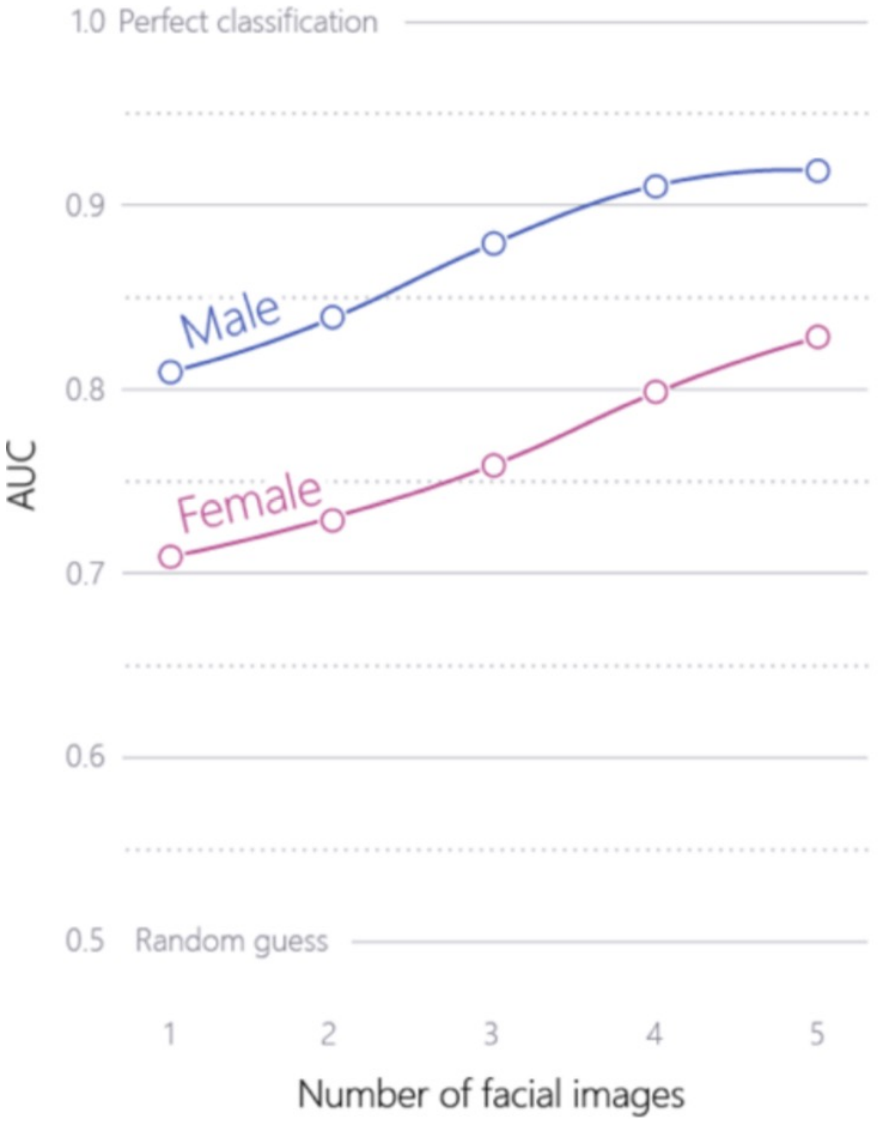
時間: 一789


請先pip install tensorflow keras
!!

Deep Learning Nets的應用(1/5)

利用VGG-Face來判斷是否為同志(classification)





Deep Learning Nets的應用(2/5)

判斷誰可能犯罪(classification)

(a) Three samples in criminal ID photo set S_c.

(b) Three samples in non-criminal ID photo set S_n

Figure 1. Sample ID photos in our data set.

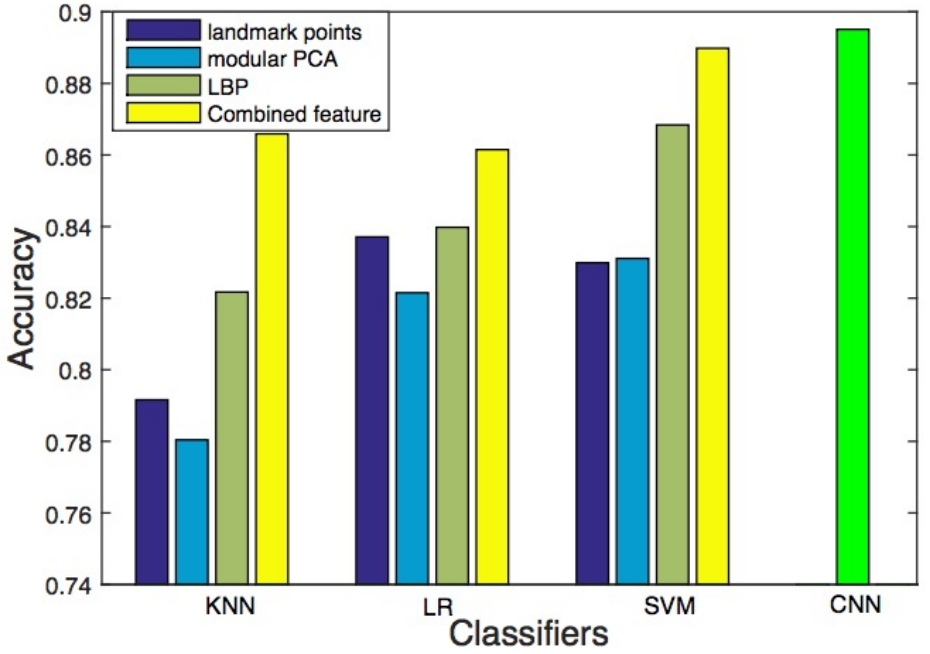


Figure 2. Accuracy of all four classifiers in all thirteen cases.

Deep Learning Nets的應用(3/5)

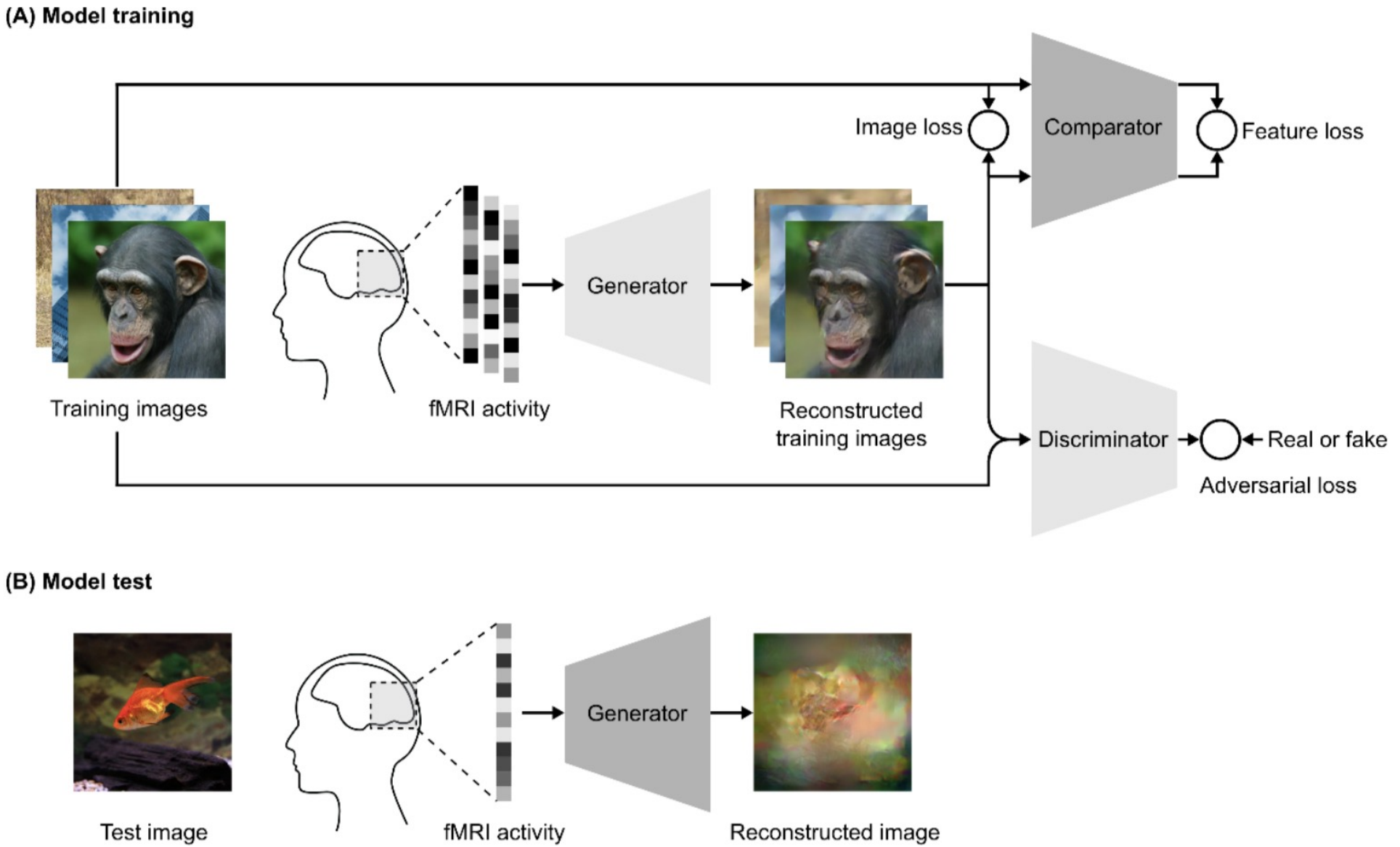
判斷誰比較美(regression)

TABLE VI. CORRELATION COEFFICIENTS IN SINGLE NETWORK

Exp.	1	2	3	4	5	Average
PC	0.8509	0.8050	0.8112	0.7817	0.8446	0.8187

Deep Learning Nets的應用(4/5)

利用生成對抗網絡(GAN)來做image reconstruction

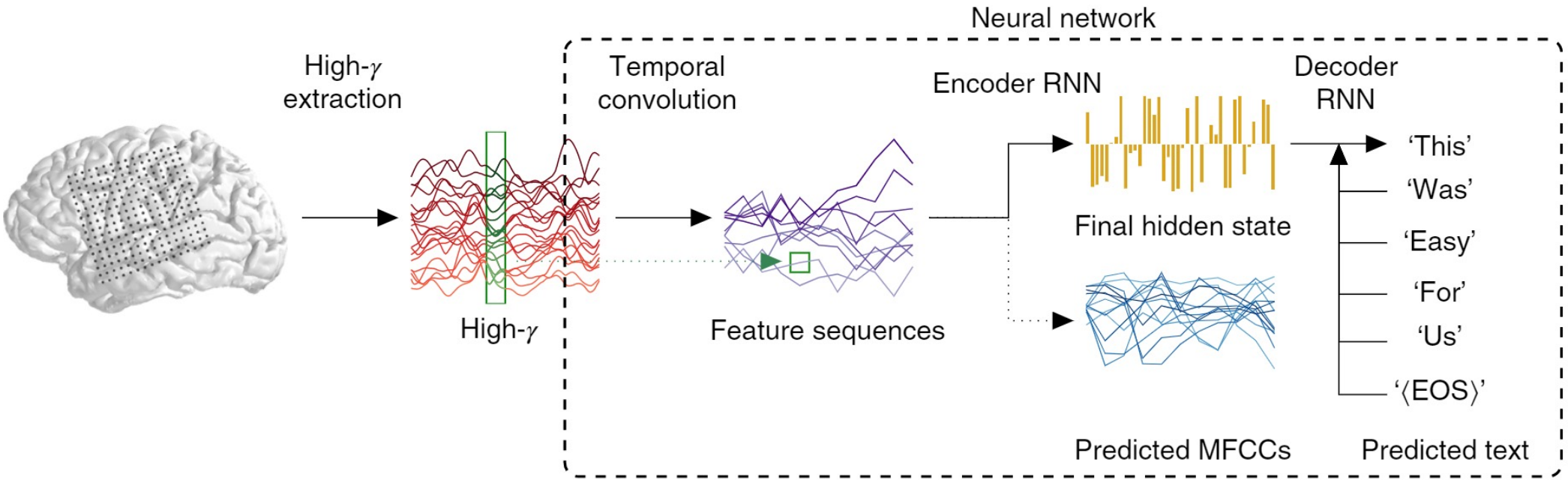


Deep Learning Nets的應用(5/5)

利用循環神經網路 (RNN)來做brain decoding

Machine translation of cortical activity to text with an encoder–decoder framework

Joseph G. Makin[1,2] ✉, David A. Moses[1,2] and Edward F. Chang[1,2] ✉

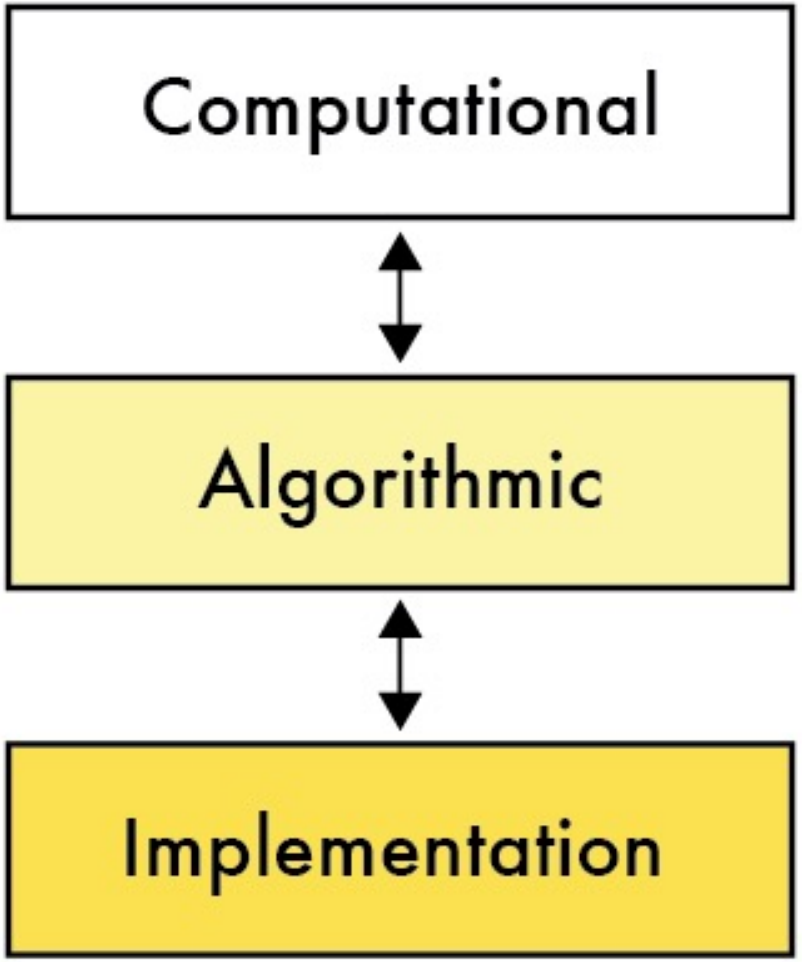


神經計算&類神經網路

(Neural Computation & Neural Networks)

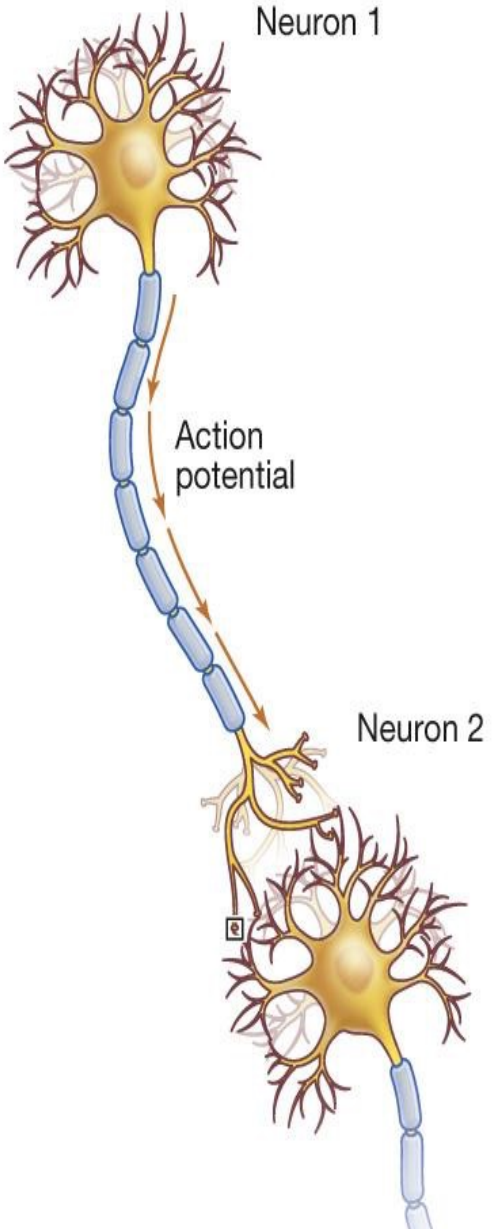
評估認知系統的特性與極限

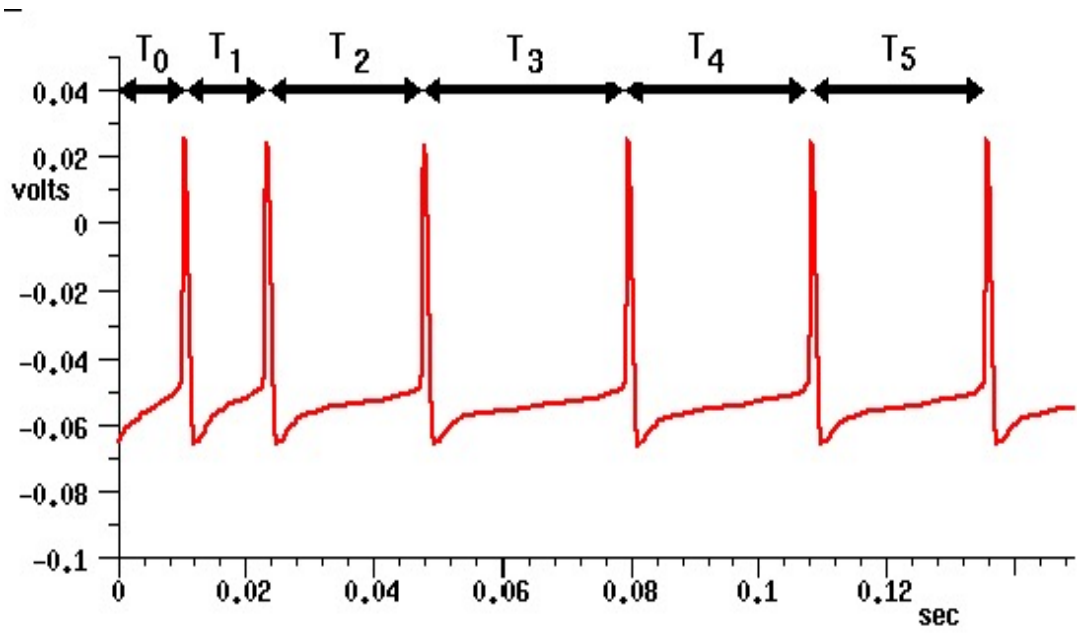
David Marr's 3 levels of analysis

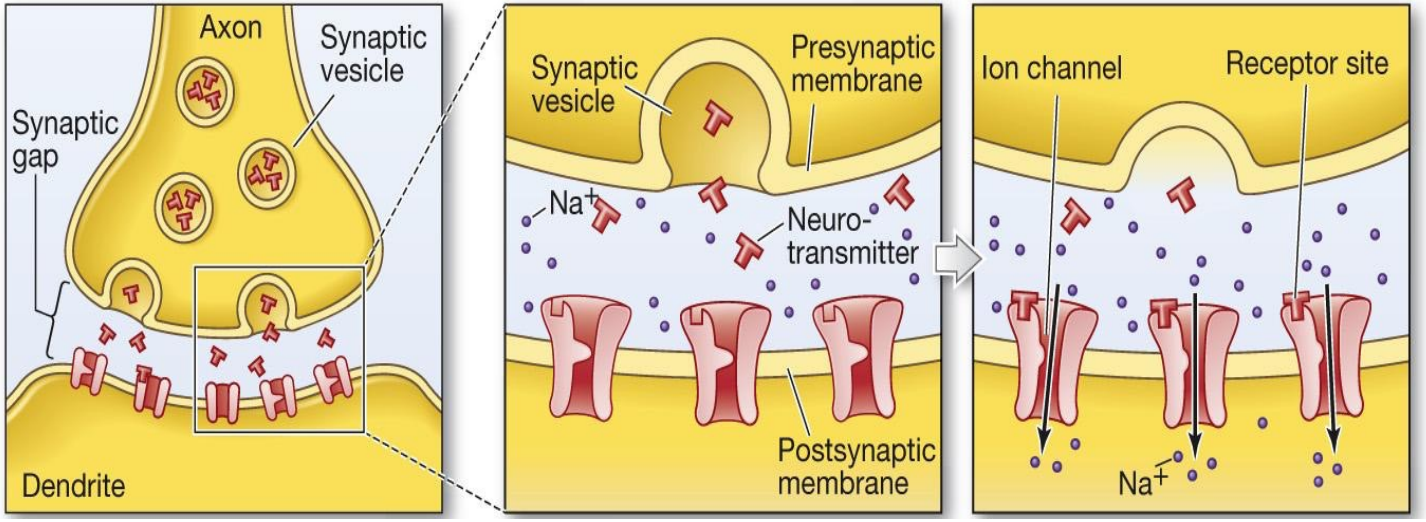


- 計算上的問題是什麼?
- 能夠用什麼演算法來解決?
- 要透過什麼硬體來實現算法?

Implementation:有0與1的神經元

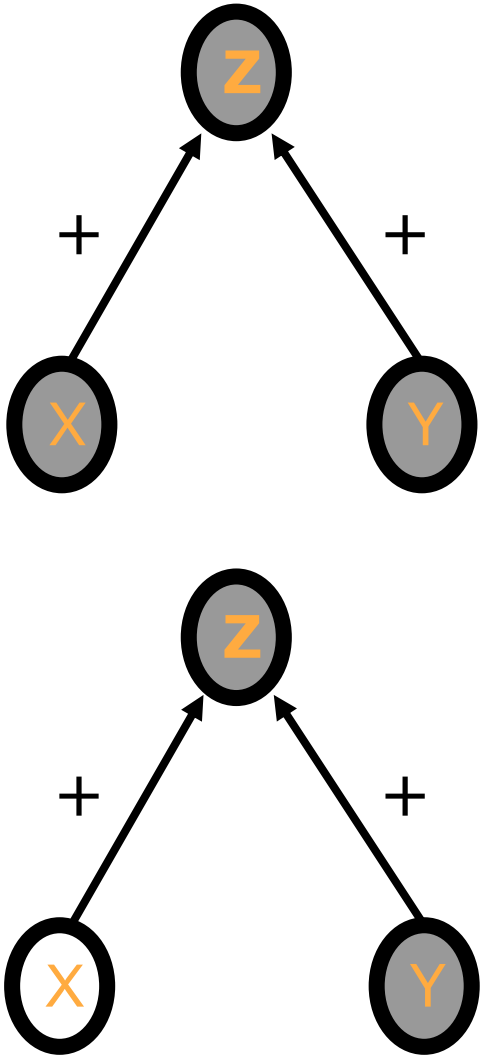






Implementation:神經元做加法

如果接受刺激的神經元有個低閾值

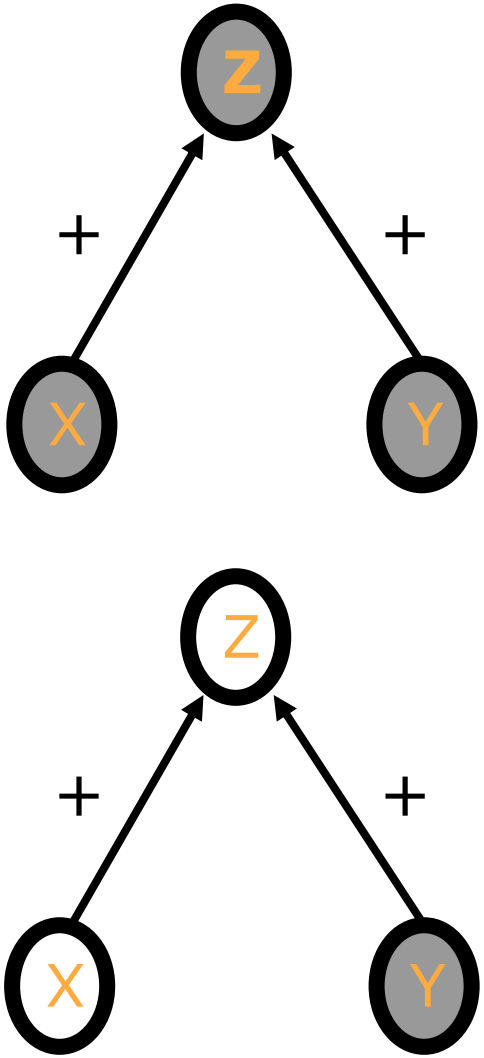


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Z=X+Y (OR)

Implementation:神經元做乘法

如果接受刺激的神經元有個高閾值

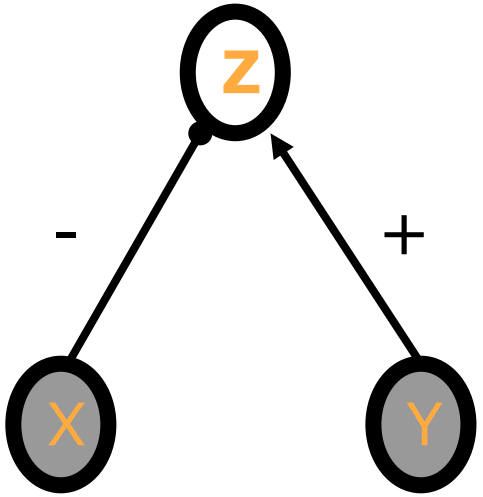


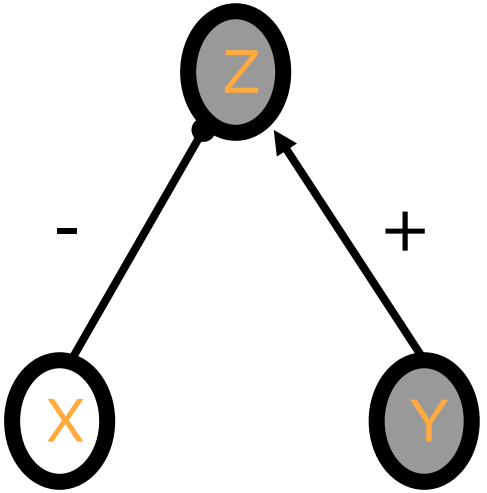
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Z=X*Y (AND)

Implementation:神經元做減法

如果開始考慮神經元彼此的抑制關係



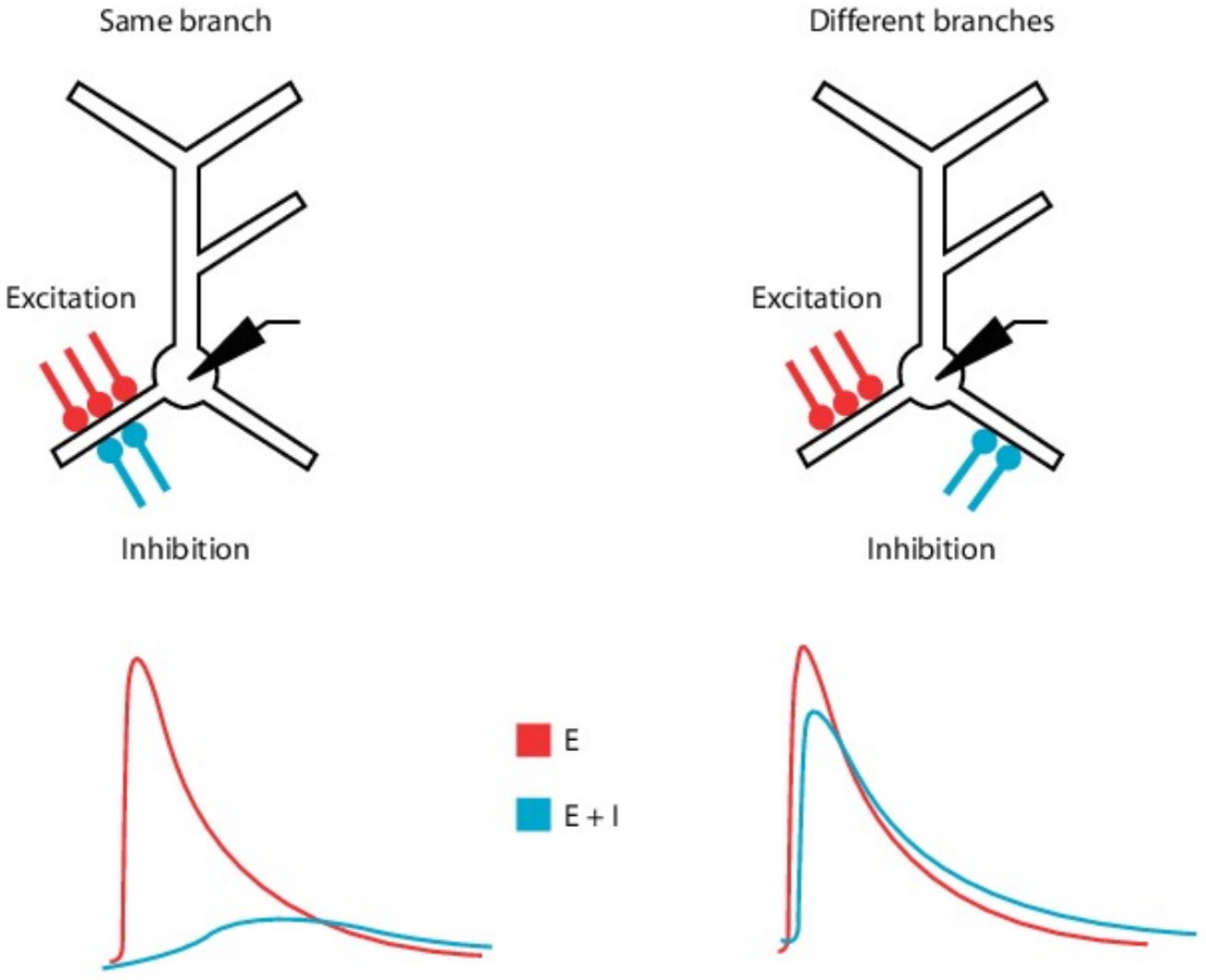


X	Z
0	1
1	0

Z=1-X (NOT)

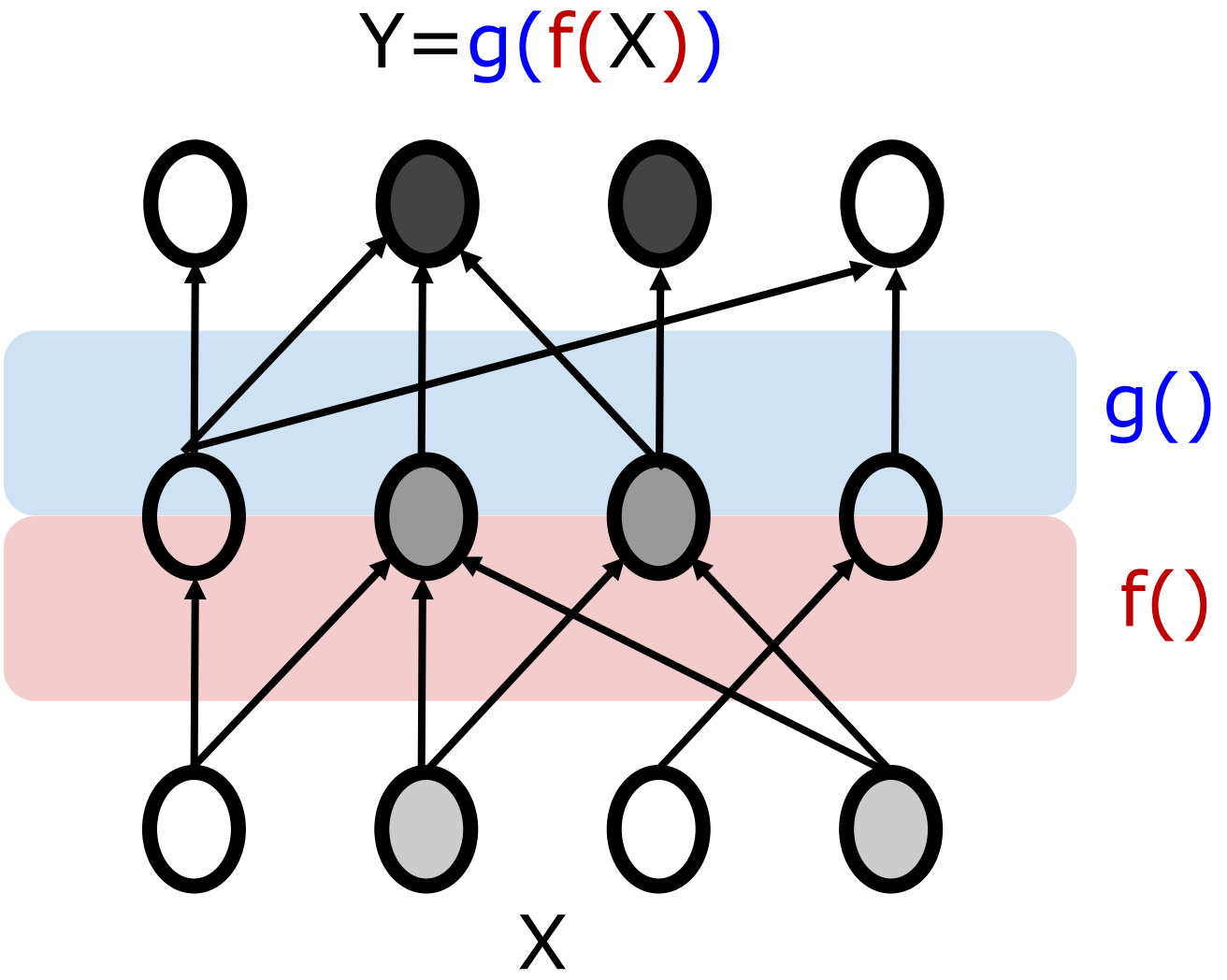
Implementation:神經元做除法

神經元間的抑制可以導致減法或除法



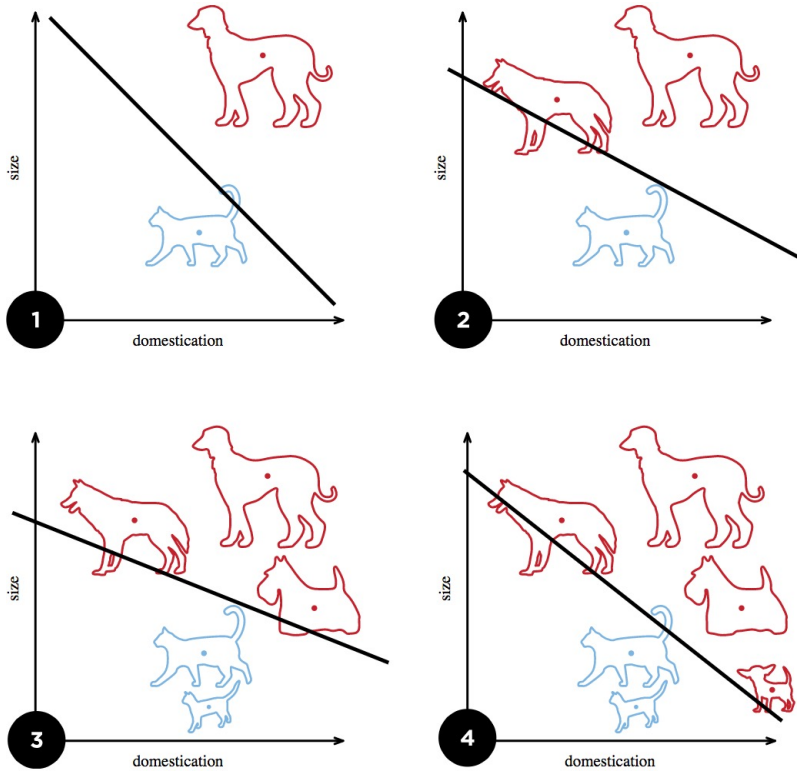
Algorithm:基本計算的組合

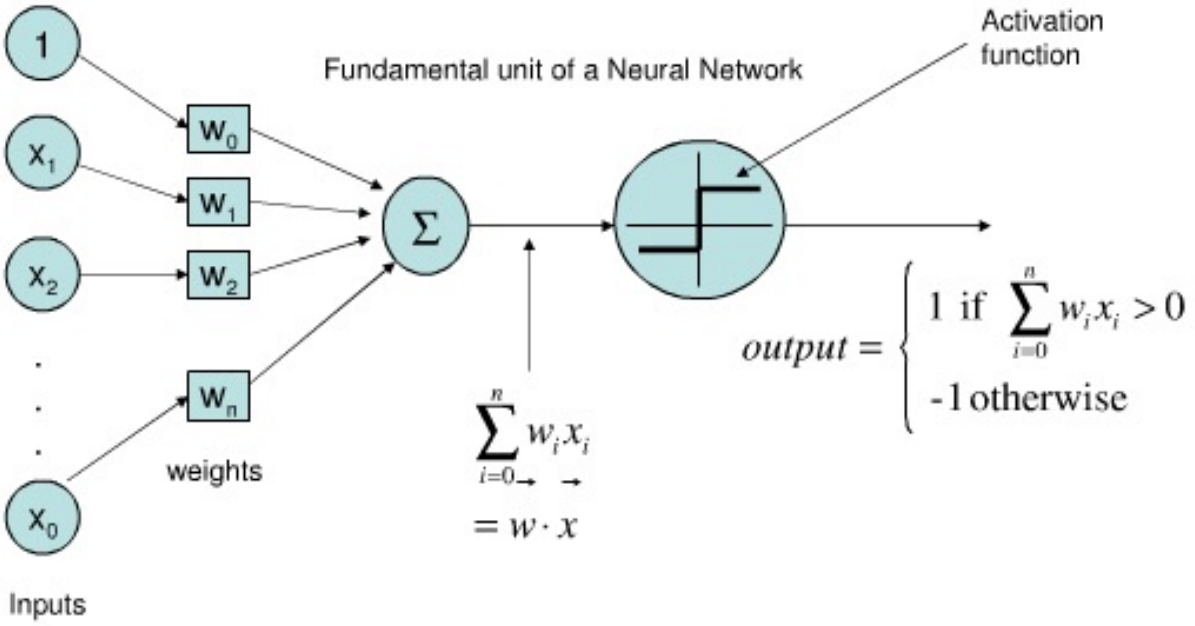
一個神經網路透過組合基本的計算來實現演算法



Computational Problem:區分貓狗

Perceptron (Rosenblatt, 1958)
Cognitron (Fukushima, 1975)





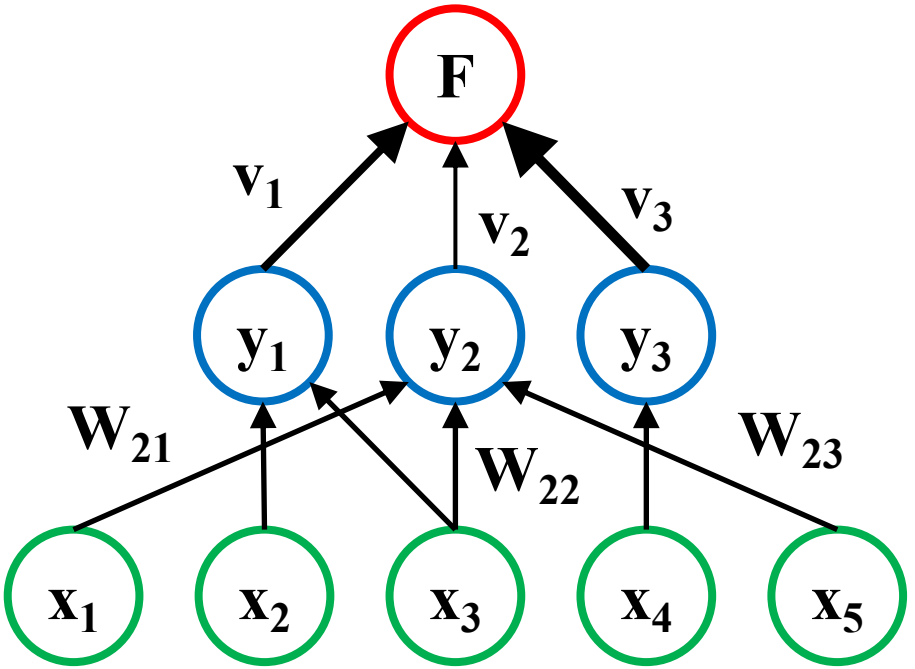
決策邊界: $Y=aX+b \Rightarrow Y-aX-b=0$

$\Rightarrow (1,-a,-b)\cdot(Y,X,1)=0$

類神經網路的學習
(Learning of Neural Networks)

Universal Approximation Theorem

3層網路就可以逼近任何連續函數(cf. 傅立葉分析)



$$F(x) = \sum_{i=1}^{N} v_i \underbrace{\varphi\left(w_i^T x + b_i\right)}_{y_i}$$

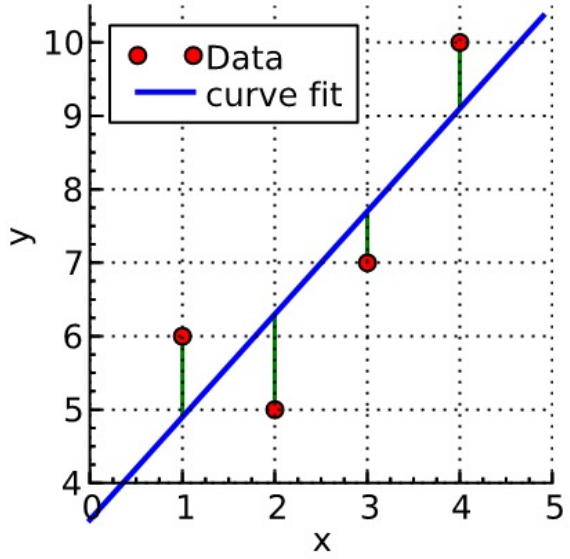
as an approximate realization of the function f where f is independent of φ; that is,

$$|F(x) - f(x)| < \varepsilon$$

for all $x \in I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$.

Data Fitting

通常用最小平方法一次性地去最小化整體預測錯誤

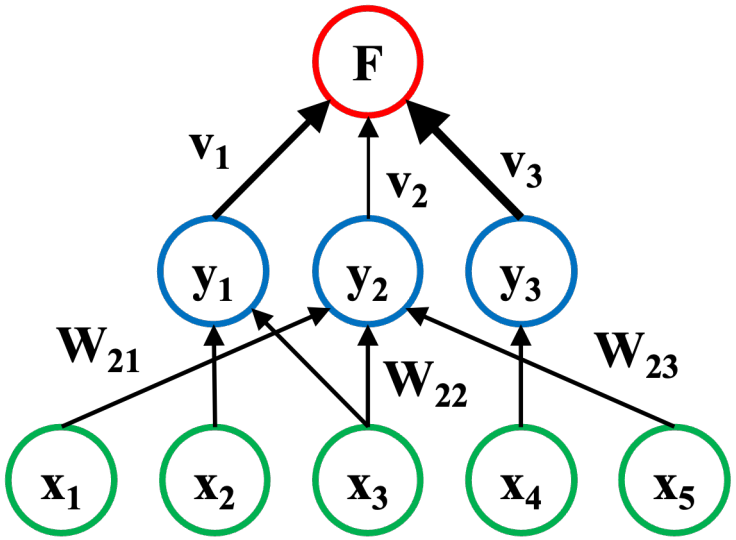


$$y = \alpha + \beta x$$

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

$$\hat{\varepsilon}_i = y_i - a - b x_i$$

$$\mathbf{L}(a,b) = \sum_{i=1}^{n} \hat{\varepsilon}_i^{\,2} = \sum_{i=1}^{n} (y_i - a - b x_i)^2$$

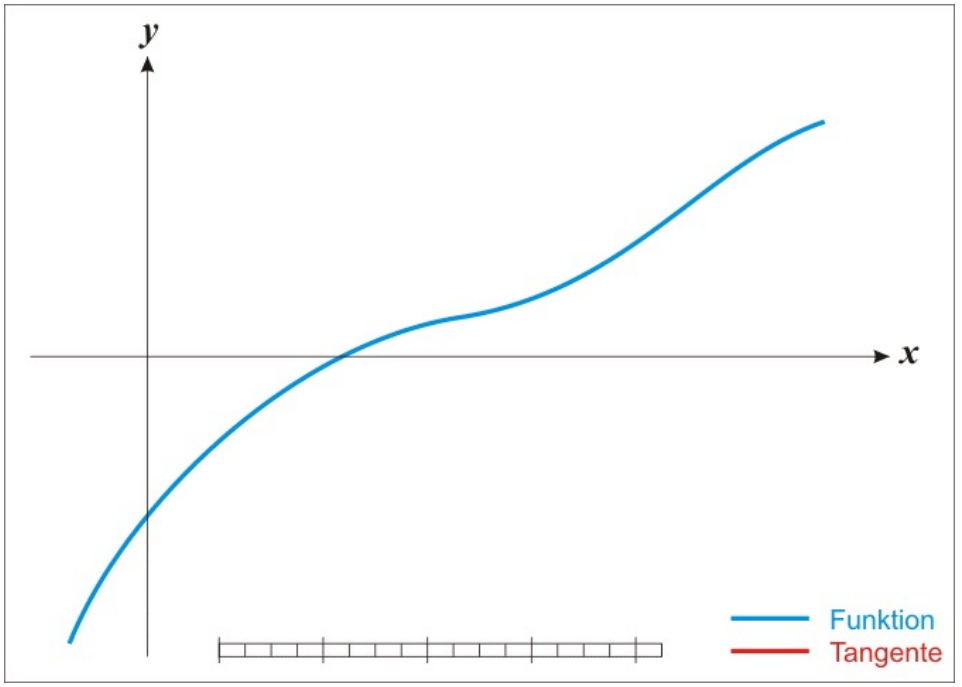


$$\hat{\alpha} = \bar{y} - \hat{\beta}\,\bar{x},$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Symbolic vs. Numerical Methods

數值法可以避免解(多變量的聯立)方程式 $x : f(x) = 0$



$$y = f'(x_n)(x - x_n) + f(x_n)$$

$$0 = f'(x_n)(x_{n+1} - x_n) + f(x_n)$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

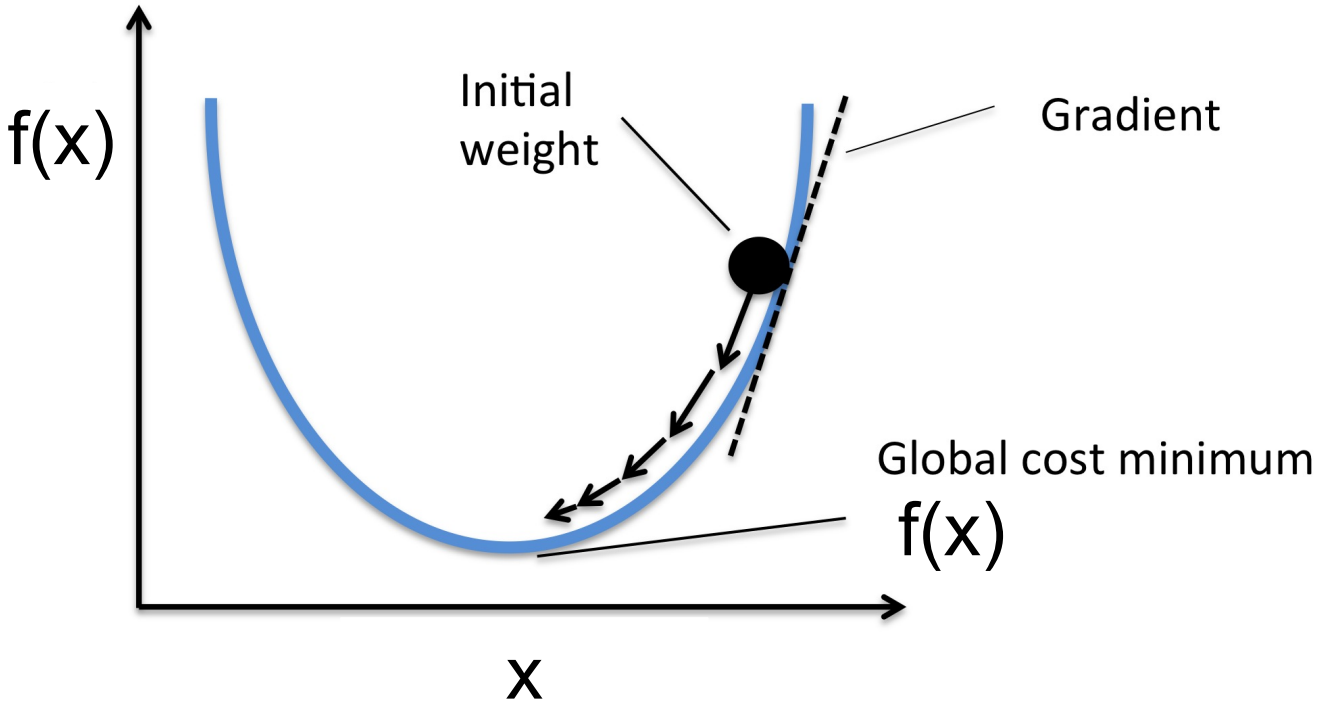
牛頓法透過多次調整x_n來解$f(x_n)=0$

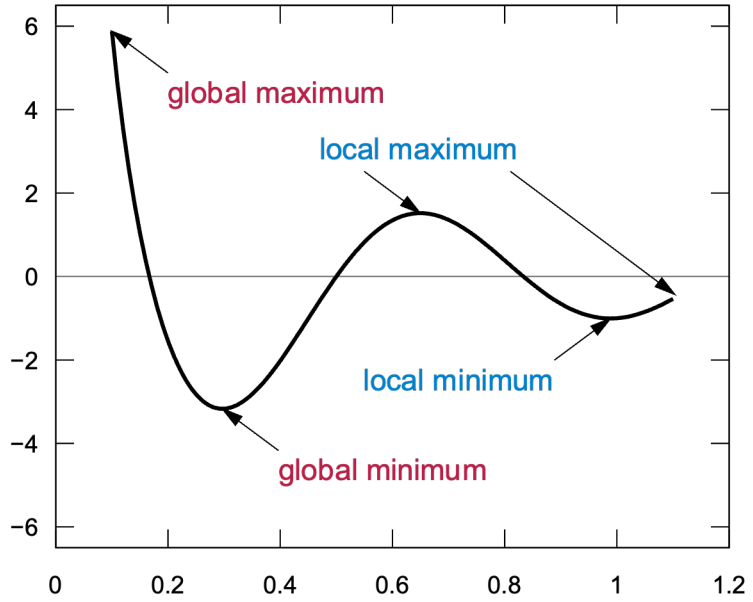
用Gradient Descent來最小化Error/Loss Function

局部最佳化: Gradient Descent

數次調整x_t使得Error function $f(x_t)$能夠逐步最小

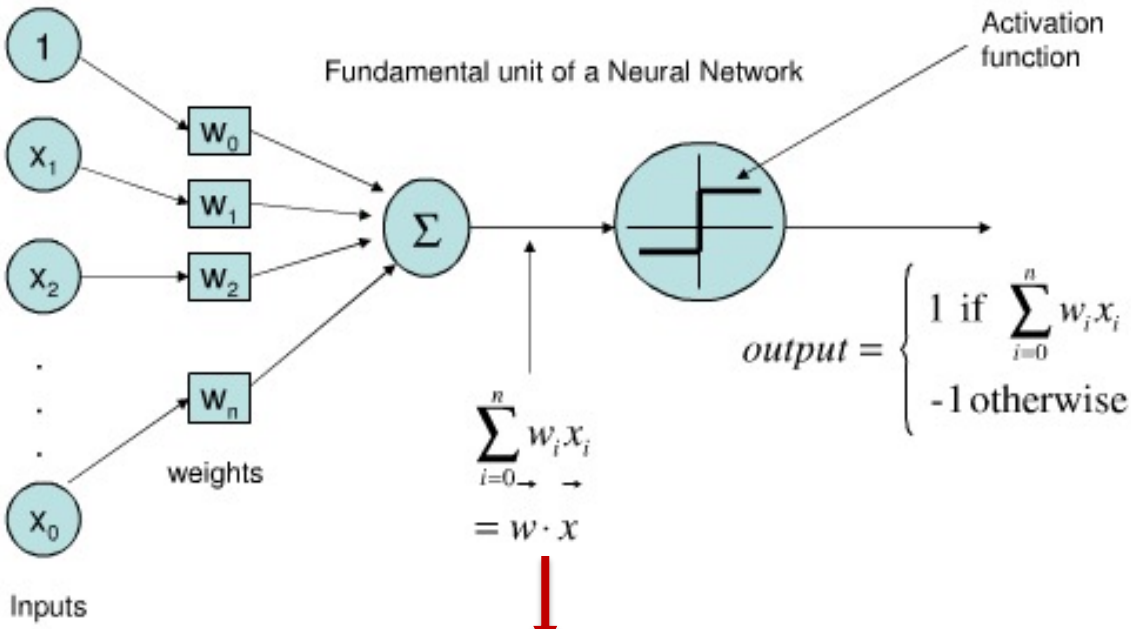
$$x_{t+1} = x_t - \gamma_t \nabla f(x_t) \Rightarrow f(x_0) \geq f(x_1) \geq f(x_2) \geq \ldots$$





分類問題: 以Perceptron為例

學習規則可由gradient descent推導而來



$$E = -\sum_p (WX^p) Y^p$$

分類問題就只看預測值和ground truth的sign是否一樣就好

迴歸問題E才用 $MSE = (Y^p - sgn(\sum_i WX^p))^2$

多類別問題通常act. F使用softmax:

$$\begin{bmatrix} 1.2 \\ 0.9 \\ 0.4 \end{bmatrix} \rightarrow \boxed{\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}} \rightarrow \begin{bmatrix} 0.46 \\ 0.34 \\ 0.20 \end{bmatrix}$$

而E使用cross-entropy:

$$\begin{bmatrix} 0.1 \\ 0.5 \\ 0.4 \end{bmatrix} \quad D(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_j y_j \ln \hat{y}_j \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$\hat{\mathbf{y}}$ $\mathbf{y}$

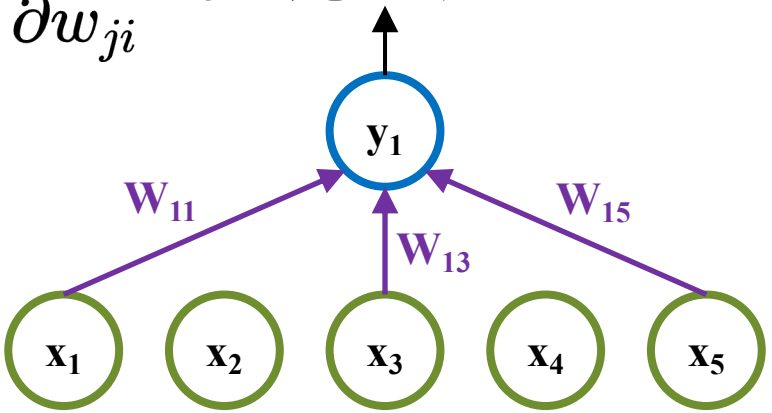
$$W^{new} = W^{old} + \Delta W = W^{old} - \partial E(W)/\partial W = W^{old} + X^p Y^p$$

迴歸問題: 以兩層網路為例

Error/Loss Function用Mean Squared Error (MSE)

$$o_j = \varphi(\text{net}_j) = \varphi\left(\sum_{i=1}^{n} w_{ji} o_i\right) \qquad \varphi(z) = \frac{1}{1+e^{-z}} \qquad E = \frac{1}{2}(t-y)^2$$

$\Delta w_{ji} = -\alpha \frac{\partial E}{\partial w_{ji}}$ 要算 $\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \text{net}_j} \frac{\partial \text{net}_j}{\partial w_{ji}}$ 裡有3項:



$$\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(t-y)^2 = y - t \equiv \delta$$

y_j Error

$$\frac{\partial o_j}{\partial \text{net}_j} = \frac{\partial}{\partial \text{net}_j}\varphi(\text{net}_j) = \varphi(\text{net}_j)(1-\varphi(\text{net}_j))$$ 因為 $\frac{\partial \varphi}{\partial z} = \varphi(1-\varphi)$

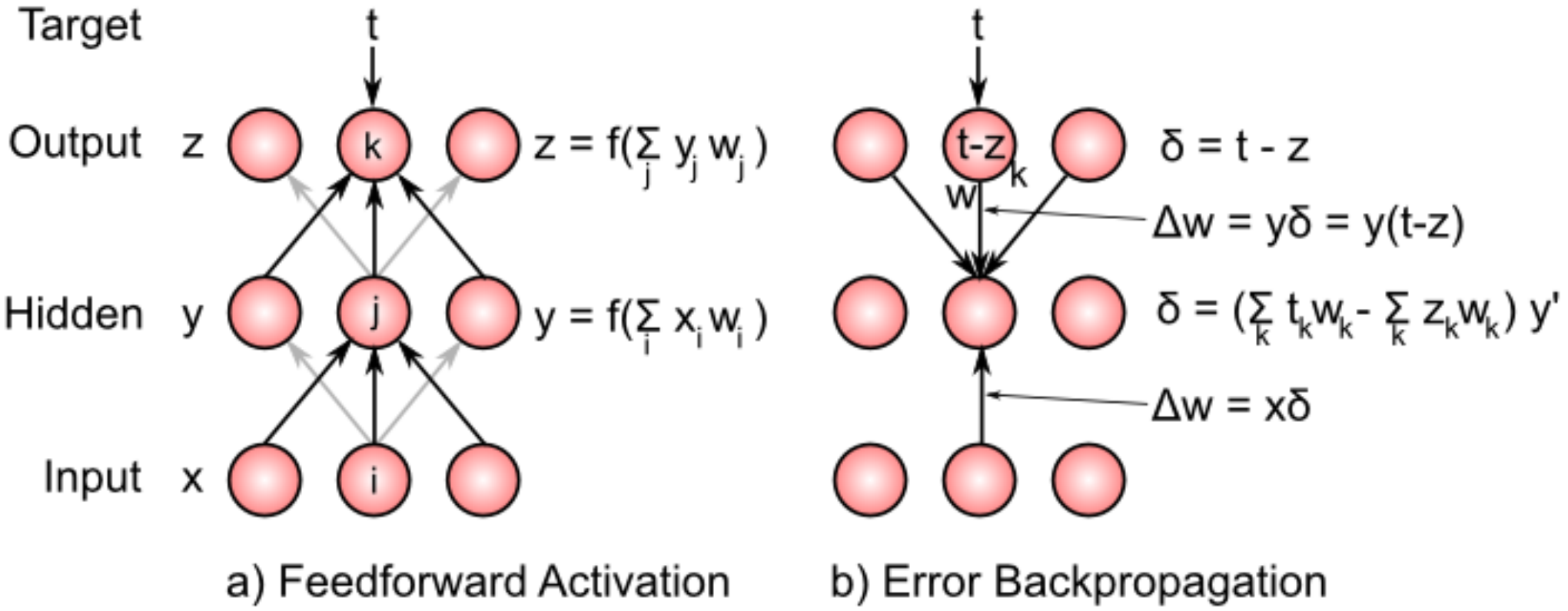
Local Gradient

$$\frac{\partial \text{net}_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\left(\sum_{k=1}^{n} w_{jk} o_k\right) = o_i$$

x_i Activation

多層網路的學習: Backpropagation

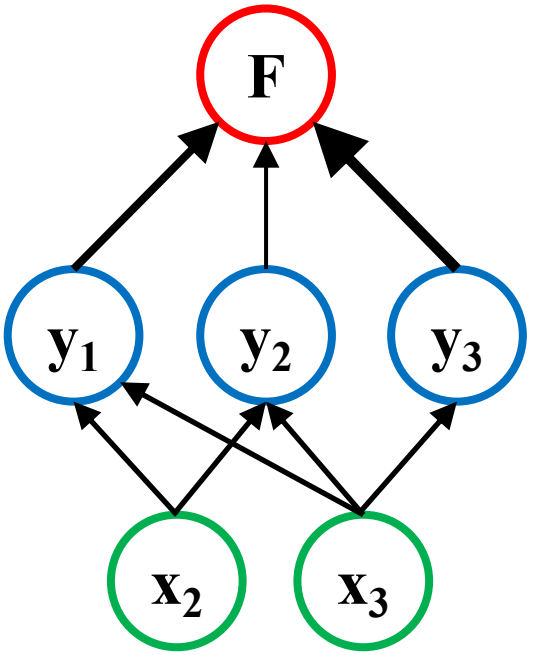
把errors從output層往input層傳以定義內層的errors



在deep neural nets中離output層最遠的δ≈0

ANN as a Machine Learner

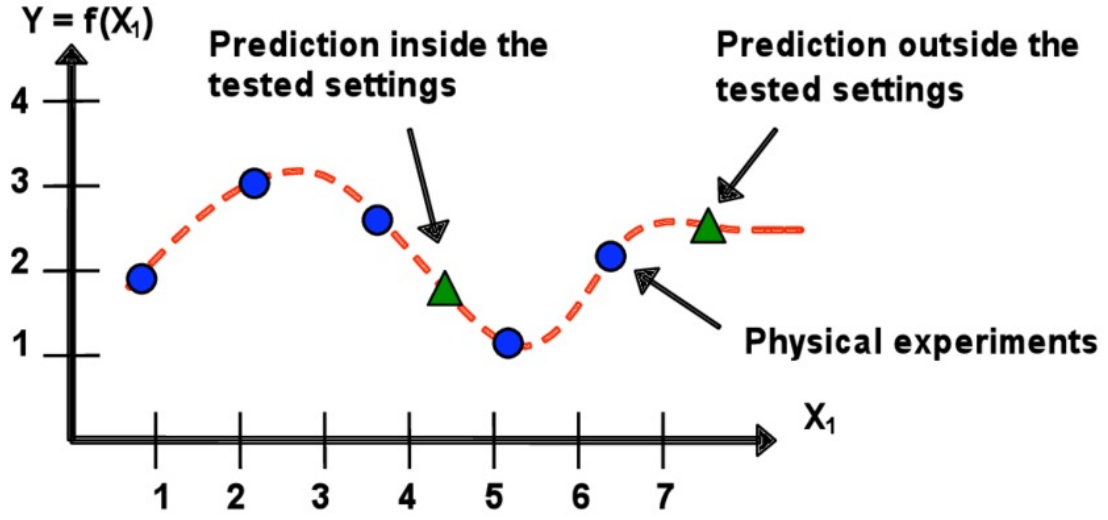
也會有各種歸納法(induction)所衍生出來的問題



Train

Test

p	q	p∧q
T	T	T
T	F	F
F	T	F
F	F	F

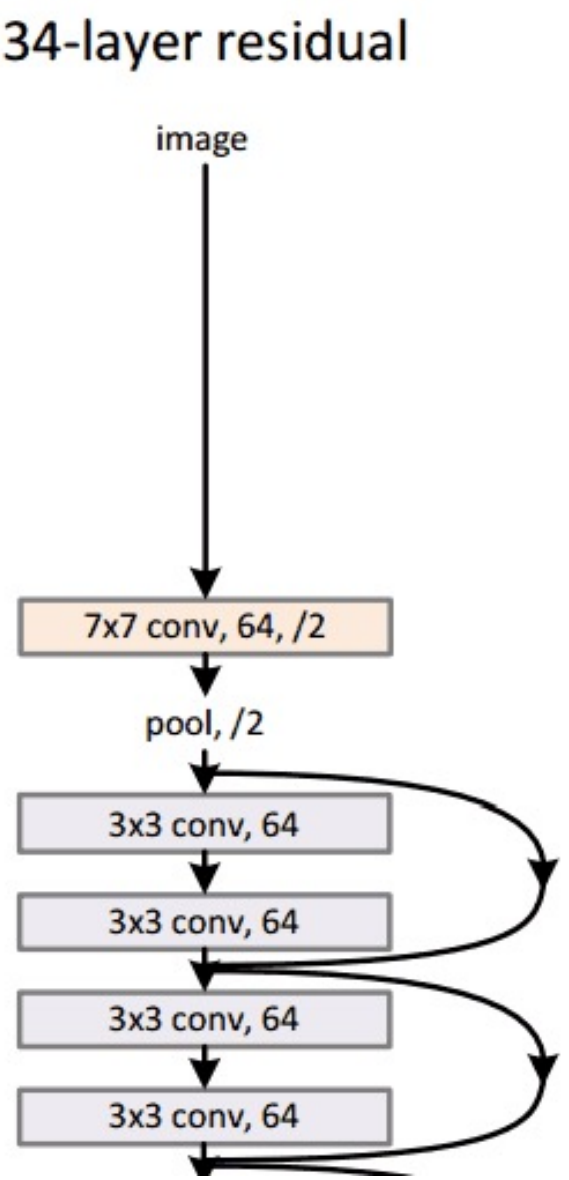


深度學習神經網路

(Deep Learning Neural Networks)

Deep Neural Network (1/2)

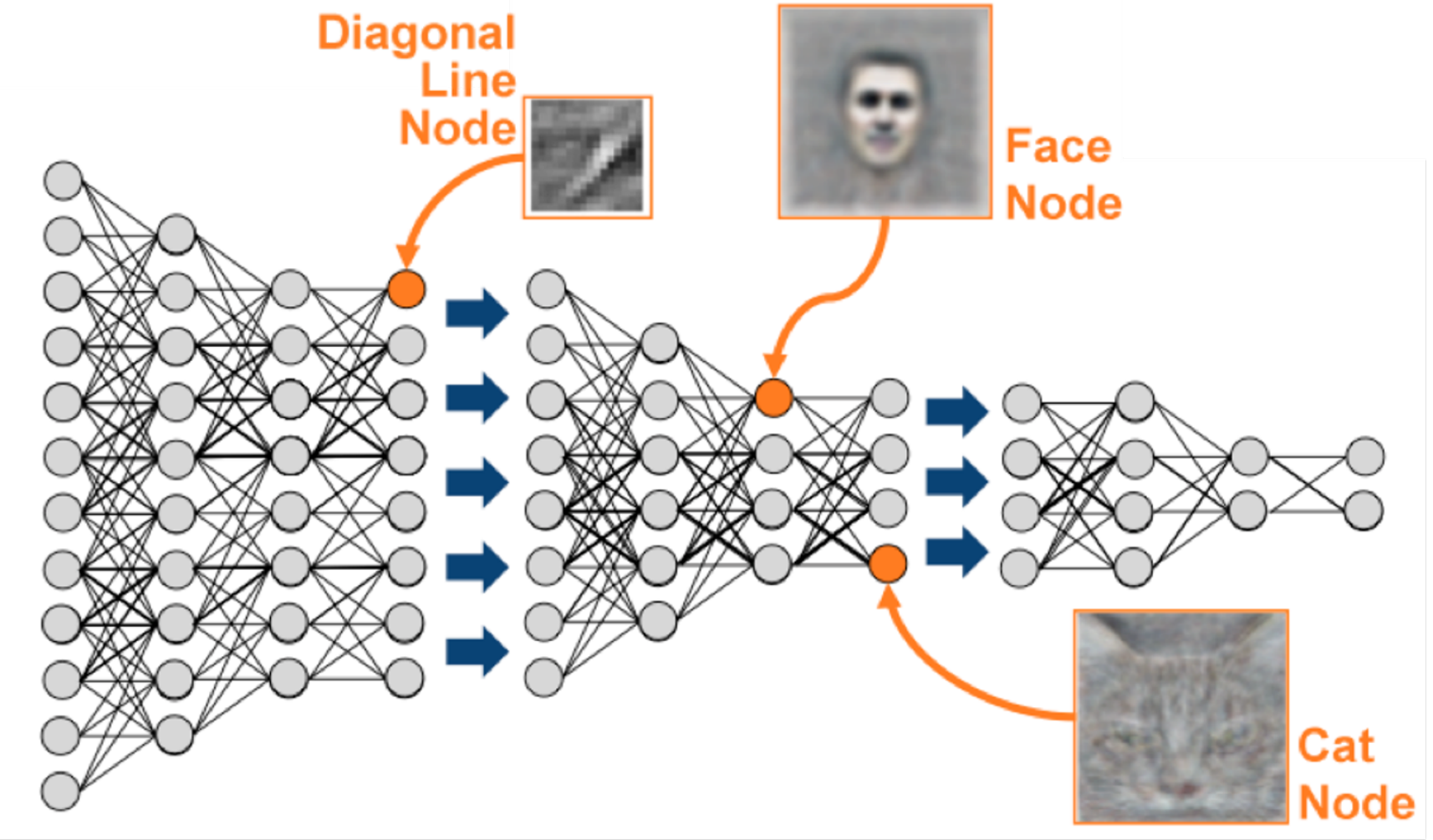
就是層數比較多的Neural Net



Deep Neural Network (2/2)

淺層神經元偵測簡單特徵;深層神經元偵測複雜特徵

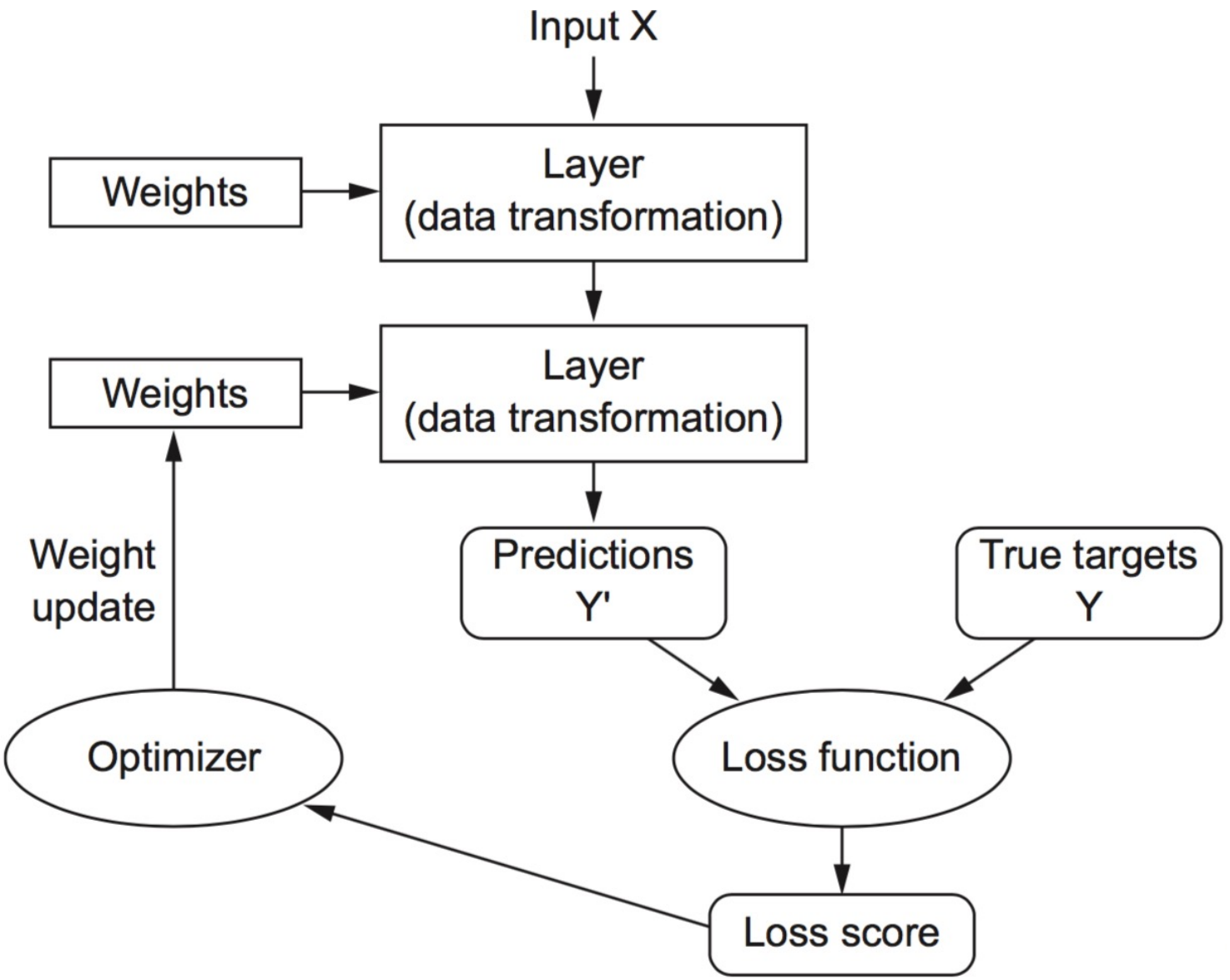
(visual bag of words / basis functions)



解perceptual problems w/o feature engineering

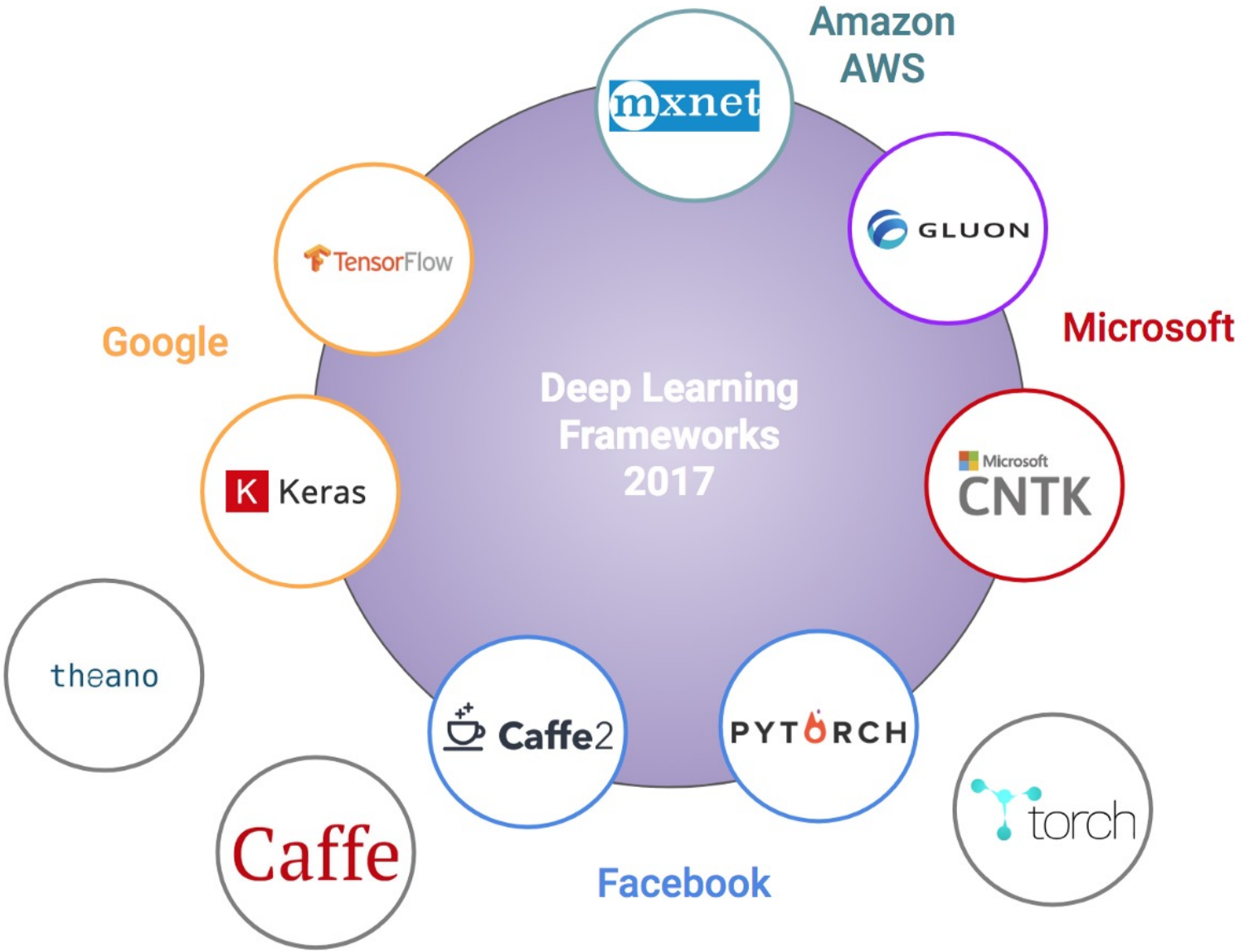
Deep Supervised Learning

和Shallow Supervised Learning程序一樣



Deep Learning Frameworks (2/2)

幾家大公司有各自的架構



Deep Learning Frameworks (2/2)

Python For Data Science *Cheat Sheet*

Keras

Learn Python for data science Interactively at www.DataCamp.com

Keras

Keras is a powerful and easy-to-use deep learning library for Theano and TensorFlow that provides a high-level neural networks API to develop and evaluate deep learning models.

A Basic Example

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random((1000,100))
>>> labels = np.random.randint(2,size=(1000,1))
>>> model = Sequential()
>>> model.add(Dense(32,
                    activation='relu',
                    input_dim=100))
>>> model.add(Dense(1, activation='sigmoid'))
>>> model.compile(optimizer='rmsprop',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
>>> model.fit(data,labels,epochs=10,batch_size=32)
>>> predictions = model.predict(data)
```

Data

Also see NumPy, Pandas & Scikit-Learn

Your data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ideally, you split the data in training and test sets, for which you can also resort to the `train_test_split` module of `sklearn.cross_validation`.

Keras Data Sets

```
>>> from keras.datasets import boston_housing,
                               mnist,
                               cifar10,
                               imdb
>>> (x_train,y_train),(x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2),(x_test2,y_test2) = boston_housing.load_data()
>>> (x_train3,y_train3),(x_test3,y_test3) = cifar10.load_data()
>>> (x_train4,y_train4),(x_test4,y_test4) = imdb.load_data(num_words=20000)
>>> num_classes = 10
```

Other

```
>>> from urllib.request import urlopen
>>> data = np.loadtxt(urlopen("http://archive.ics.uci.edu/
ml/machine-learning-databases/pima-indians-diabetes/
pima-indians-diabetes.data"),delimiter=",")
>>> X = data[:,0:8]
>>> y = data [:,8]
```

Model Architecture

Sequential Model

```
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model2 = Sequential()
>>> model3 = Sequential()
```

Multilayer Perceptron (MLP)

Binary Classification

```
>>> from keras.layers import Dense
>>> model.add(Dense(12,
                    input_dim=8,
                    kernel_initializer='uniform',
                    activation='relu'))
>>> model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
>>> model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
```

Multi-Class Classification

```
>>> from keras.layers import Dropout
>>> model.add(Dense(512,activation='relu',input_shape=(784,)))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(512,activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10,activation='softmax'))
```

Regression

```
>>> model.add(Dense(64,activation='relu',input_dim=train_data.shape[1]))
>>> model.add(Dense(1))
```

Convolutional Neural Network (CNN)

```
>>> from keras.layers import Activation,Conv2D,MaxPooling2D,Flatten
>>> model2.add(Conv2D(32,(3,3),padding='same',input_shape=x_train.shape[1:]))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(32,(3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))

>>> model2.add(Conv2D(64,(3,3), padding='same'))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(64,(3, 3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))

>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

Recurrent Neural Network (RNN)

```
>>> from keras.klayers import Embedding,LSTM
>>> model3.add(Embedding(20000,128))
>>> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
```

Preprocessing

Also see NumPy & Scikit-Learn

Sequence Padding

```
>>> from keras.preprocessing import sequence
>>> x_train4 = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
```

One-Hot Encoding

```
>>> from keras.utils import to_categorical
>>> Y_train = to_categorical(y_train, num_classes)
>>> Y_test = to_categorical(y_test, num_classes)
>>> Y_train3 = to_categorical(y_train3, num_classes)
>>> Y_test3 = to_categorical(y_test3, num_classes)
```

Train and Test Sets

```
>>> from sklearn.model_selection import train_test_split
>>> X_train5,X_test5,y_train5,y_test5 = train_test_split(X,
                                                         y,
                                                         test_size=0.33,
                                                         random_state=42)
```

Standardization/Normalization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(x_train2)
>>> standardized_X = scaler.transform(x_train2)
>>> standardized_X_test = scaler.transform(x_test2)
```

Inspect Model

`>>> model.output_shape`	Model output shape
`>>> model.summary()`	Model summary representation
`>>> model.get_config()`	Model configuration
`>>> model.get_weights()`	List all weight tensors in the model

Compile Model

MLP: Binary Classification

```
>>> model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
```

MLP: Multi-Class Classification

```
>>> model.compile(optimizer='rmsprop',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

MLP: Regression

```
>>> model.compile(optimizer='rmsprop',
                  loss='mse',
                  metrics=['mae'])
```

Recurrent Neural Network

```
>>> model3.compile(loss='binary_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])
```

Model Training

```
>>> model3.fit(x_train4,
               y_train4,
               batch_size=32,
               epochs=15,
               verbose=1,
               validation_data=(x_test4,y_test4))
```

Evaluate Your Model's Performance

```
>>> score = model3.evaluate(x_test,
                            y_test,
                            batch_size=32)
```

Prediction

```
>>> model3.predict(x_test4, batch_size=32)
>>> model3.predict_classes(x_test4,batch_size=32)
```

Save/ Reload Models

```
>>> from keras.models import load_model
>>> model3.save('model_file.h5')
>>> my_model = load_model('my_model.h5')
```

Model Fine-tuning

Optimization Parameters

```
>>> from keras.optimizers import RMSprop
>>> opt = RMSprop(lr=0.0001, decay=1e-6)
>>> model2.compile(loss='categorical_crossentropy',
                   optimizer=opt,
                   metrics=['accuracy'])
```

Early Stopping

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4,
               y_train4,
               batch_size=32,
               epochs=15,
               validation_data=(x_test4,y_test4),
               callbacks=[early_stopping_monitor])
```

DataCamp
Learn Python for Data Science Interactively

Game Over