Tensorflow Training Class

Joel luo 羅靖淵

- 羅靖淵 (JingYuan Luo) – Joel
  - E-mail: Joel.jy.luo@gmail.com
  - Linkedin: https://www.linkedin.com/in/joel-jy-luo/

- EDUCATION
  - **Master of Science - Sep. 2015 ~ Nov. 2017**
    - National Cheng Kung University Tainan, Taiwan - Computer Science and Information Engineering
  - **Bachelor of Science - Sep. 2011 ~ Jun. 2015**
    - Chung Yuan Christian University Taoyuan, Taiwan - Information and Computer Engineering

- EXPERIENCE
  - **Senior Software Engineer – Jan. 2018 ~ Present**
Delta Electronics, Inc. - Taoyuan, Taiwan

Machine Learning – Day 1

Recommend Books

- Introducing Python - Modern Computing in Simple Packages (2014)
  - Link
- A Practical Introduction to Python Programming
  - Link
- NumPy 高速運算徹底解說：六行寫一隻程式？你真懂深度學習？手工算給你看！
  - Link
- TensorFlow+Keras深度學習人工智慧實務應用
  - Link
- Google Teacher
  - 學習如何Google, 並且學會查詢第一手資料.

How do we learn?

- **The First 20 Hours: How to Learn Anything... Fast** - Kaufman, Josh
  - https://youtu.be/5MgBikgcWnY

1. Deconstruct the skill
2. Learn enough to self-correct
3. Remove practice barrier
4. Practice at least 20 Hours

How to learn Machine Learning

1. Deconstruct the skill
   - Find important skill, EX:
     python, **Linux basic cmd**, **numpy, Math** …. etc.

2. Learn enough to self-correct
   - Use these skills to find more info for correcting/verifying self idea.

3. Remove practice barrier
   - Keep focus.
   - Turn off your phone, TV…etc. Anything may distract your attention.

4. Practice at least 20 Hours
   - More practice.
   - Coding, Coding and Coding.

What is Machine Learning? (1)

1. Collecting data

2. Filter the properties and features of data

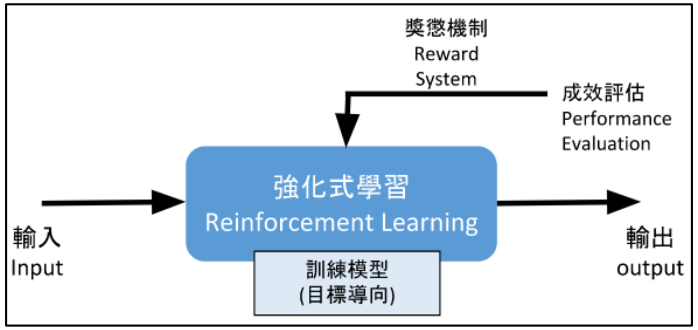
3. 利用資料屬性/特徵找出規則或分類

4. 預測未來資料/使用該規則創造新資料

What is Machine Learning? (2)

1. 監督式學習(Supervised Learning)
   - 一比一對照資訊
     - 學習過程進行分類(Classification): True/false
     - 學習過程進行回歸(Regression): predict value

2. 強化式學習(Reinforcement Learning)
   - 在未知探索與遵從既有知識間取得平衡
     - 學習過程只告訴你結果是好或不好, 沒有正確解答

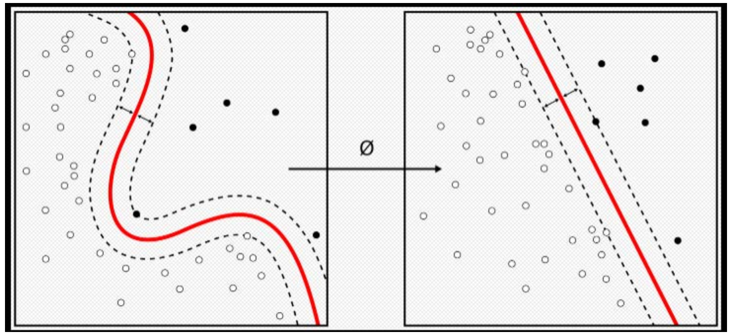


3. 非監督式學習(Unsupervised Learning)
   - 機器自行摸索出資料規律
     - 學習過程讓模型自由發揮, 最後看結果

資料引用於: 網址連結

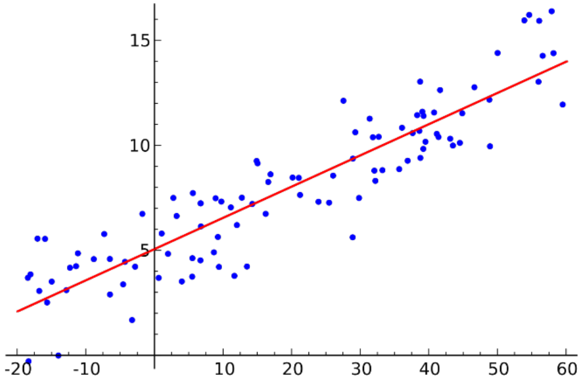
Machine Learning - Supervised Learning (1)

1. 分類(Classification)
   - 畫一條線進行分類
   - y = ax + b; y’ > 0 or y’ < 0



圖片來源: wiki

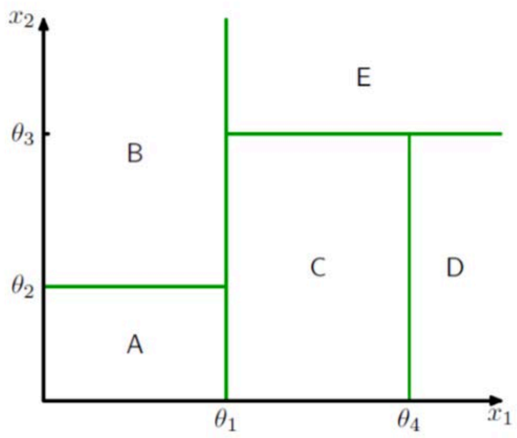
2. 回歸(Regression)
   - 找出一條最接近的直線(Linear regression)
   - y = ax + b ; | y – y’ | ≈ 0



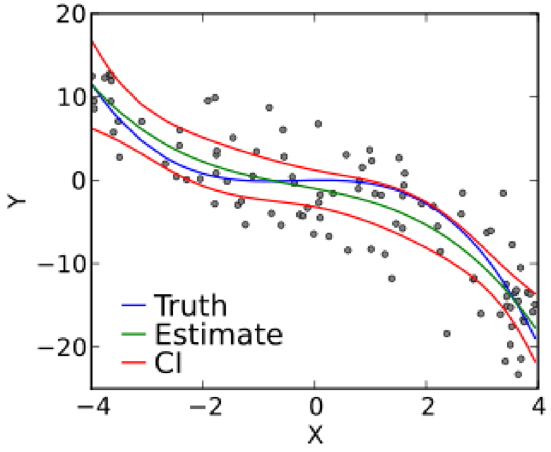
圖片來源: wiki

Machine Learning - Supervised Learning (2)

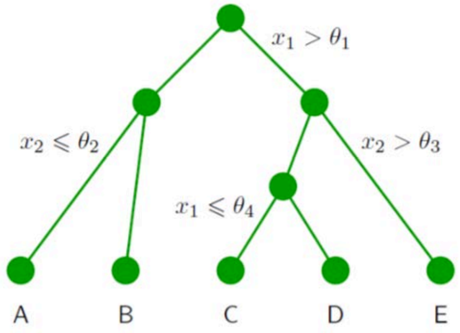
1. 分類(Classification)
   1. SVM (Support Vector Machine) – 分兩類
   2. Decision tree – 兩類再分兩類, 類似binary tree



2. 回歸(Regression)
   1. Linear regression - 找出一條最接近的直線
   2. Polynomial regression- 找出一條最接近的曲線



圖片來源: wiki



圖片來源: Link

Machine Learning - Supervised Learning (3)

- 分類(Classification) – 應用情境
  - Face Recognition (臉部辨識)

    [PHILLIPS, P. Jonathon. Support vector machines applied to face recognition. In: *Advances in Neural Information Processing Systems*. 1999. p. 803-809.]

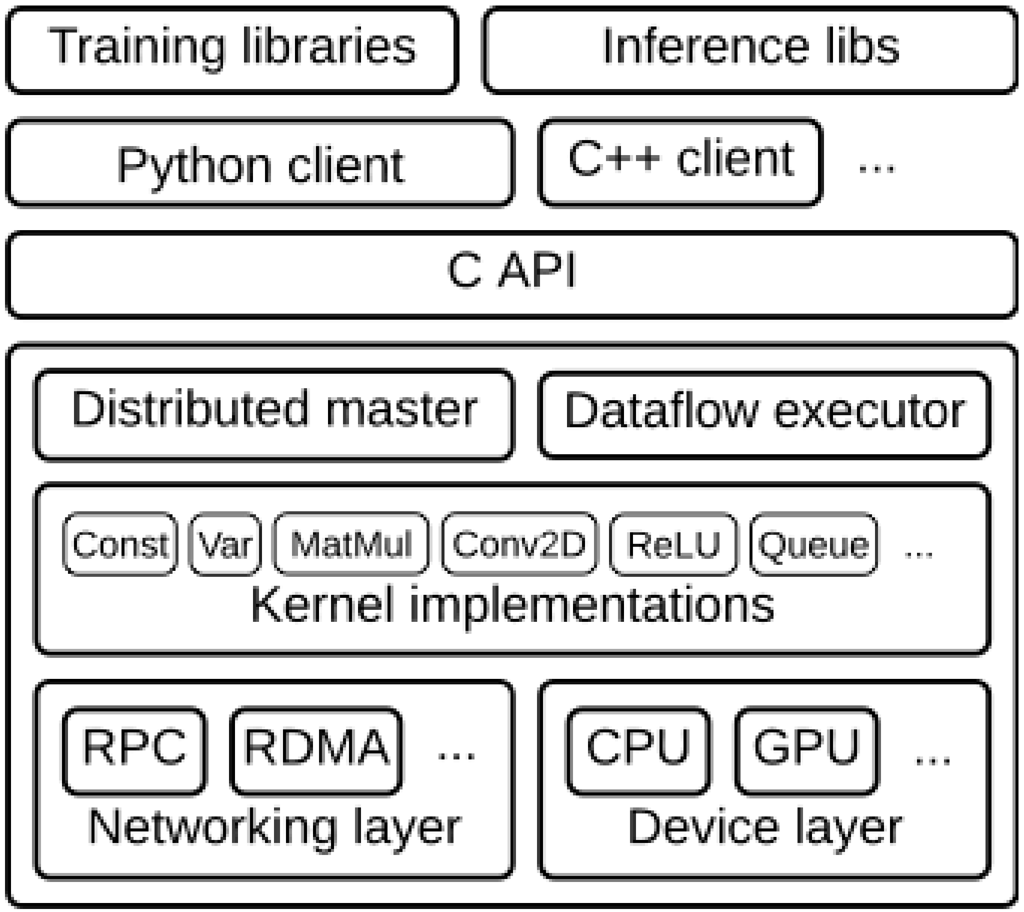
- 回歸(Regression) – 應用情境
  - Stock Prediction / Forecasting 股票預測

    [ALTAY, Erdinç; SATMAN, M. Hakan. Stock market forecasting: artificial neural network and linear regression comparison in an emerging market. Journal of Financial Management & Analysis, 2005, 18.2: 18.]

自我思考: 請各舉一個分類/回歸的使用情境(2~3人一組)
[嘗試透過尋找Paper文獻取得第一手知識]

Machine Learning Framework – Tensorflow

- **Application (Front-end)**
  - We only focus on here
- Application Programming Interface (API)
- Tensorflow Kernel
  - Execute Engine
- Hardware Device
  - CPU / GPU / TPU / FPGA



Tensorflow Architecture (Link)

Setup Develop Environment

1. Environment (Operation System) – Linux
   - Linux Distribution – Ubuntu
   - 基本系統操作指令ls, cp, cd, mv, mkdir, vim ....
   - Linux 基本功: 基礎學習, 基礎訓練

2. Simulation Tool (Virtual Machine) – Virtual Box
   - Download URL – VirtualBox 6.0.22 (released May 15 2020): Link
   - 載入系統映像檔

3. Machine Learning Framework – Tensorflow
   - 系統環境套件管理佈署: Anaconda
   - 版本: tensorflow-cpu 1.14.0
   - 程式語言: python-3.5+
   - 搭配套件: Numpy-1.16.4, Pandas-1.0.1

Setup Develop Environment

- Startup Lab from VM
  1. Open Virtual Box
  2. Load System Image
  3. Login Linux
     - Account: cycu-lab715, Password: 123456
  4. Go to Directory /home/cycu-lab715/Workspace/tf_lesson_lab
  5. Open Sublime to view lab1 sample code
  6. Try run Lab1
     - python main.py

Homework-1

- 1. Use VM Environment (Command line) to create a Project Directory:
  - HW_01/
    - | - C_Program
      | - main.c
    - | - Python_Program
      | - main.py

- 2. Implement a star printer
  - Execute program: ./main [Tips: How to compile main.c ?]
  - Output:

```
----*----
---***---
--*****--
-*******-
*********
```

- 3. Use python to implement star printer
  - Execute program: python main.py

Machine Learning – Day 2

Review

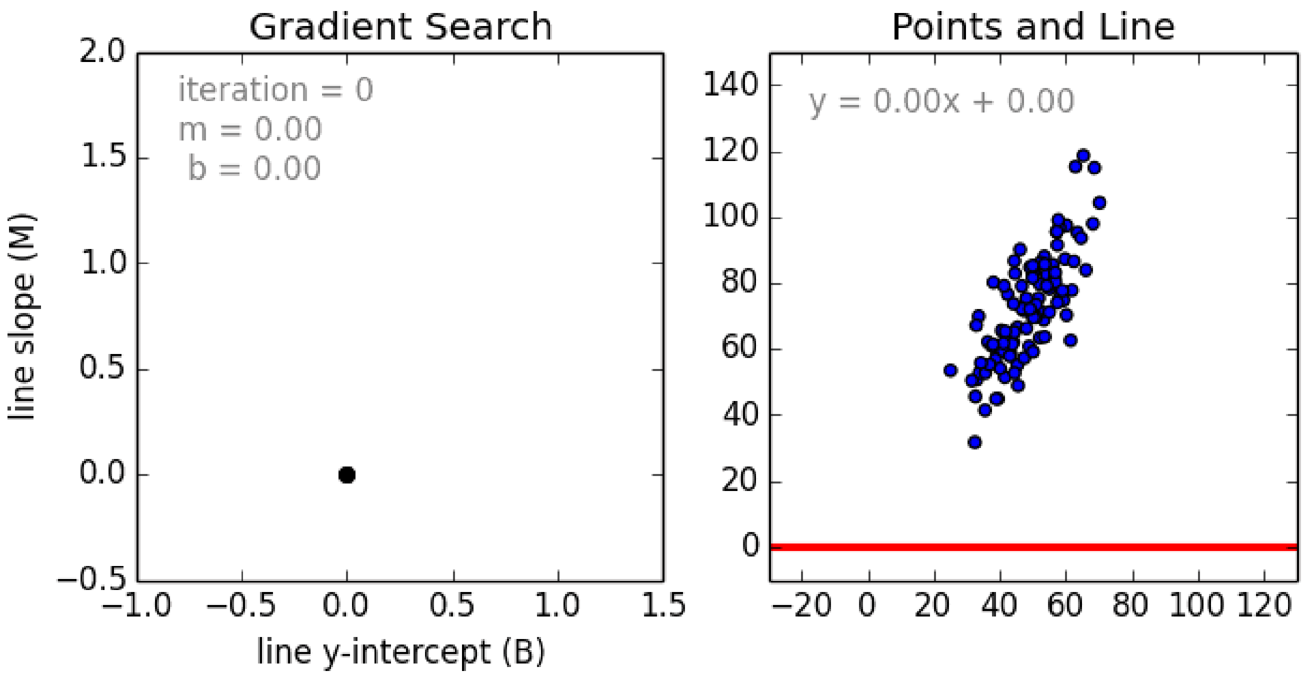
1. 請各舉一個分類的使用情境

2. 請各舉一個回歸的使用情境

Lab1: Basic Linear Regression (1)

- Loss function (評估近似度或準確度)
  1. **Mean square error - MSE**
  2. Root MSE – RMSE
  3. cross-entropy

- Optimizer (找出近似解方法)
  1. **gradient descent**
  2. Backpropagation
  3. AdamOptimizer

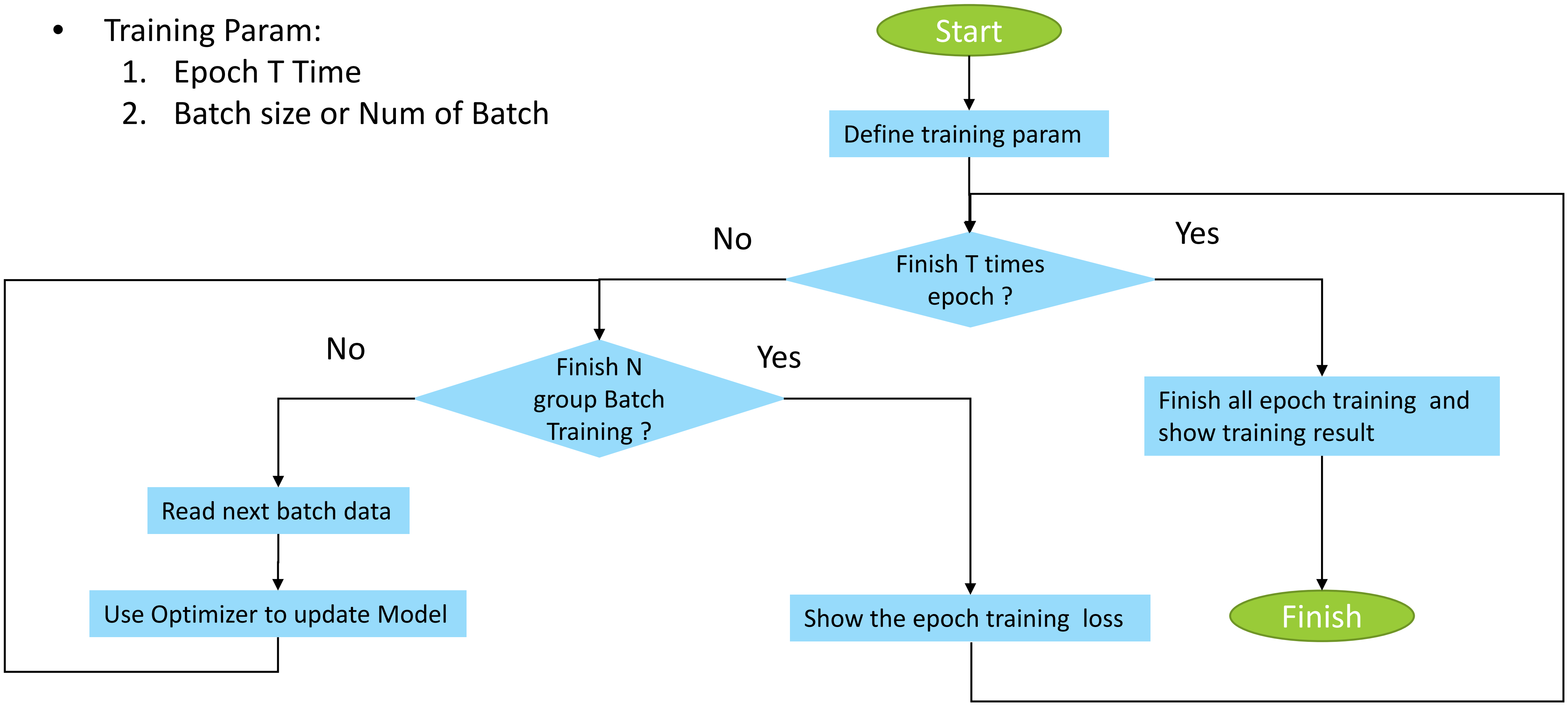


圖片來源: Link

課程科目:
1. 微積分
2. 工程數學
3. 機率統計

Lab1: Basic Linear Regression (2)

- Traing Graphic Flow Chart
  - Training Param:
    1. Epoch T Time
    2. Batch size or Num of Batch



Lab1: Basic Linear Regression (3)

- Tensorflow training flow
  1. Get original data
  2. Generate Input Data / Output Data
  3. Create Tensorflow training graphic (Neural Network)
  4. Traing Graphic
     1. Define training **epoch** T times (Repeat training times)
     2. Define **batch size** or **number of batch** (divide data into N group)
     3. Execute training epoch
        - Each epoch run **N batch** of training data

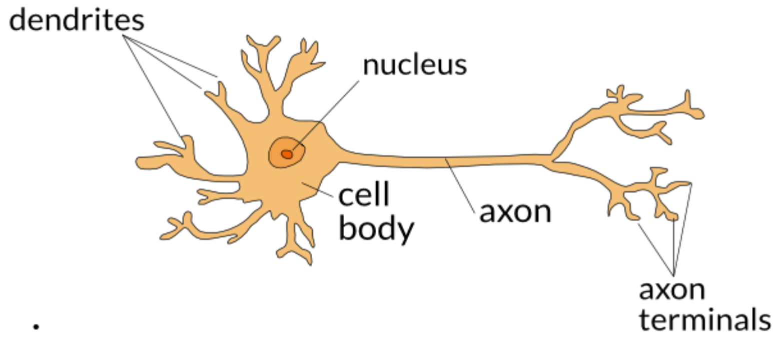
Lab1: Basic Linear Regression (4)

- Code Review
  - Path:
    - /lesson_01/Tensorflow_Sample_01_main.py
  - Main idea
    - Create simple model to demonstrate tensorflow working flow.
    - Train a model to preset linear formula. [y = 9.5*x + 2.7].

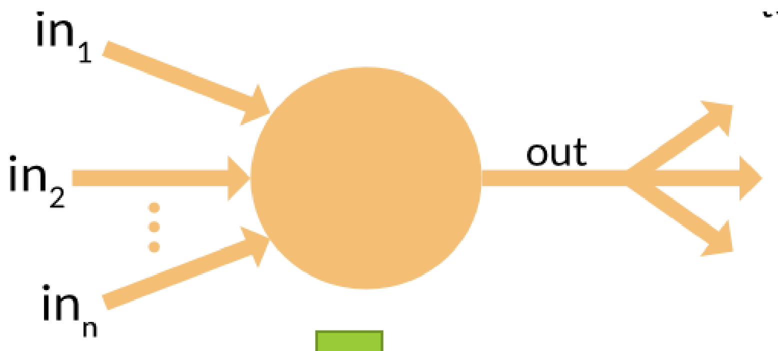
Basic Neural Network (1)

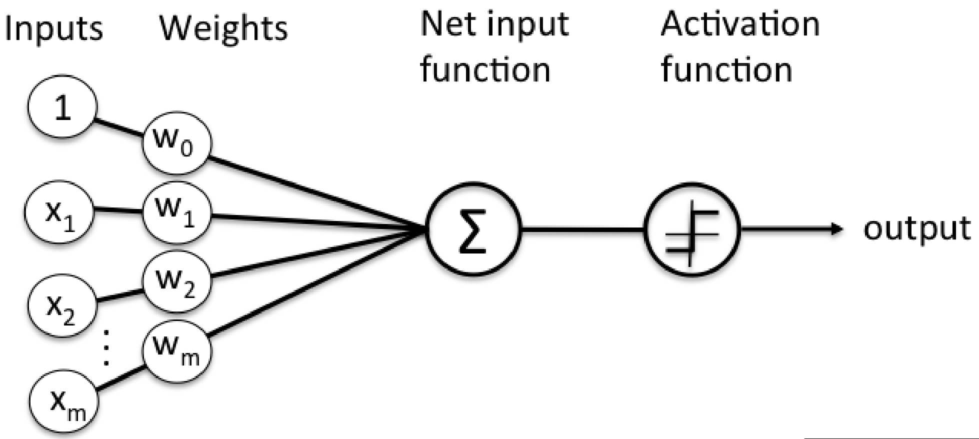
圖片來源: Link1, Link2, Link3

- Neuron V.S Perceptron



Neuron





Perceptron

$$y = f(w_{10} + w_{11}x_{1} + w_{12}x_2 + \cdots + w_{1d}x_d) = f(\boldsymbol{W}_1^T\boldsymbol{x} + w_{10})$$

$$f = \begin{cases} 1, & \boldsymbol{W}_1^T\boldsymbol{x} + w_{10} > 0 \\ 0, & O.W. \end{cases}$$

$$\boldsymbol{W}_1 = \begin{bmatrix} w_{11} \\ \vdots \\ w_{1d} \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

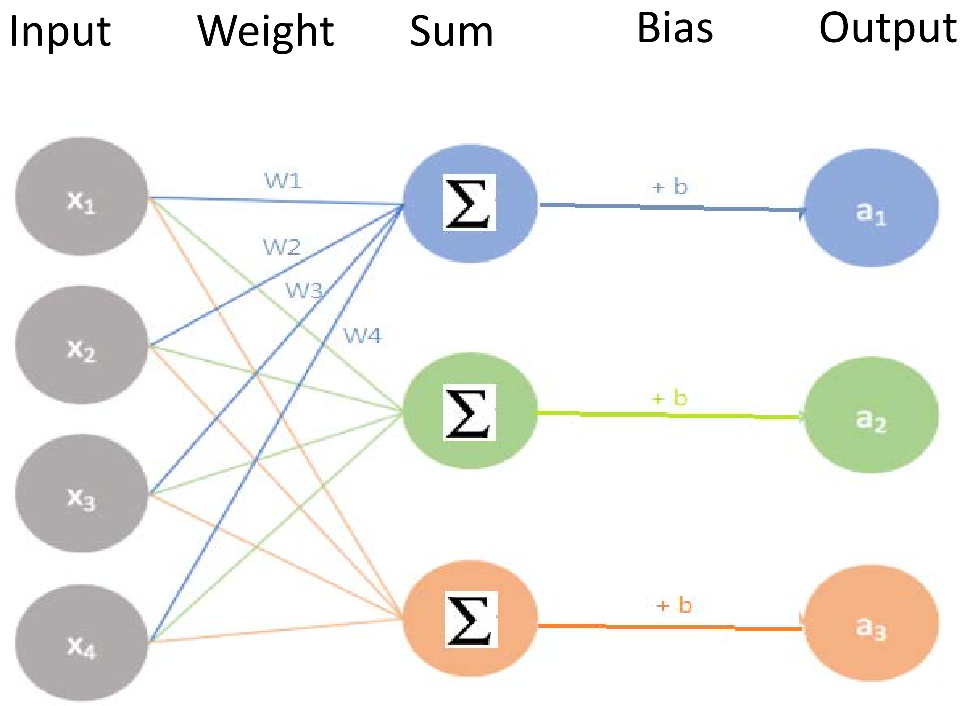
Perceptron Formula

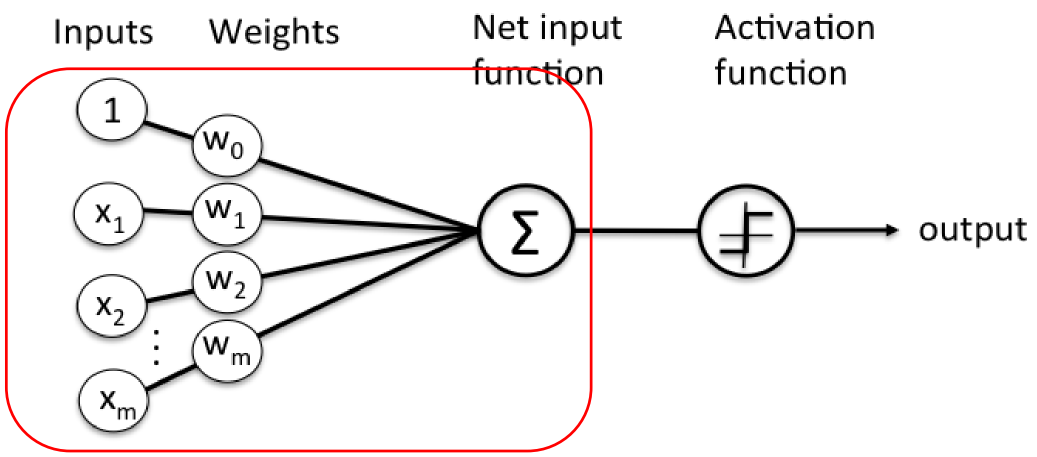
課程科目:
1. 線性代數

Basic Neural Network (2)

- Basic mathematical concept of Perceptron
  1. Matrix Calculation
  2. Active function
  3. Loss function
  4. Normalize

Matrix Calculation

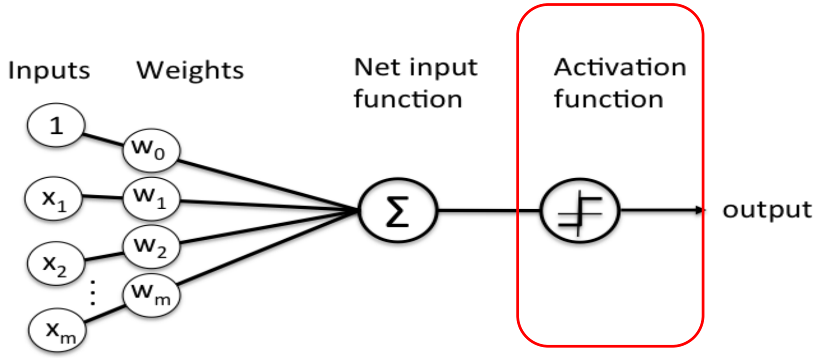




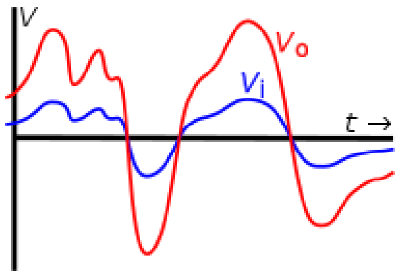
$$\underbrace{\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix}}_{\text{Weight}} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \underbrace{\begin{bmatrix} b \\ b \\ b \end{bmatrix}}_{\text{Bias}} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \rightarrow \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

課程科目:
1. 線性代數

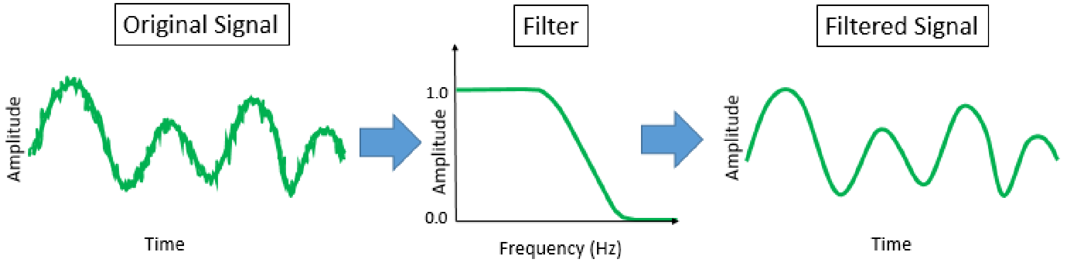
Active Function(1)



- 核心觀念:
  - 訊號處理(Signal Processing)
    1. 放大/縮小 (Amplifier)



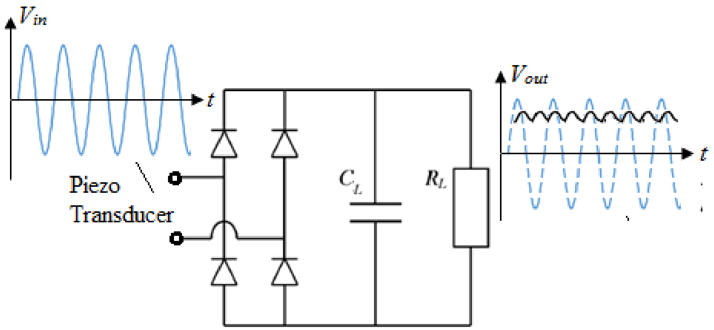
    2. 過濾(Filter)



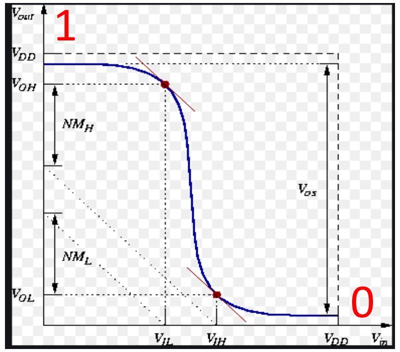
圖片來源

延伸:

- 類比訊號處理 => 電子電力(AC2DC) / 聲頻



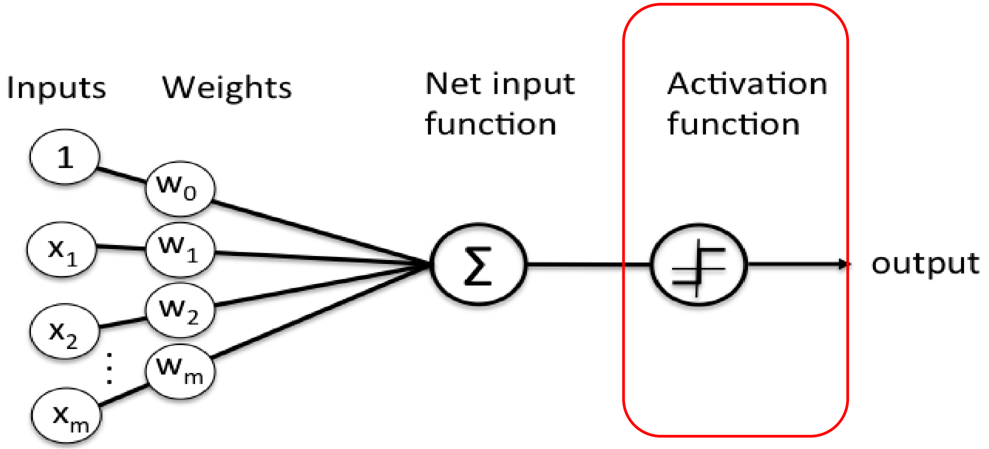
- 數位訊號處理 => 晶片訊號(0/1)/ 邏輯閘

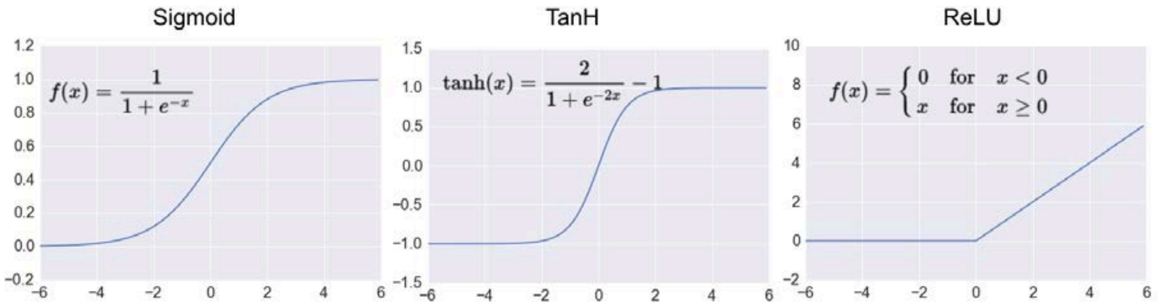


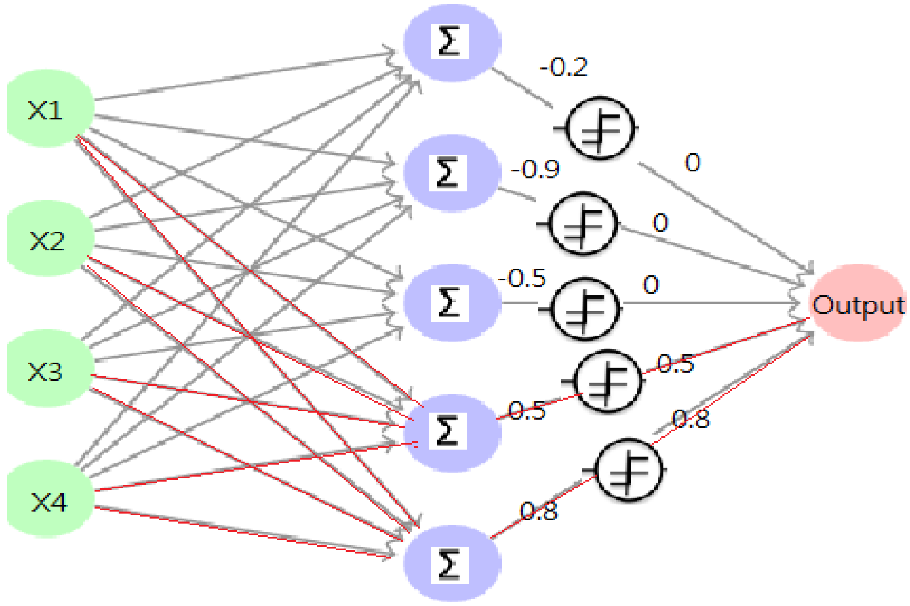
課程科目:
1. 電子學
2. 電路學
3. 邏輯設計
4. 數位訊號處理

Active Function(2)

- 應用於ML
  - 強化權重(選擇/過濾)資料
- 概念
  - 決定參考資料X的重要性
  - 強化因果關係相依性







課程科目:
1. 微積分
2. 工程數學

主流使用Active Function (圖片來源)

Active Function 以ReLU為例

Loss Function

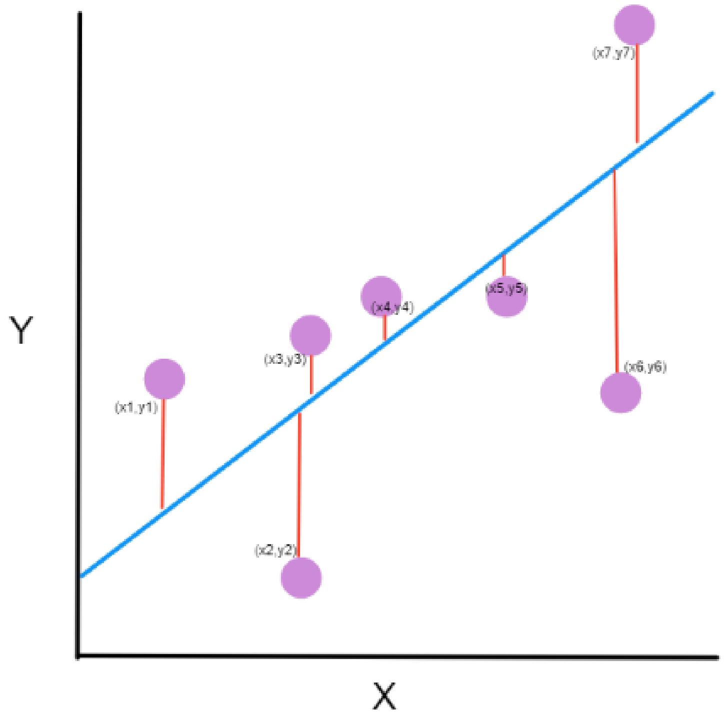
課程科目:
1. 機率統計

- 核心概念 - 評估資料**近似度**或**準確度**
  1. **Mean square error - MSE**
  2. Root MSE – RMSE
  3. cross-entropy

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

參考來源

- **MSE/RMSE**
  - 取**目標值**與**預測值**差平方計算平均
  - 如右圖紅線平方, 加總平均. (RMSE僅MSE 開根號)
  - 意義:
    - 如MSE/RMSE 越小表示預測函數越接近實際資料分布

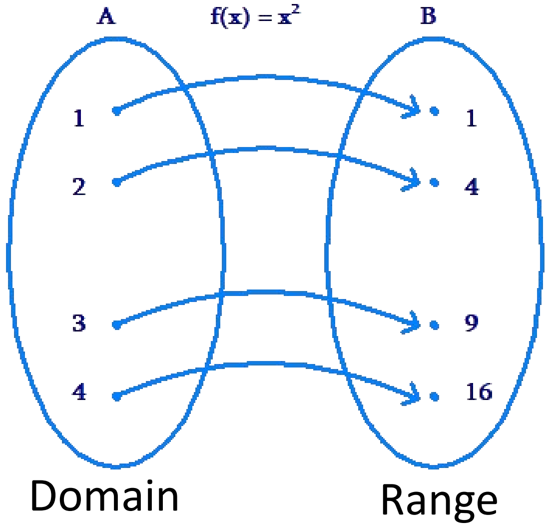


圖片來源

- **Cross-entropy**
  - 新增考慮機率關係,
    - MSE 依據平均分布, 平均所有誤差
    - cross-entropy採取機率統計方式**忽略低頻率誤差數據**

Normalize(1)

- 核心觀念:
  - 將數據範圍參照某個標準轉換至另一個映射空間範圍
    - 如右圖: A 參照 $f(x) = X^2$ 轉換至 B

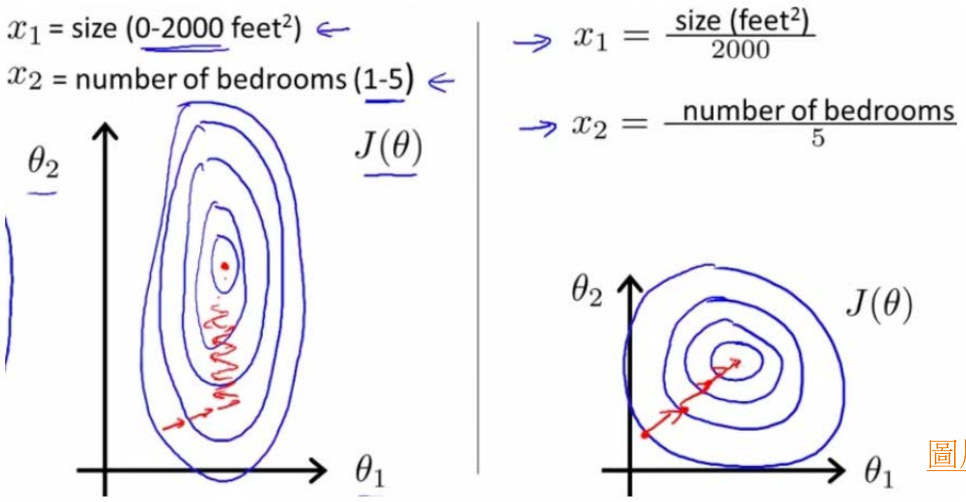


圖片來源

- 目的:
  - 針對真實資料進行前處理(Preprocessing)

1. 因Model 有時無法直接接受Real data,
   必須透過normalize進行資料映射

2. 因真實資料數據值間隔太大,
   必須透過normalize 進行縮小domain,
   優化Optimizer找出最佳解的效率,
   縮短訓練時間



圖片來源

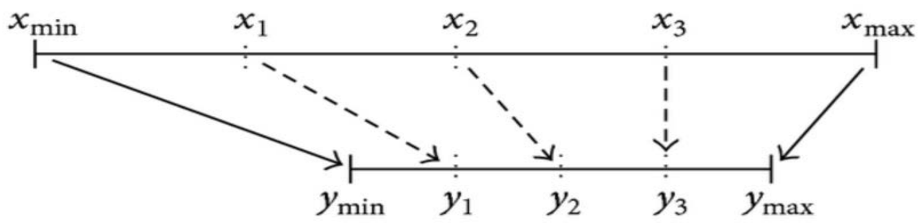
課程科目:
1. 機率統計
2. 離散數學
3. 線性代數

資料來源

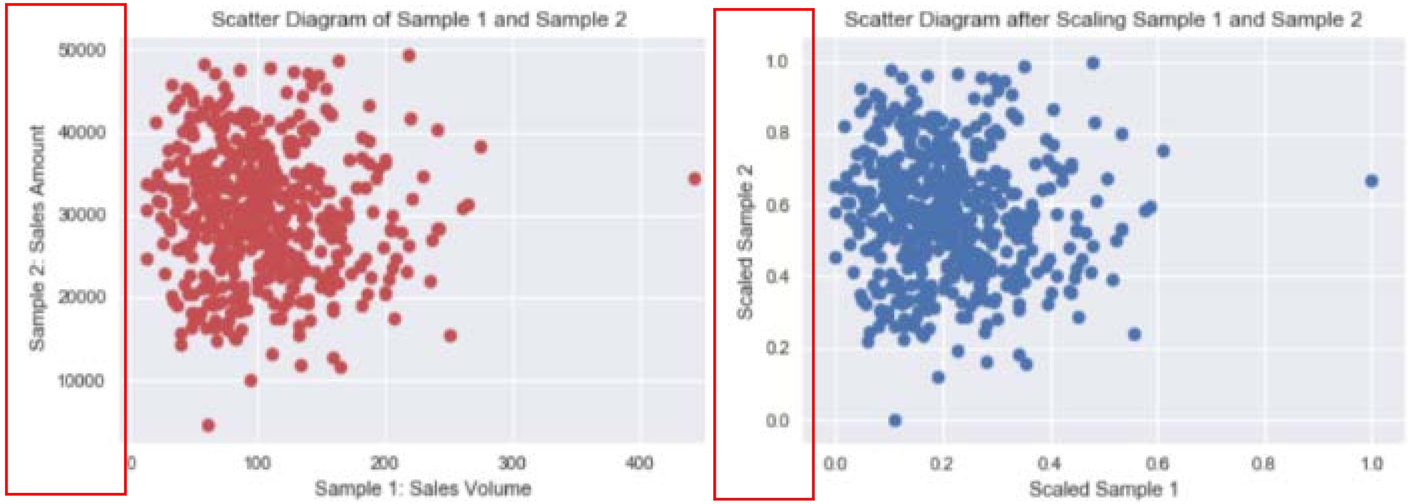
Normalize(2)

- Feature scaling

1. min-max normalization



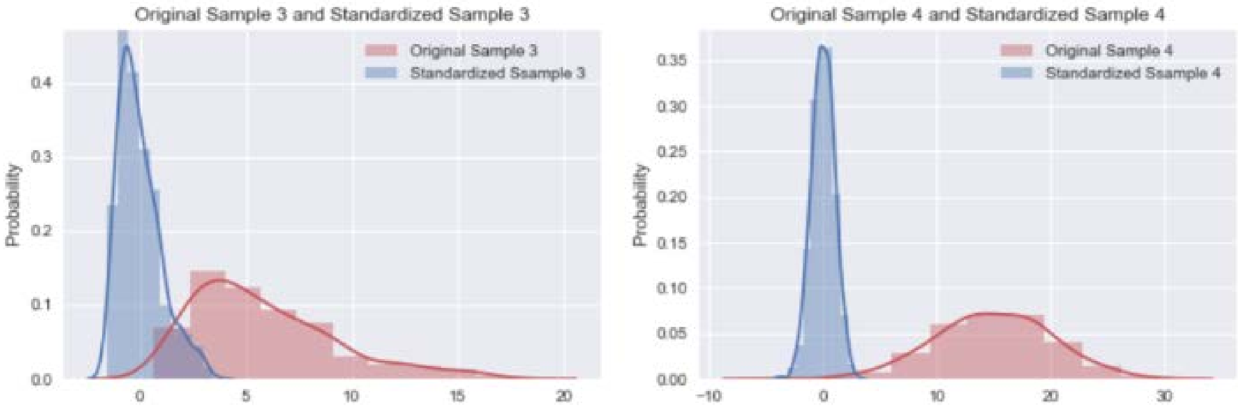
$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$



- 依據Min-Max範圍縮小domain

2. Z-score Normalization

$$Z = \frac{X - \mu}{\sigma} \sim N(0,1)$$



- 加入Normal Distribution特性,進行Normalize,以降低離群數據影響映射分布

Lab2: Basic Neural Network(1)

1. Read Training and Test data.
   - Read from CSV
2. Normalize all data. (Training and Test data)
   - Use Min-Max normalize
3. Split Input / Output data from data normalized
   - Get training dataset

→ Preprocessing dataset

4. Create Neural Network structure

→ Tensorflow training graphic

5. Define Training Epoch and Run

→ Training and Verify result

Lab2: Basic Neural Network(2)

- Code Review
  - Path:
    - /lesson_02/Tensorflow_Sample_02_main.py
  - Main idea
    - Create simple model to demonstrate tensorflow working flow.
    - Train a model to preset linear formula. [y = 9.5*x + 2.7].
    - Use normalize methodology to scale(transform) data in range(-1, 1).
    - Use MSE as loss function to do training
    - Use the model trained during step 2 to predict 20 pair of data.
    - Use RMSE to verify the prediction results.

Machine Learning – Day 3

Review

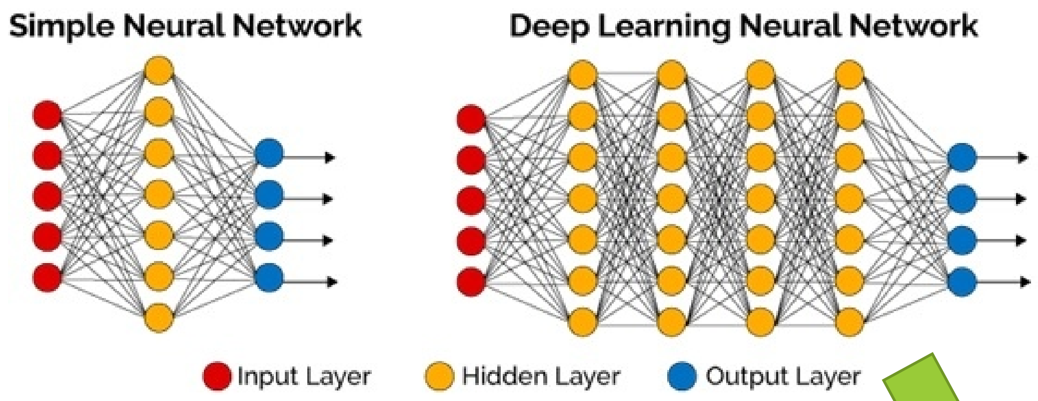
1. Matrix Calculation

2. Active function

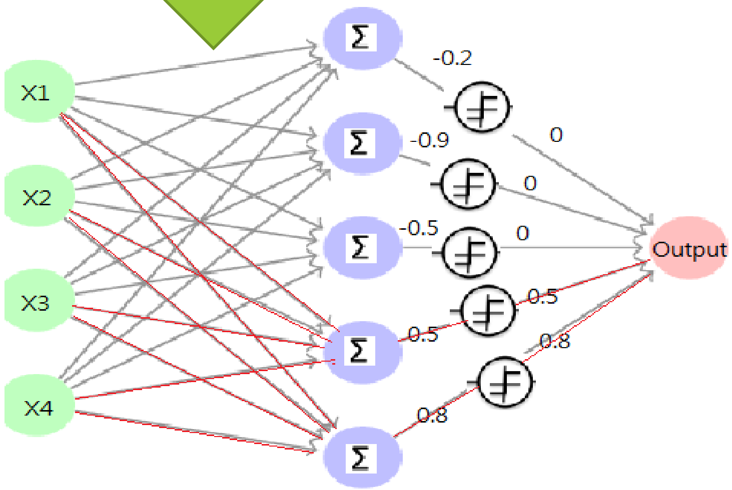
3. Loss function

4. Normalize

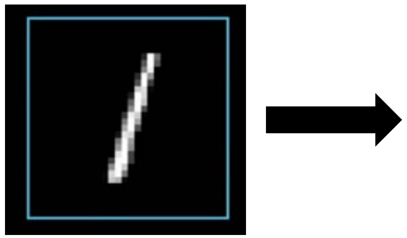
Deep Neural Network(1)

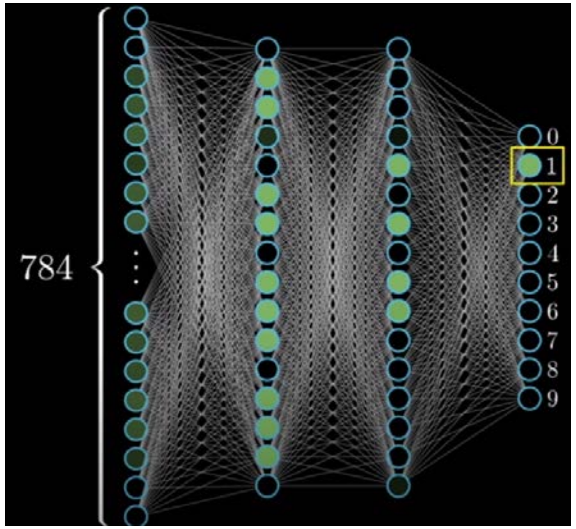


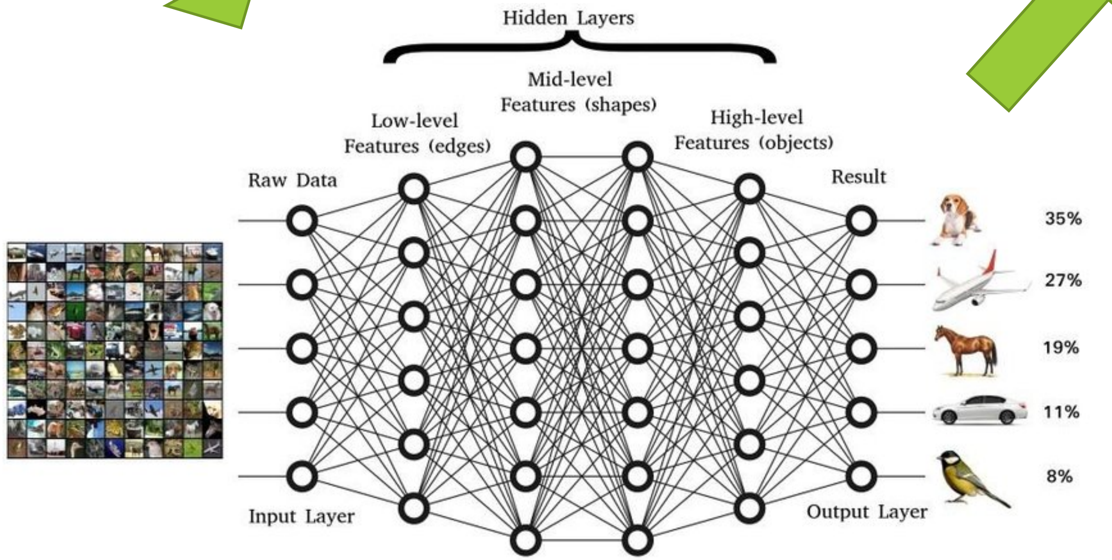
圖片來源



YouTube – Neural Network 簡介







圖片來源

Lab3: Deep Neural Network(1)

1. Read Training and Test data.
   - Read from CSV
2. Normalize all data. (Training and Test data)
   - Use Min-Max normalize
3. Split Input / Output data from data normalized
   - Get training dataset

→ Preprocessing dataset

4. Create Neural Network structure (2 layer DNN, MLP)

→ Tensorflow training graphic

5. Define Training Epoch and Run

→ Training and Verify result

Lab3: Deep Neural Network(2)

- Code Review
  - Path:
    - /lesson_03/Tensorflow_Sample_03_main.py
  - Main idea
    - Extend the Lab 02, create more complex NN model.
    - Demonstrate that how to build multi-layer neural network.

Lab4: Save/Restore Training Model (1)

- Code Review
  - Path:
    - /lesson_04/Tensorflow_Sample_04_main.py
  - Main idea
    - Use **tf saver** module to demonstrate saving trained model as metadata file.
    - Use **tf saver** module to demonstrate restoring model from metadata file.

Homework-2

1. **DNN Full Connection** - Change 2 layer as N layer
   - N is a var.
   - Tips: use loop to create Deep Neural Network

2. **Digit Recognition**
   - Change Input data - mnist dataset
   - Loss function: **cross_entropy**
   - Optimzer: **AdamOptimizer**

YouTube - DataSet visualization

YouTube - Neural Network 3D Simulation

圖片來源 / sample code參考

Classifiers [edit]

This is a table of some of the machine learning methods used on the database and their error rates, by type of classifier:

Type	Classifier	Distortion	Preprocessing	Error rate (%)
Linear classifier	Pairwise linear classifier	None	Deskewing	7.6[9]
Decision stream with Extremely randomized trees	Single model (depth > 400 levels)	None	None	2.7[21]
K-Nearest Neighbors	K-NN with non-linear deformation (P2DHMDM)	None	Shiftable edges	0.52[22]
Boosted Stumps	Product of stumps on Haar features	None	Haar features	0.87[23]
Non-linear classifier	40 PCA + quadratic classifier	None	None	3.3[9]
Random Forest	Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC)[24]	None	Simple statistical pixel importance	2.8[25]
Support-vector machine (SVM)	Virtual SVM, deg-9 poly, 2-pixel jittered	None	Deskewing	0.56[26]
Deep neural network (DNN)	2-layer 784-800-10	None	None	1.6[27]
Deep neural network	2-layer 784-800-10	Elastic distortions	None	0.7[27]
Deep neural network	6-layer 784-2500-2000-1500-1000-500-10	Elastic distortions	None	0.35[28]
Convolutional neural network (CNN)	6-layer 784-40-80-500-1000-2000-10	None	Expansion of the training data	0.31[16]
Convolutional neural network	6-layer 784-50-100-500-1000-10-10	None	Expansion of the training data	0.27[29]
Convolutional neural network	Committee of 35 CNNs, 1-20-P-40-P-150-10	Elastic distortions	Width normalizations	0.23[11]
Convolutional neural network	Committee of 5 CNNs, 6-layer 784-50-100-500-1000-10-10	None	Expansion of the training data	0.21[18][19]
Random Multimodel Deep Learning (RMDL)	10 NN-10 RNN - 10 CNN	None	None	0.18[20]
Convolutional neural network	Committee of 20 CNNS with Squeeze-and-Excitation Networks[30]	None	Data augmentation	0.17[31]

資料來源

Machine Learning – Day 4

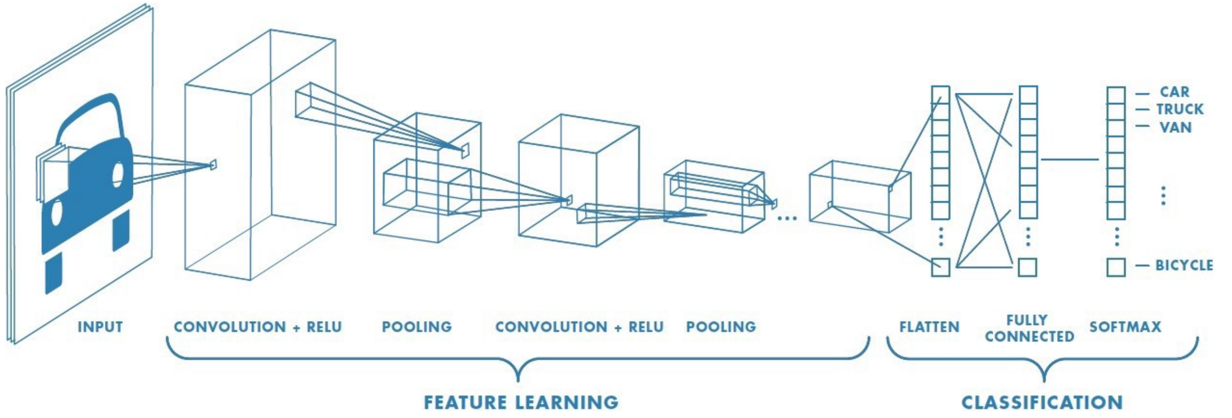
Neural Network Architecture(1)

- Most popular network arch:
  - DNN – Deep Neural Network
  - CNN - Convolutional Neural Network
    - **Computer vision:** Image recognition, Video analysis,
    - **Audio:** Natural language processing
  - RNN - Recurrent Neural Network
    - Speech recognition, Time series prediction, Rhythm learning, Machine translation
  - LSTM - Long short-term memory
    - One of RNN Architecture

Neural Network Architecture – CNN (1)

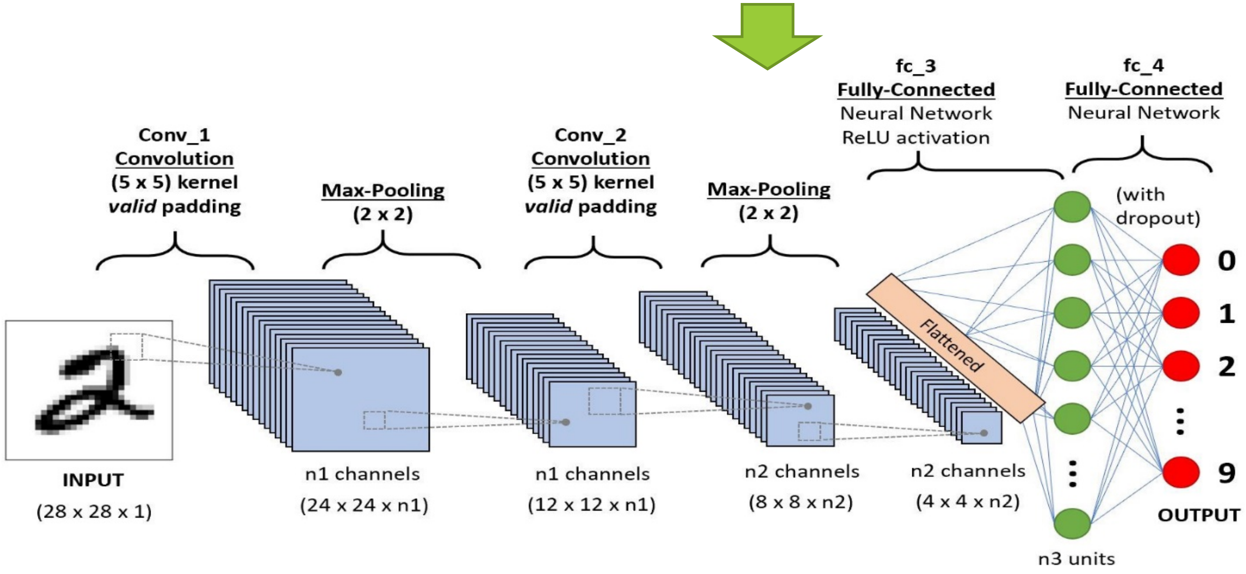
- Convolutional Neural Network – Architecture

Source



- Combine two components:

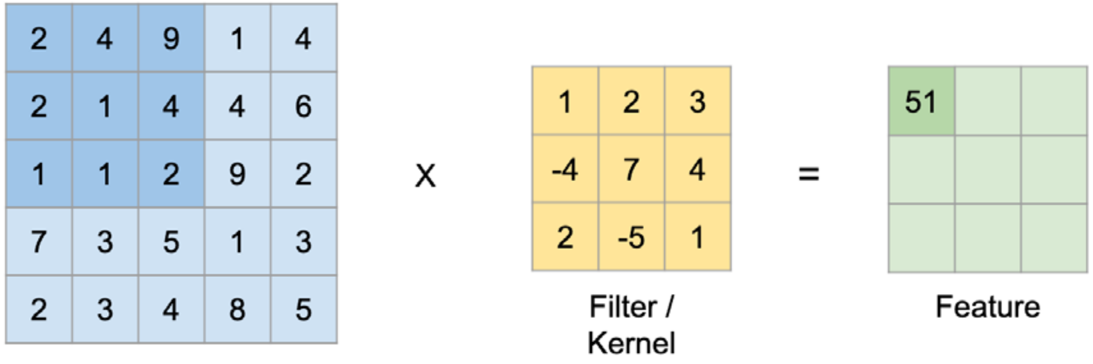
1. Convolution operation method
   - To get more feature of inputs



2. Deep Neural Network

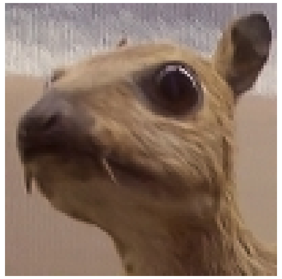
Neural Network Architecture – CNN (2)

- Convolution Operation - (Computer Vision)



Source

Input image

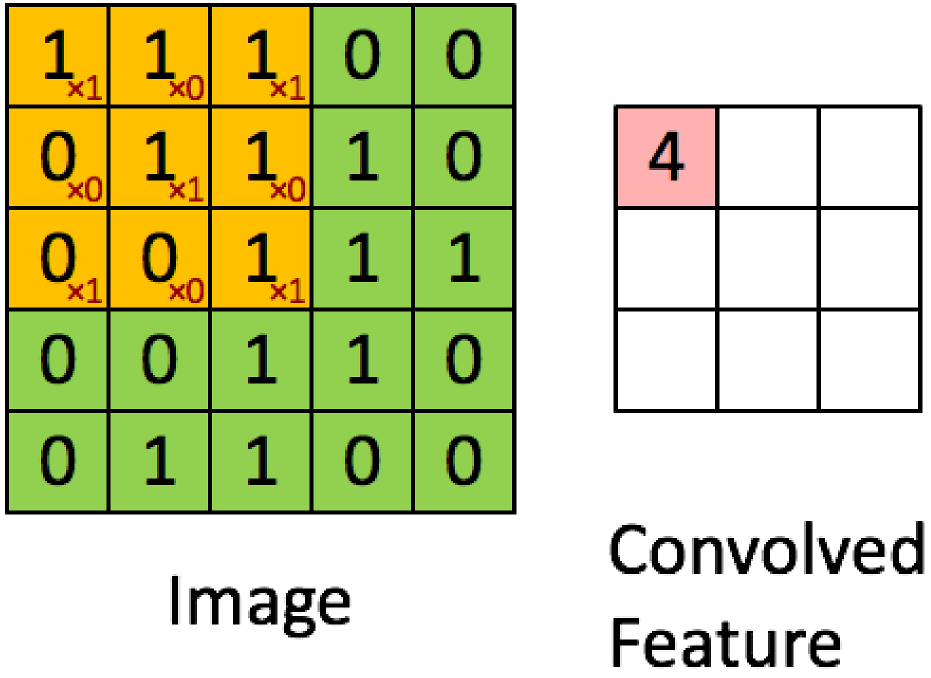


Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

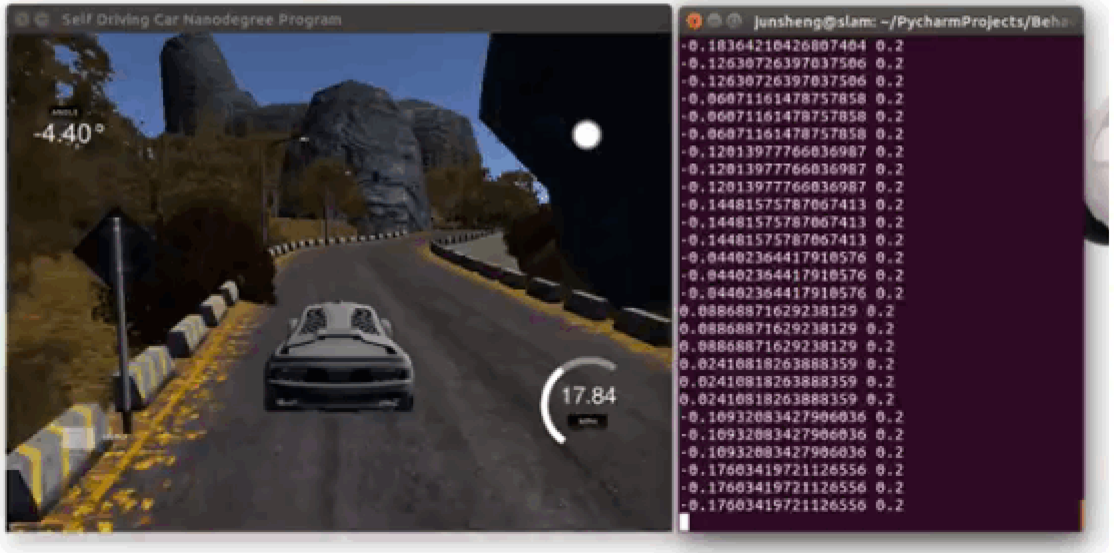
Feature map

Source



Neural Network Architecture – CNN (3)

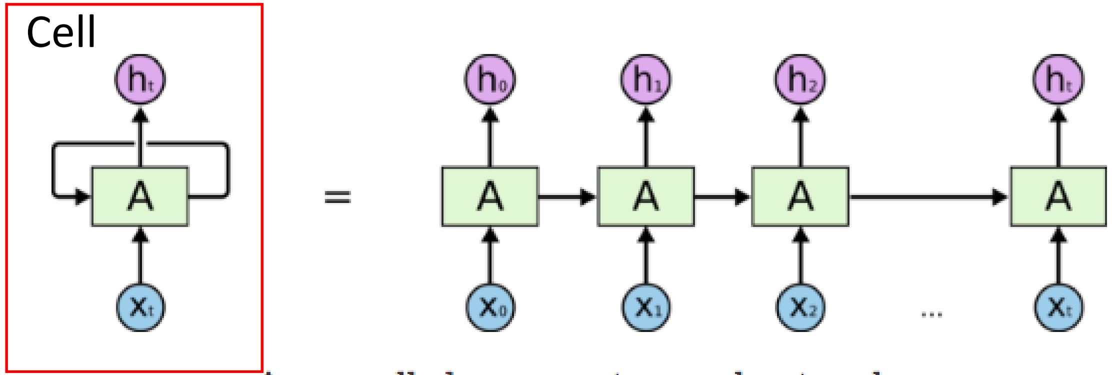
- Convolutional Neural Network – Application
  - Autonomous Driving with a Convolutional Neural Network
    - Source - https://github.com/JunshengFu/driving-behavioral-cloning
    - Video - https://www.youtube.com/watch?v=QhQUy30ZPNk



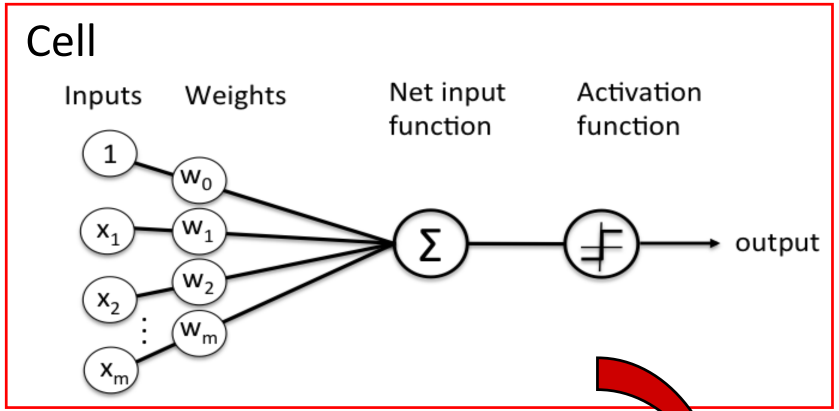
Layer (type)	Output Shape	Param #	Connected to
lambda_1 (Lambda)	(None, 160, 320, 3)	0	lambda_input_1[0][0]
cropping2d_1 (Cropping2D)	(None, 65, 320, 3)	0	lambda_1[0][0]
convolution2d_1 (Convolution2D)	(None, 63, 318, 32)	896	cropping2d_1[0][0]
maxpooling2d_1 (MaxPooling2D)	(None, 31, 159, 32)	0	convolution2d_1[0][0]
dropout_1 (Dropout)	(None, 31, 159, 32)	0	maxpooling2d_1[0][0]
convolution2d_2 (Convolution2D)	(None, 29, 157, 64)	18496	dropout_1[0][0]
maxpooling2d_2 (MaxPooling2D)	(None, 14, 78, 64)	0	convolution2d_2[0][0]
dropout_2 (Dropout)	(None, 14, 78, 64)	0	maxpooling2d_2[0][0]
convolution2d_3 (Convolution2D)	(None, 12, 76, 128)	73856	dropout_2[0][0]
maxpooling2d_3 (MaxPooling2D)	(None, 6, 38, 128)	0	convolution2d_3[0][0]
convolution2d_4 (Convolution2D)	(None, 4, 36, 256)	295168	maxpooling2d_3[0][0]
maxpooling2d_4 (MaxPooling2D)	(None, 2, 18, 256)	0	convolution2d_4[0][0]
flatten_1 (Flatten)	(None, 9216)	0	maxpooling2d_4[0][0]
dense_1 (Dense)	(None, 120)	1106040	flatten_1[0][0]
dense_2 (Dense)	(None, 20)	2420	dense_1[0][0]
dense_3 (Dense)	(None, 1)	21	dense_2[0][0]

Neural Network Architecture – RNN (1)

- Recurrent Neural Network – Cell



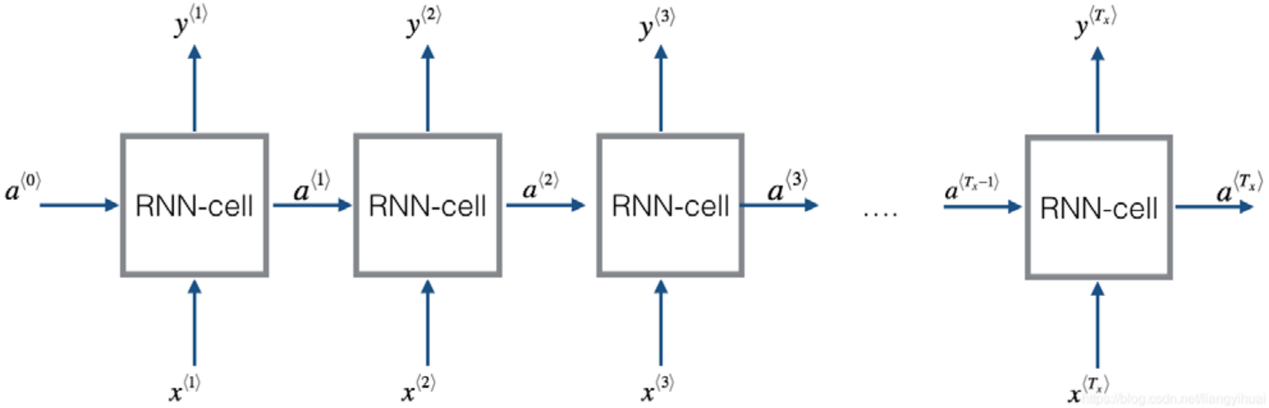
An unrolled recurrent neural network.

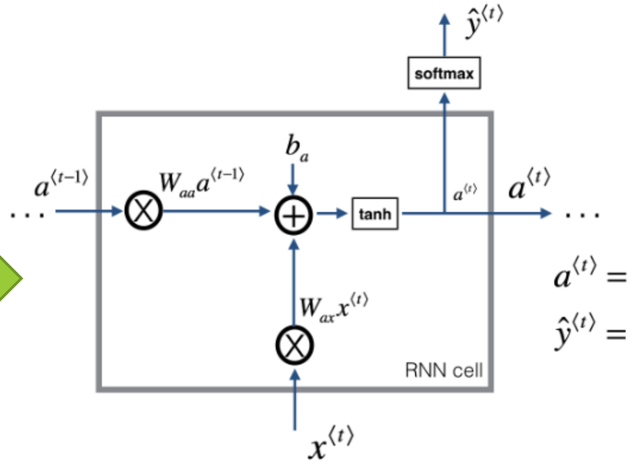


Source1

Source2

Transform



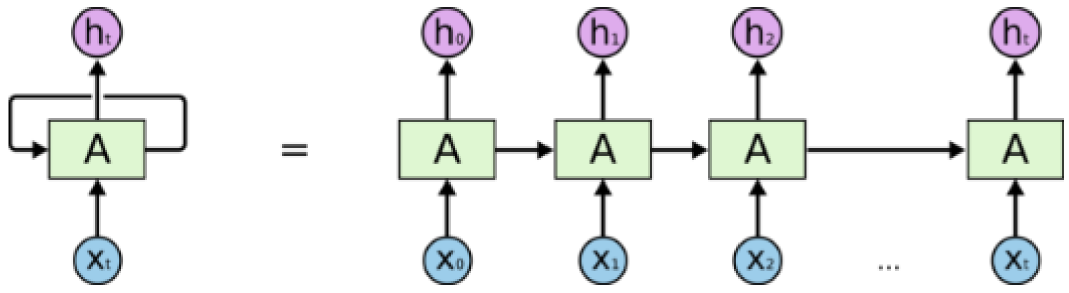


$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

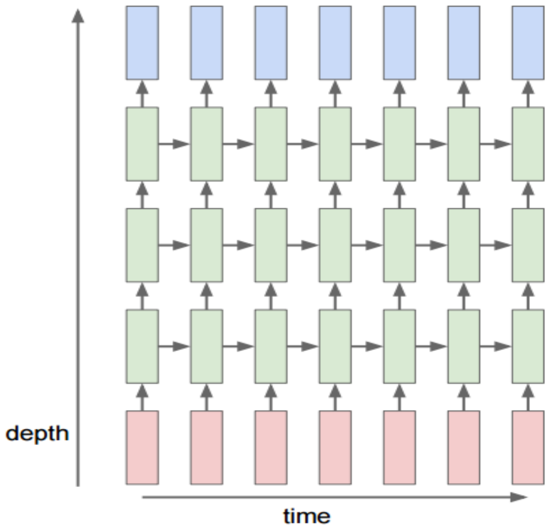
$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

Neural Network Architecture – RNN (2)

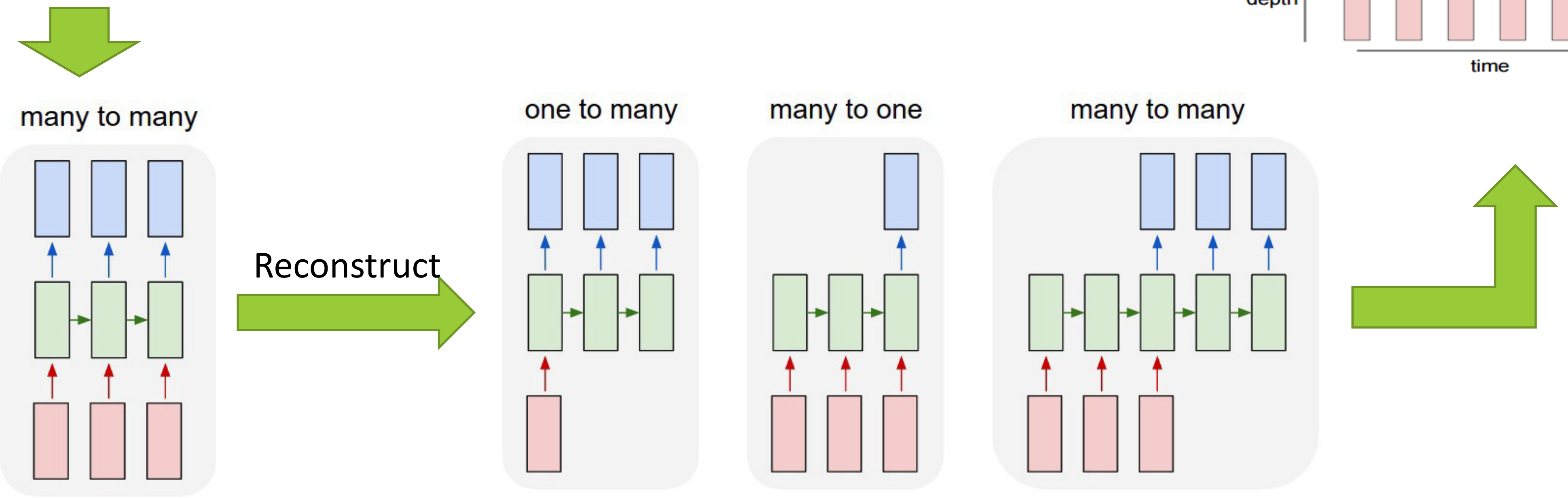
- Recurrent Neural Network – Architecture



An unrolled recurrent neural network.



Source1
Source2
Source3



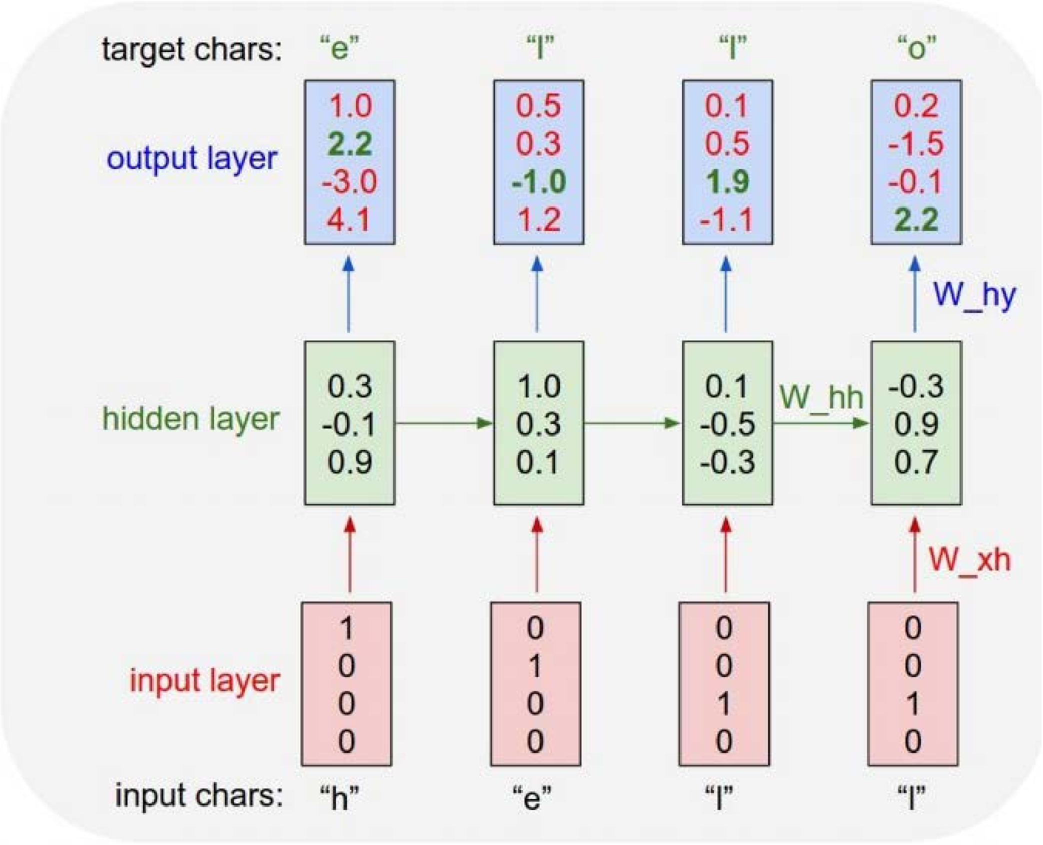
Neural Network Architecture – RNN (3)

- Recurrent Neural Network – Application
  - Process Sequences Data - character-level language model (Predict word)

Input: “h, e, l, l”

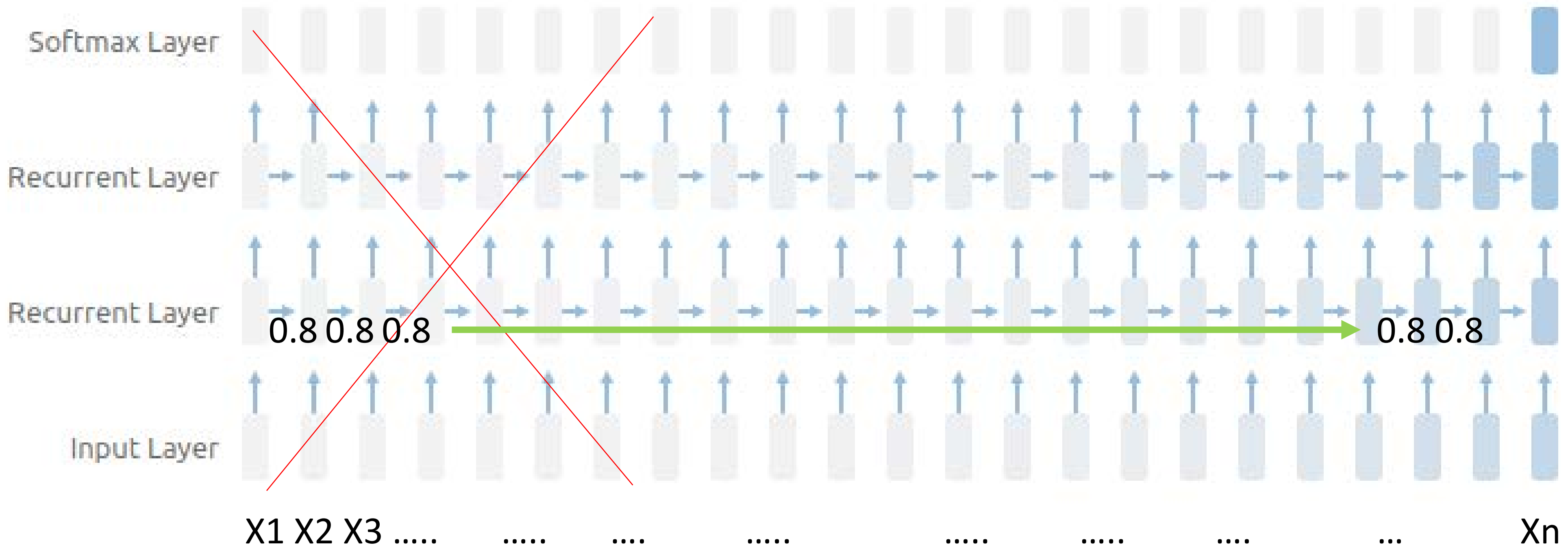
output: “e, l, l, o”

Predict Word: “hello”



Neural Network Architecture – RNN (4)

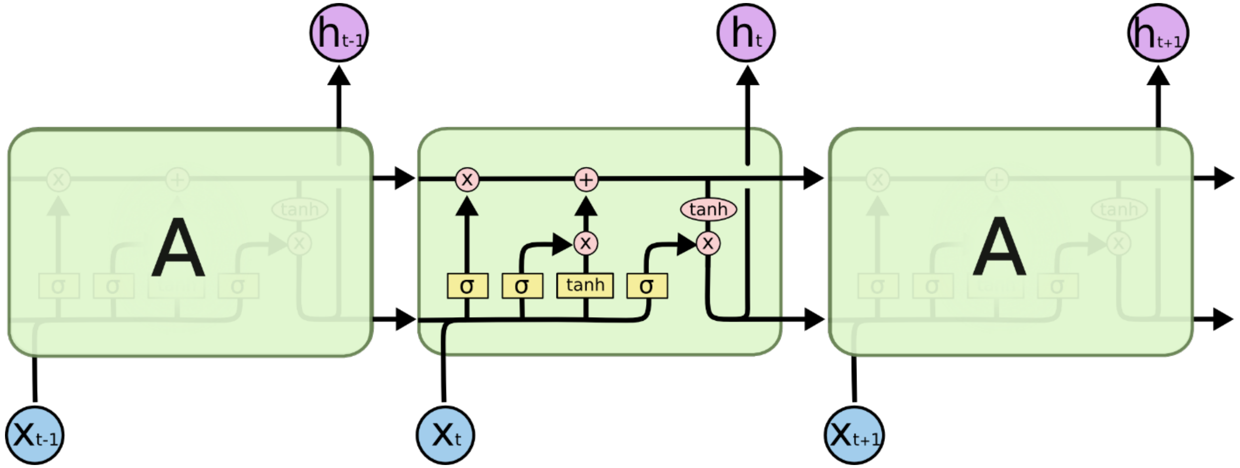
- Recurrent Neural Network – Vanishing gradient problem

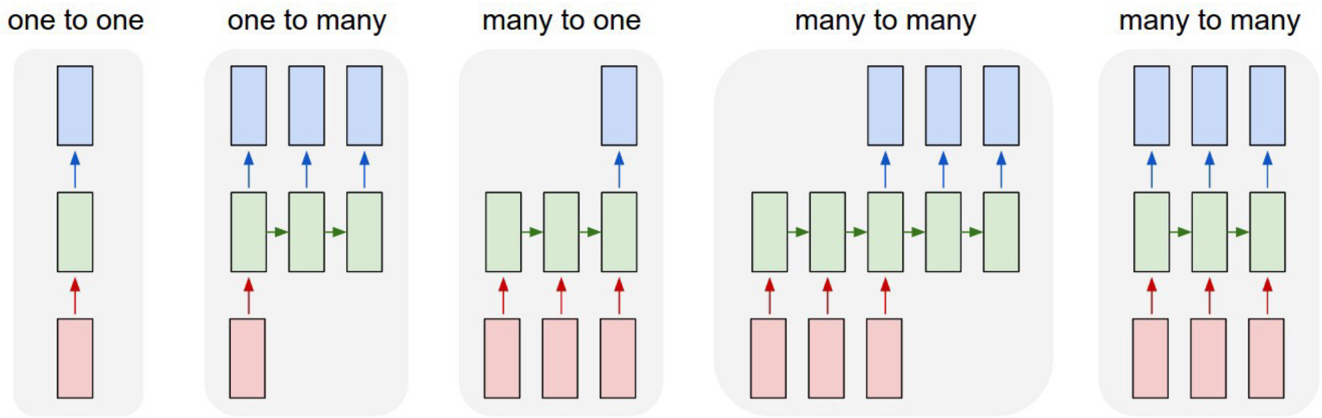


Neural Network Architecture – LSTM (1)

- Long Short Term Memory Networks – Architecture

Source1



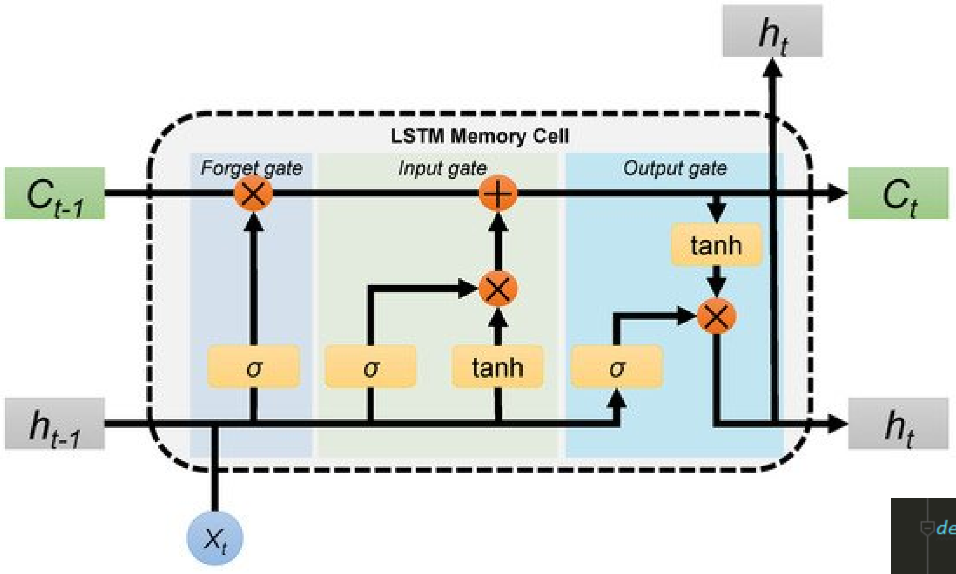


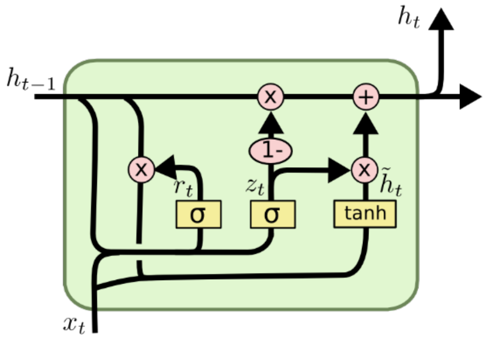
Neural Network Architecture – LSTM (2)

- Long Short Term Memory Networks – Cell

Source1

Source2





$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

```python
def LSTM_cell(input, output, state):
    input_gate = tf.sigmoid(tf.matmul(input, weights_input_gate) + tf.matmul(output, weights_input_hidden) + bias_input)

    forget_gate = tf.sigmoid(
        tf.matmul(input, weights_forget_gate) + tf.matmul(output, weights_forget_hidden) + bias_forget)

    output_gate = tf.sigmoid(
        tf.matmul(input, weights_output_gate) + tf.matmul(output, weights_output_hidden) + bias_output)

    memory_cell = tf.tanh(
        tf.matmul(input, weights_memory_cell) + tf.matmul(output, weights_memory_cell_hidden) + bias_memory_cell)

    state = state * forget_gate + input_gate * memory_cell

    output = output_gate * tf.tanh(state)
    return state, output
```

Neural Network Architecture – LSTM (3)

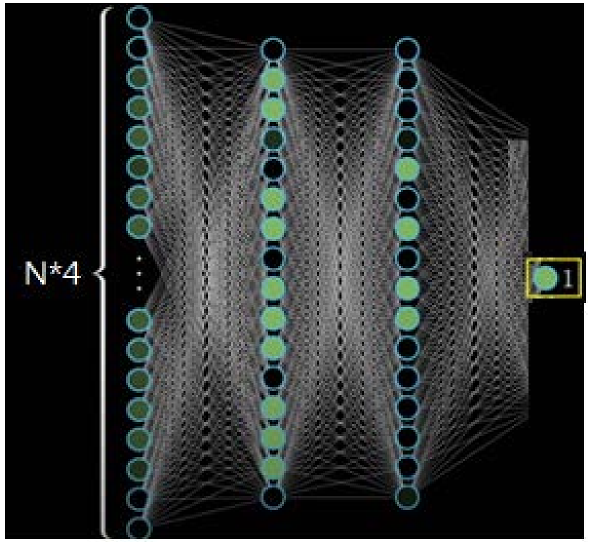
- Long Short Term Memory Networks - Application LSTM - Wiki

Applications [edit]

Applications of LSTM include:

- Robot control[46]
- Time series prediction[42]
- Speech recognition[47][48][49]
- Rhythm learning[35]
- Music composition[50]
- Grammar learning[51][34][52]
- Handwriting recognition[53][54]
- Human action recognition[55]
- Sign language translation[56]
- Protein homology detection[57]
- Predicting subcellular localization of proteins[58]
- Time series anomaly detection[59]
- Several prediction tasks in the area of business process management[60]
- Prediction in medical care pathways[61]
- Semantic parsing[62]
- Object co-segmentation[63][64]
- Airport passenger management[65]
- Short-term traffic forecast[66]

Demo Project – Stock Predictor(1)



- **Predict Target**
  - 下一個交易日收盤價漲跌比率
- **Data Set**
  - Input Data: N個交易日價格漲跌率 [開, 高, 低, 收]
  - Output Data: N+1 交易日收盤價漲跌比率 [收]
- **Model Structure - MLP**
  - Input layer: N*4
  - Hidden layer: 2 layer
  - Output Layer: 1
- **Training Param**
  - Loss function: MSE
  - Optimizer: AdamOptimizer (learning_rate=0.001)
  - BatchSize: 80
  - Epoch: Training loss less than '0.000001'

- **Training Dataset sample**
  2330-台積電股價

OpenPrice	HighPrice	LowPrice	ClosePrice	Next_ClosePrice
-0.29	-0.29	-2.46	-1.73	-1.62
-2.94	-1.03	-2.94	-1.62	-0.45
-0.15	0.3	-0.75	-0.45	0
0.3	0.75	0	0	-4.95
-2.1	-2.1	-4.95	-4.95	-0.47
-0.47	0	-1.42	-0.47	3.17
1.27	3.49	0.95	3.17	0.77
1.23	1.38	-0.15	0.77	1.53
0.61	1.53	0.46	1.53	-1.35
-0.75	-0.6	-1.95	-1.35	-0.15

Demo Project – Stock Predictor(2)

- Code Review
  - Path:
    - /StockPredictor/main.py

- Note:
  - Object Oriented Programming