H01

R語言深度學習入門

Introduction to Deep Learning with R

吳漢銘

國立政治大學 統計學系

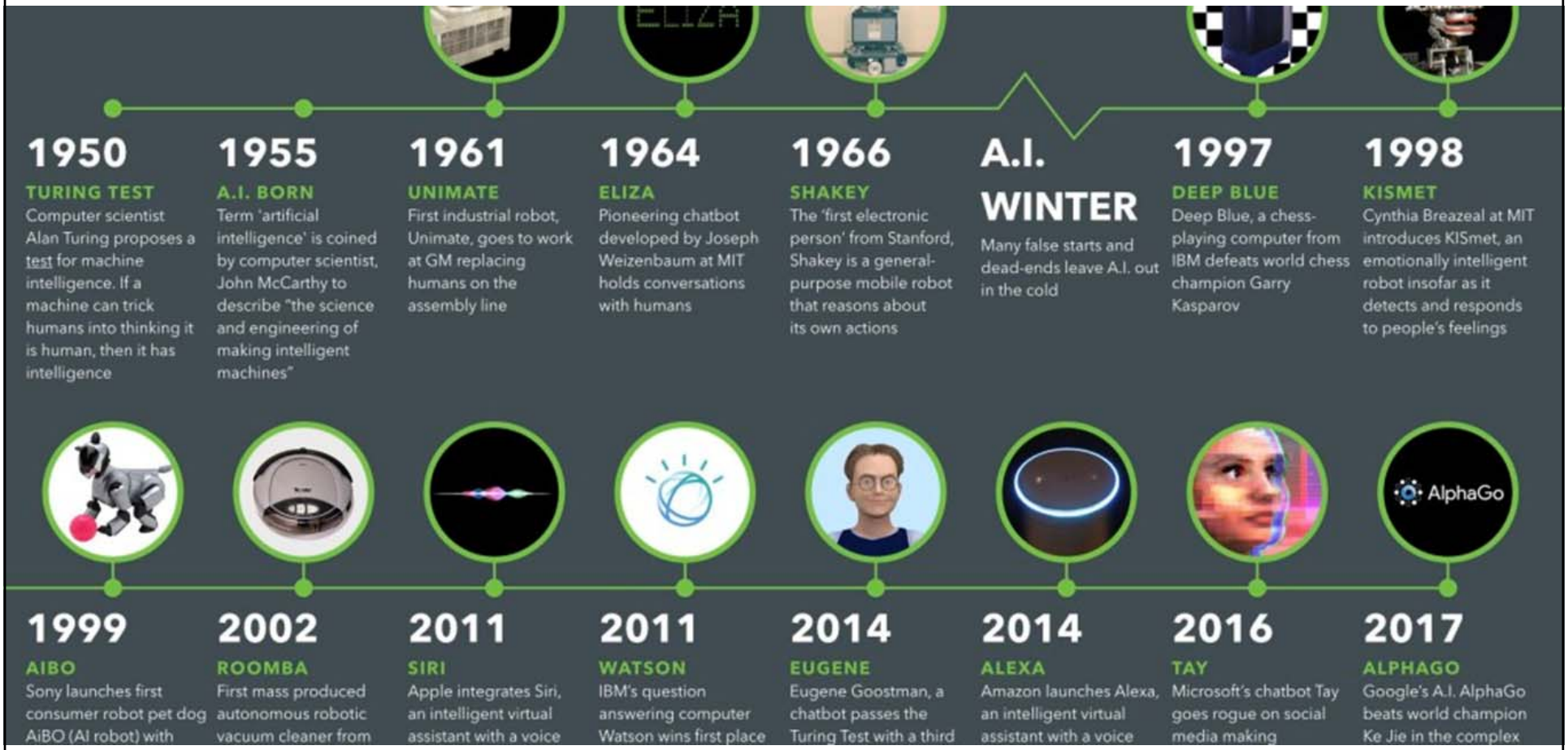
https://hmwu.idv.tw

本章大綱&學習目標

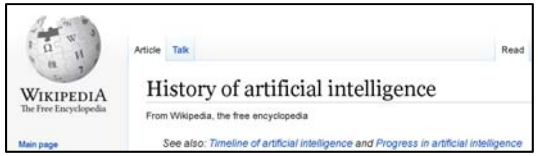
Source: https://wiki.komica.org/

- AI 簡介，學習路徑 (+統計人的觀點)。
- 機器學習與深度學習的數學統計基礎: regularization, gradient descent method, activation functions used in neural networks, loss functions of a model.
- ML中常用的Datasets.
- 實作Simple neural network architecture: forward propagation with backpropagation.
- Overview of the deep learning packages in R & 範例
- R to TensorFlow/Keras 安裝 (CPU/GPU版本)
- TensorFlow/Keras 範例

Artificial Intelligence Timeline Infographic – From Eliza to Tay and beyond (August 21, 2017)



https://digitalwellbeing.org/artificial-intelligence-timeline-infographic-from-eliza-to-tay-and-beyond/
https://medium.com/@amritenduroy/a-perspective-on-the-history-of-artificial-intelligence-ai-13fbb5a53ec7



https://en.wikipedia.org/wiki/History_of_artificial_intelligence

https://en.wikipedia.org/wiki/Timeline_of_artificial_intelligence

The next generation of AI

22,563 views | Oct 12, 2020, 09:22pm EDT

The Next Generation Of Artificial Intelligence

Rob Toews Contributor

AI

I write about the big picture of artificial intelligence.

AI legend Yann LeCun, one of the godfathers of deep learning, sees self-supervised learning as the key to AI's future. [-] © 2018 BLOOMBERG FINANCE LP

1. Unsupervised Learning
2. Federated Learning
3. Transformers

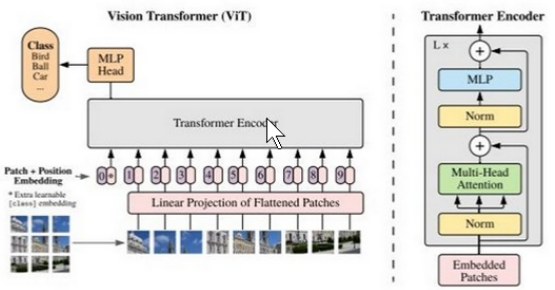
iThome

新聞

AI趨勢周報第146期：再見了卷積網路？一篇Transformer匿名論文引起ML社群圍觀

卷積網路CNN一直是影像辨識的首選，但近日一篇匿名論文（An Image Is Worth 16×16 Words）引起ML社群關注，指出直接用Transformer來處理影像Patch序列，其影像分類能力遠比CNN出色，連DeepMind、Tesla AI總監和發明AlexNet的OpenAI首席科學家都表示期待。該論文正接受AI頂級盛會ICLR 2021的評審，因此無法透露作者姓名。

文/ 王若樸 | 2020-10-15 發表

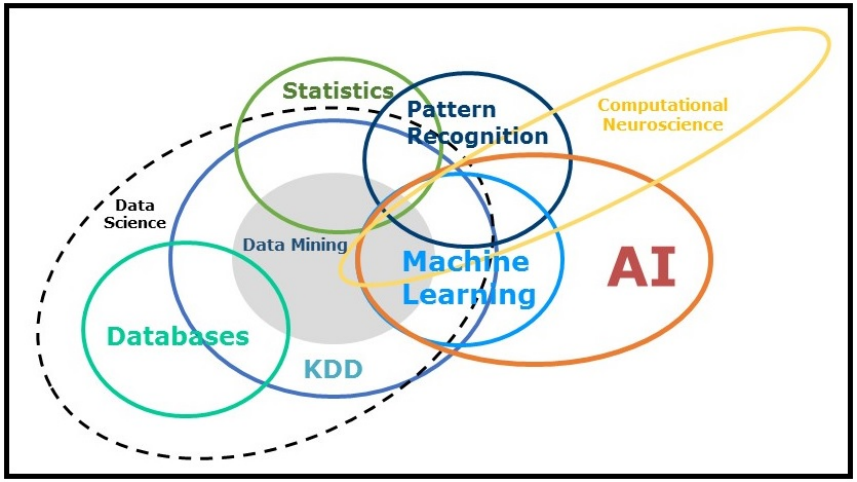


https://www.forbes.com/sites/robtoews/2020/10/12/the-next-generation-of-artificial-intelligence

楊立昆的個人網頁 (http://yann.lecun.com/)

Statistics, Data Mining, Machine Learning, Deep Learning, and AI

- **Statistics (STAT)**: Statistics is the discipline that concerns the collection, organization, analysis, interpretation and presentation of data.
- **Data Mining (DM)**: Data mining is a process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems
- **Machine learning (ML)**: Machine learning (ML) is the study of computer algorithms that improve automatically through experience. Machine learning algorithms build a mathematical model based on the training data, in order to make predictions or decisions.
- **Deep learning (DL)**: Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning.
- **Artificial intelligence (AL)**: the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.

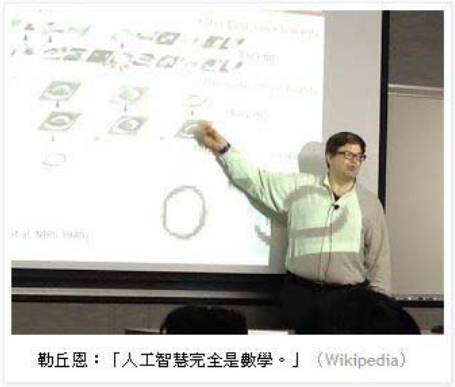


人工智慧是讓機器具有如同人類甚至更多的思辨能力。**機器學習**則是能夠達成人工智慧的方法，透過與人類相似的學習方法，訓練機器進行資料分類、處理與預測。**深度學習**代表實現機器學習的一種技術。

https://jamesmccaffrey.wordpress.com/2016/09/29/machine-learning-data-science-and-statistics/

統計 vs 機器學習

- 統計vs機器學習，數據領域的「少林和武當」！https://read01.com/O3dPexn.html
- 數據科學「內戰」：統計vs.機器學習 https://read01.com/ePRGMz7.html
- 機器學習VS統計模型 https://kknews.cc/zh-tw/tech/gz22r3y.html
- 統計學和機器學習到底有什麼區別？http://bangqu.com/iw4cp6.html
- 不要只關心怎麼優化模型，這不是機器學習的全部 http://bangqu.com/niYN6Z.html
- 運算思維：一張圖看懂機器翻譯（人工智慧）的原理 https://web.ntnu.edu.tw/~samtseng/present/CT_STM.html [運用電腦來做自動翻譯: 機率、貝氏定理]
- **臉書人工智慧研究主管、紐約大學教授** 楊立昆（**Yann LeCun**）：「**人工智慧完全是數學**。」


勒丘恩：「人工智慧完全是數學。」（Wikipedia）

- 人工智慧浪潮下的數學教育https://www.ettoday.net/news/20180508/1161306.htm
- 人工智慧大商機https://www.hbrtaiwan.com/article_content_AR0007381.html

每日頭條

首頁 | 健康 | 娛樂 | 時尚 | 遊戲 | 3C | 親子 | 文化 | 歷史 | 動

AI（人工智慧）就是統計學？

2018-10-26 *由 素思生涯規劃 發表于科技*

諾貝爾經濟學獎托馬斯·薩金特在《財經》世界科技創新論壇上的演講中說過一句話：人工智慧首先是一些很華麗的辭藻。人工智慧其實就是統計學，只不過用了一個很華麗的辭藻，其實就是統計學。

但是為什麼不說搞統計學呢？很簡單，因為不如人工智慧說法高大上，為什麼要高大上？因為高大上有人投錢。人工智慧代表了最新科技、最熱行業，能吸引投資，你要說是做統計學的誰投錢？誰買單？實際上這些工作很早就有人研究，只是那時候都歸類於統計學領域。

就像顯示系統縱橫位置指示器就是滑鼠；人體表皮污垢學就是搓灰；人體表皮死細胞分離器就是搓澡巾；智能高端數字通訊設備表面高分子化合物線性處理就是手機貼膜……

所以現在大家都學聰明了，那怕是老生常談也得包裝一個好聽的名稱，你說搞機器學習、深度學習就有人投錢，有人出大價錢挖你，你要說搞統計學大家立馬就不感興趣了，其實做的還是一回事。當然資本也是知道這些道理的，那為什麼還要投錢，因為資本是逐利的，投錢是為了掙更多錢，高大上的外衣就是掙錢的保障之一。

人工智慧和統計學不能完全劃等號

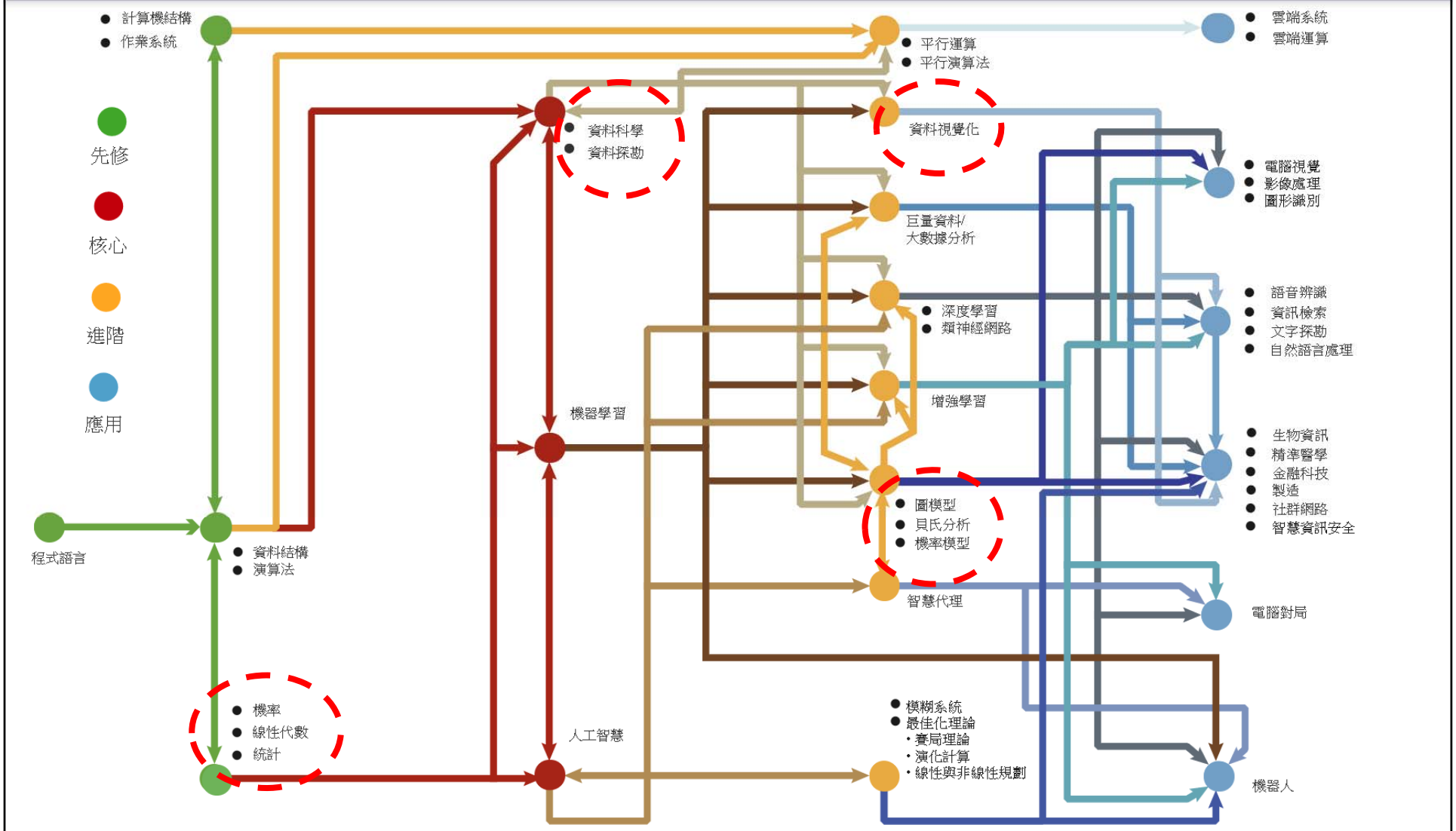
人工智慧和統計學存在莫大的關係，或者說統計學是人工智慧的最重要的理論基礎，但統計學和人工智慧依然有著很大不同，更不是一回事。

教育部人工智慧技術及應用人才培育計畫:
台灣人工智慧教育平台

https://idea.cs.nthu.edu.tw/~AIcoursemap/home/roadmap-course-series-chinese.html

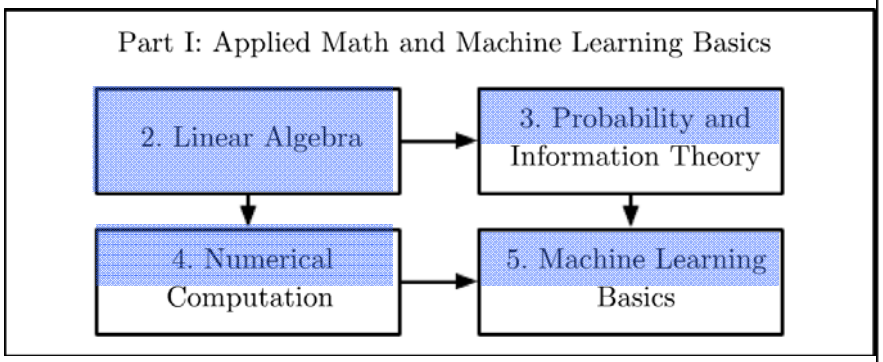
首頁　課程　課程地圖

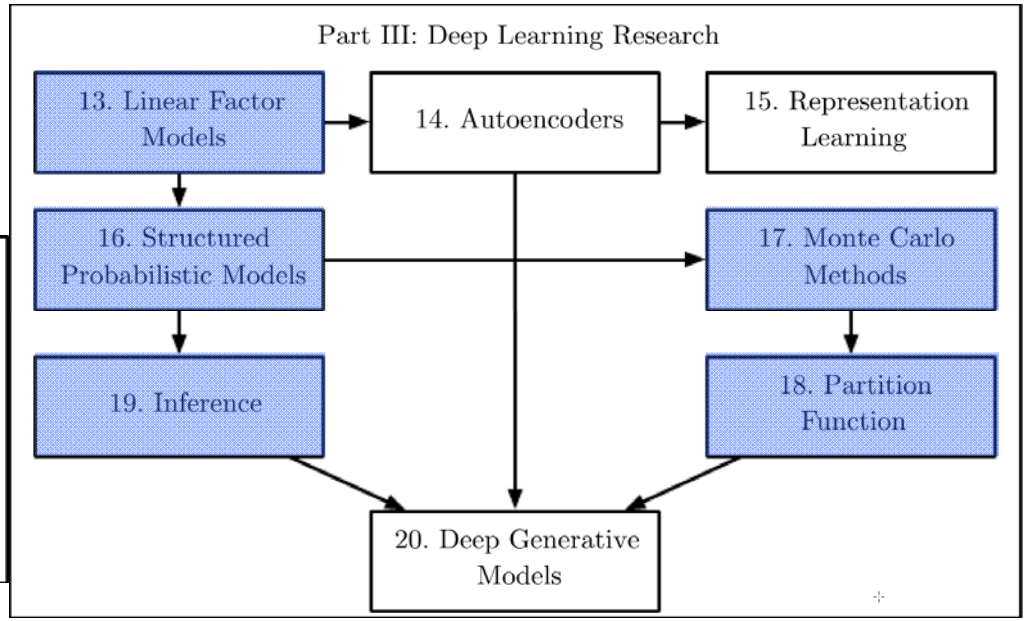
全部課程 | 人工智慧系統人才 | 電腦視覺人才 | 自然語言處理人才 | 應用領域人才 (巨量資料) | 應用領域人才 (深度學習) | 機器人人才

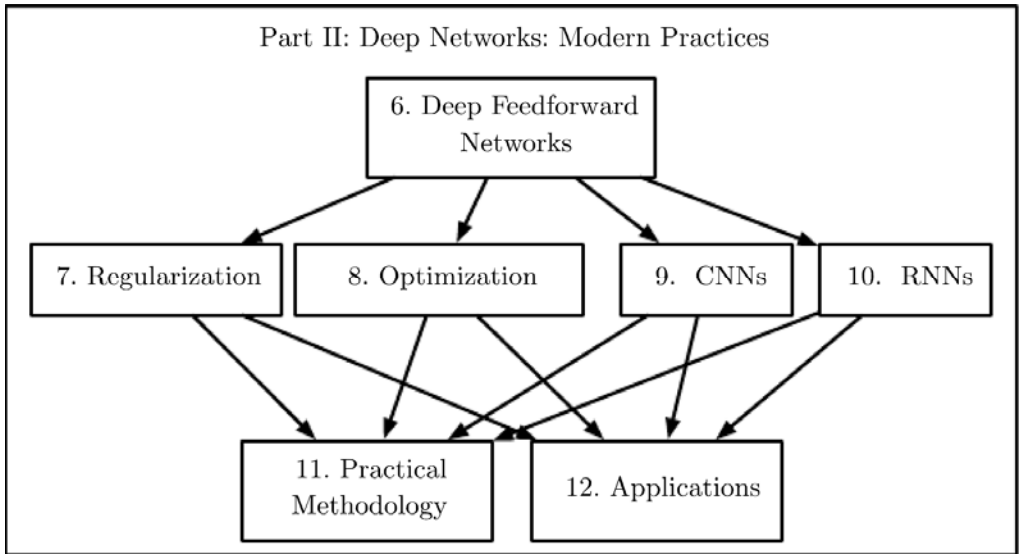


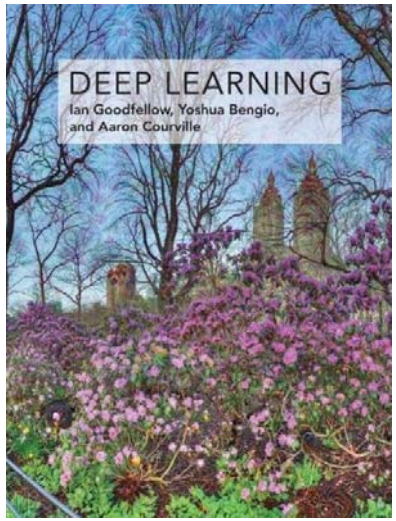
Learning Path

https://www.deeplearningbook.org/



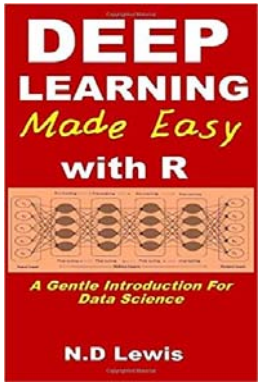


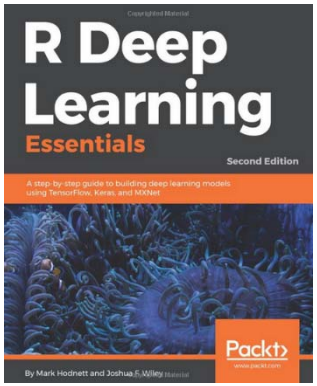


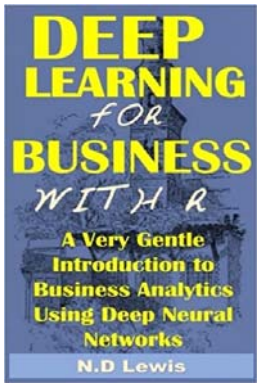


Deep Learning with R

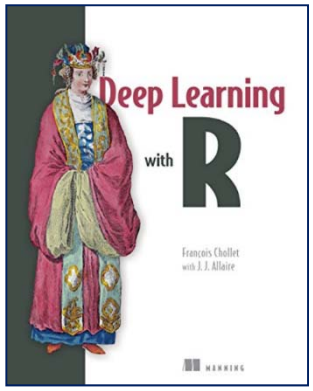
2016

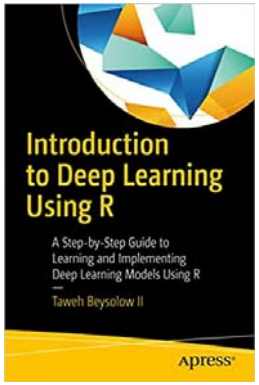


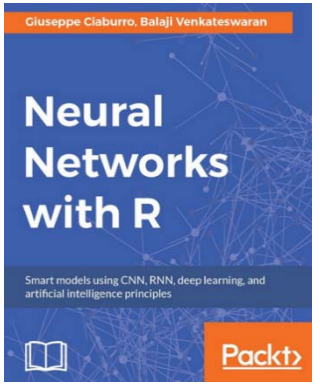


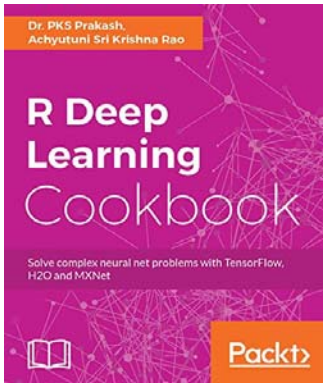


2017

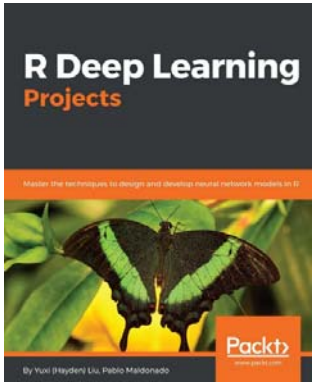


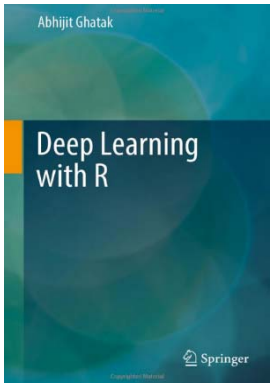




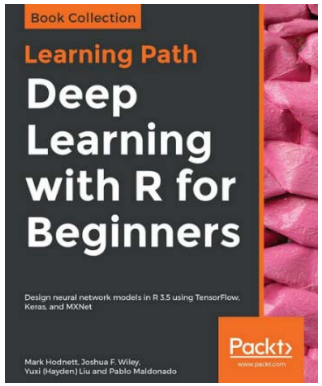


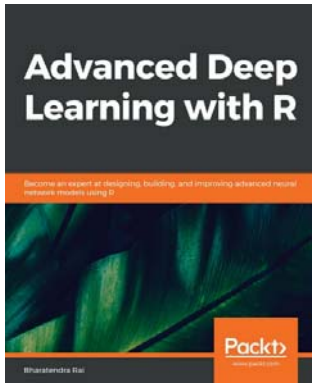
2018

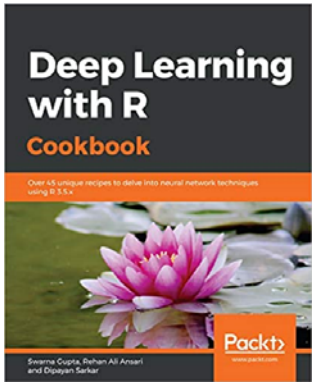




2019

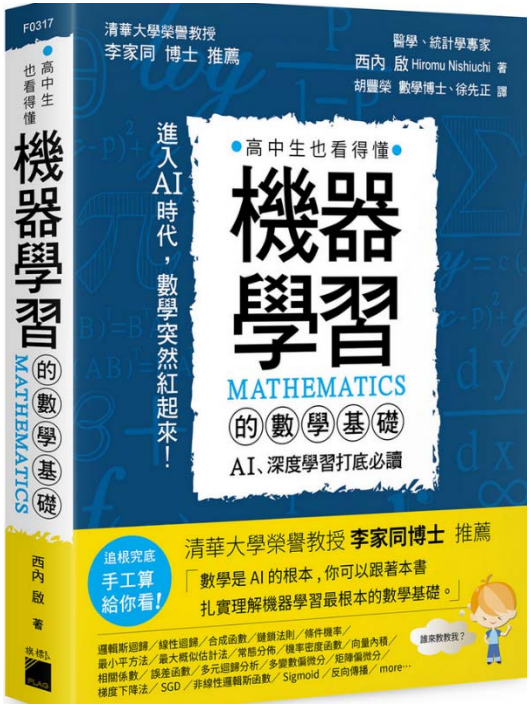






2020

機器學習、深度學習、AI中的數學與統計



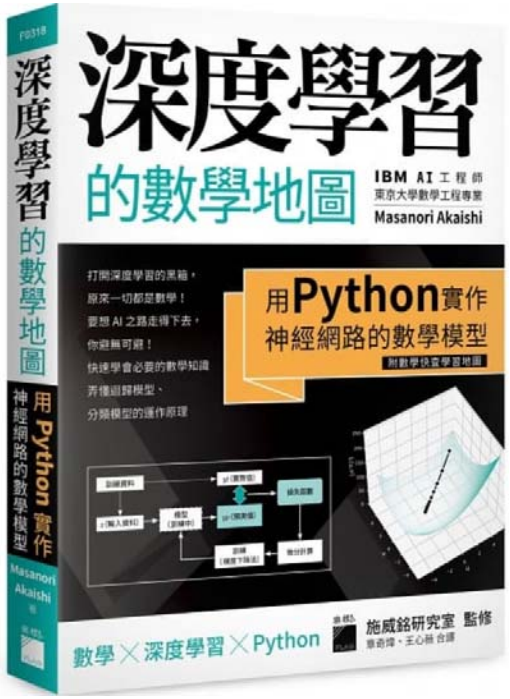
機器學習的數學基礎：
AI、深度學習打底必讀

醫學統計學專家 西內啟 著 ，胡豐榮博士, 徐先正 合譯，出版商: 旗標科技出版(2020-01-31)

[日]涌井良幸、涌井貞美
深度學習的數學
出版商: 人民郵電
出版日期: 2019-06-01

目錄：

機器學習、深度學習、AI中的數學與統計

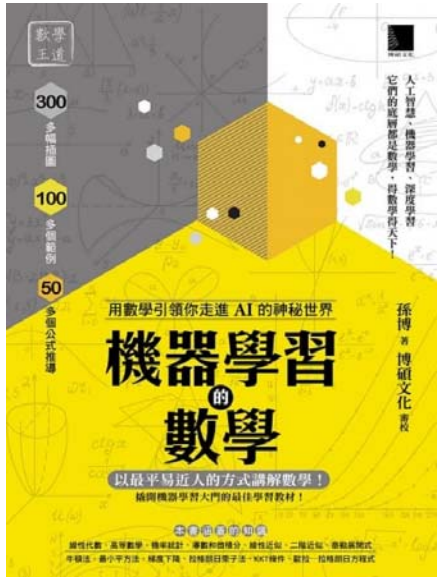


深度學習的數學地圖：用 Python 實作神經網路的數學模型
（附數學快查學習地圖）
最短コースでわかるディープラーニングの数学
作者： Masanori Akaishi
譯者： 章奇煒, 王心薇
出版社：旗標 出版日期：2020/05/28

第2章: 微分、積分
第3章: 向量、矩陣
第4章: 多變數函數的微分
第5章: 指數函數、對數函數
第6章: 機率、統計
第7章: 線性迴歸模型（迴歸）
第8章: 邏輯斯迴歸模型（二元分類）
第9章: 邏輯斯迴歸模型（多類別分類）
第10章: 深度學習
第11章: 以實用的深度學習為目標

深度學習的數學：用數學開啟深度學習的大門
作者： 涌井良幸, 涌井貞美
譯者： 楊瑞龍
出版社：博碩
出版日期：2020/05/04

機器學習的數學：用數學引領你走進AI的神秘世界
作者： 孫博
譯者： 博碩文化
出版社：博碩
出版日期：2020/09/09



損失函數 (Loss Functions)

- The **objective function** of an algorithm is to reduce the **loss** (**cost**):

$$L(w) = (Measure\ of\ model's\ fit) + (Measure\ of\ model's\ complexity)$$

or

$$C(\theta) = (Measure\ of\ model's\ fit) + (Measure\ of\ model's\ complexity)$$

- The measure of the model's fit is determined by
  - the MSE (Mean Squared Error) for regression,
  - CE (Classification Error) for classification.

Higher the model complexity, higher is the propensity for the model to capture the noise in the data by ignoring the signal.

- A measure of the model complexity is determined by
  - the sum of the absolute values of the structural parameters of the model (l_1 regularization),
  - the sum of the squared values of the structural parameters of the model (l_2 regularization).

- **Loss function** is usually a function defined for a observation (data point) and, it measures the penalty (difference between a single label and the predicted label).
- **Cost function** is more generally a sum of the loss functions over the training data set and the penalty for model complexity, i.e.,regularization.
- They are sometimes used interchangeably.

正則化 (Regularization)

- In deep learning, the l_2 regularization technique is used:

$$l_2 = w_0^2 + w_1^2 + \cdots + w_n^2 = \|w\|_2^2.$$

- **Objective**: select the model's structural parameters w_i, such that the loss function $L(w)$ is minimized , by using a weight decay regularization parameter λ on the l_2 -norm of the parameters such that, it penalizes the model for larger values of the parameters.

$$L(w) = Error\ metric + \lambda\|w\|_2^2.$$

  - λ = 0: there is no regularization;
  - λ values greater than zero force the weights to take a smaller value.
  - λ=∞: the weights become zero.

- **Regularization**: significantly reduces the variance of a deep learning model, without any substantial increase in its bias.

梯度下降法 (Gradient Descent Method)

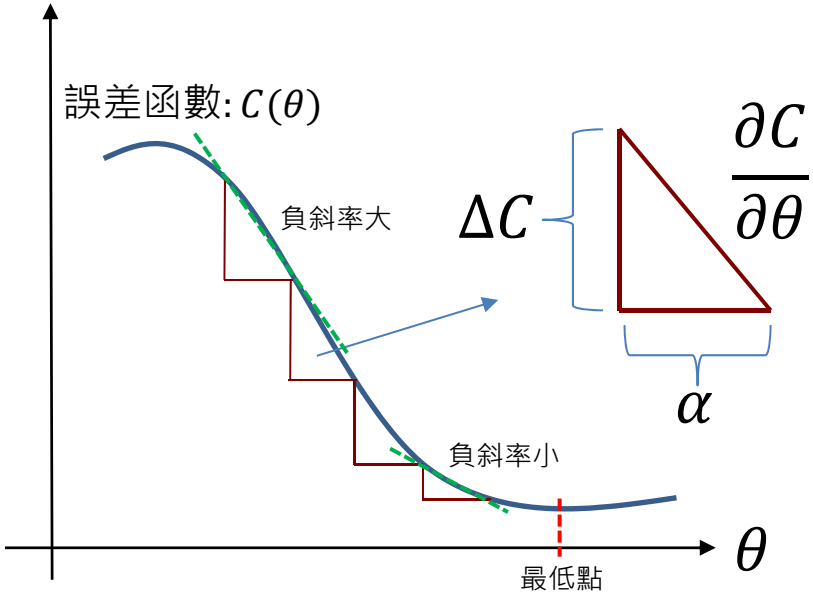
- 求「誤差平方和的最小值」，傳統算法是需解決大量的聯立方程式與計算高階反矩陣，時間成本高，且效率不彰。梯度下降法則不需要逐一代入所有數值去求解，可以提高計算的效率。

步驟:

1. 在合理範圍內取最初的參數值$\theta(t)$, $t = 0$。
2. 計算誤差(損失)函數 C的斜率 $\frac{\partial C}{\partial \theta}|_{\theta=\theta(t)}$ 。
3. 將原本參數值$\theta(t)$減去「步驟2的斜率乘以學習率α」，調整出新的參數值$\theta(t+1)$ 。
4. 將新參數值$\theta(t+1)$代回步驟2計算新的斜率。
5. 反覆迭代步驟2~4，直到新參數值與前一次參數值非常接近，就停止。

$$\theta(t+1) = \theta(t) - \alpha \times \frac{\partial C}{\partial \theta}|_{\theta=\theta(t)}$$

- 學習率α會影響機器學習的進度:
- α值大，可用較短的時間找到斜率最接近0的位置(收斂速度快)，但缺點是跳太快有跳過正確值的風險。
- α值小，收斂速度慢。



$$\frac{\Delta C}{\alpha} = \frac{\partial C}{\partial \theta}$$

$$\Rightarrow \Delta C = \alpha \frac{\partial C}{\partial \theta}$$

誤差函數的誤差值會下降ΔC的量

範例: 梯度下降法

假設主管有5位部屬，在交辦全新工作時，每個人依據自己過往經驗來評估新工作的完成時間。假設每個人評估作業時間的誤差是呈現平均數為0，標準差為sigma的常態分佈。

部屬	預估時間(日)
1	104
2	137
3	86
4	60
5	113

- 若梯度<0，則誤差函數$C(\theta)$的值會隨θ值增大而變小，表示往$C(\theta)$的低點靠近。
- 若梯度>0，則誤差函數$C(\theta)$的值會隨θ值增大而變大，表示遠離$C(\theta)$的低點。
- 目的是找到梯度變化趨近於0的θ值，此θ值能讓$C(\theta)$的值最小。

1. 初始參數值$\theta(0) = 10$。
2. 計算誤差函數C的斜率 $\frac{\partial C}{\partial \theta}\Big|_{\theta=\theta(t)}$ 。

$$C(\theta) = \sum_{i=1}^{n}(y_i - \theta)^2 = \sum_{i=1}^{n} y_i^2 - 2\theta\sum_{i=1}^{n} y_i + \sum_{i=1}^{n}\theta^2$$

$$\frac{\partial}{\partial\theta}C(10) = \lim_{\Delta\theta\to 0}\frac{C(10+\Delta\theta) - C(10)}{\Delta\theta}$$

$$\approx \frac{C(10.0001) - C(10)}{0.0001}$$

$$\approx \frac{43849.91 - 43850}{0.0001} = -900 \quad \text{(梯度)}$$

$$C(\theta) = 53350 - 2\theta \times 500 + 5\theta^2$$

範例: 梯度下降法

3. 將原本參數值$\theta(t)$減去「步驟2的斜率乘以學習率α」，調整出新的參數值$\theta(t+1)$。

$$\theta(t+1) = \theta(t) - \alpha \times \frac{\partial C}{\partial \theta}|_{\theta=\theta(t)}$$

$$10 - 0.05 \times (-900) = 55$$

4. 將新參數值$\theta(t+1)$代回步驟2計算新的斜率。

$$\frac{\partial}{\partial \theta} C(55) = \lim_{\Delta\theta \to 0} \frac{C(55+\Delta\theta) - C(55)}{\Delta\theta}$$

$$\approx \frac{C(55.0001) - C(55)}{0.0001} = -450,$$

$$55 - 0.05 \times (-450) = 77.5$$

- 處理多元或複雜的問題時，n很大時，梯度下降法運算量大無效率。
- 有比梯度下降法更精確有效的演算法被提出來，其核心都是誤差函數的斜率。
- The mini-batch method (最廣泛被採用的方法): the parameters are updated based on the current mini-batch and we continue iterating over all the mini-batches till we have seen the entire data set. The process of looking at all the mini-batches is referred to as an epoch.
- 隨機梯度下降法(Stochastic Gradient Descent, SGD): when the mini-batch size is set to one we perform an update on a single training example。可提升參數收斂的效率。

5. 反覆迭代步驟2~4，直到新參數值與前一次參數值非常接近，就停止。

$$10 \to 55 \to 77.5 \to 88.75 \to 94.375 \to 97.1875 \to \cdots \to 100$$

超參數調校 (Hyperparameter Tuning)

- The model training algorithm handles three categories of data:
  - **Input training data**: is used to configure our model to make accurate predictions from the unseen data.
  - **Model parameters**: the weights and biases (structural parameters) that our algorithm learns with the input data.
  - **Hyperparameters**: number of hidden layers, number of nodes per layer, learning rate, etc. These are configuration variables and are usually constant during a training process. (trial and error, and the optimal hyperparameters are obtained by leveraging multiple available datasets.)
- **Searching for Hyperparameters:** grid search, random search, greedy search, exact search algorithm (Breadth First Search (BFS) and Depth First Search (DFS), Beam Search, Bayesian optimization technique (Sequential Model-Based Optimization algorithms (SMBO))

最大概似估計量 (Maximum Likelihood Estimation，MLE)

- Maximum likelihood estimation (MLE) is a method used to determine parameter values of the model.
- The parameter values are found such that they maximize the likelihood that the process described by the model produces the data that were actually observed.
- We would, therefore, pick the parameter values which maximizes the likelihood of our data. This is known as the maximum likelihood estimate.

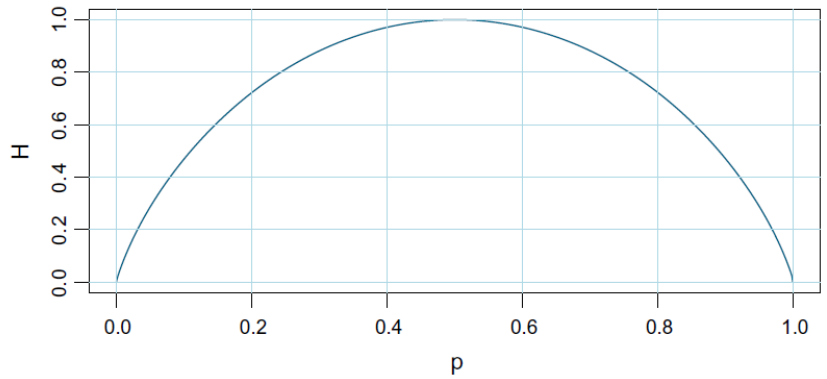
$$\hat{\theta}_{MLE} = \arg\max_{\theta} \prod_{i=1}^{n} P(y^{(i)}|x^{(i)};\theta)$$

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \sum_{i=1}^{n} \log\left(P(y^{(i)}|x^{(i)};\theta)\right)$$

熵 (Entropy)

- Entropy defines randomness.
- Entropy is like describing how unpredictable something is.
- If we consider **a set of possible events with known probabilities of occurrence** being $(y_1; y_2; \cdots, y_n)$
- H is maximum when all the pi are equal, i.e., 1/n. This is the most uncertain situation.
- Entropy is the **weighted average of the log probability of the possible events**, which measures the uncertainty present in their probability distribution.
- The higher the entropy, the less certain are we about the value.

$$H = H(y_1; y_2; \cdots, y_n) = -\sum_{i=1}^{n} y_i \log(y_i)$$



Plot of Entropy with probabilities p and $1 - p$.

注音 尸尤

交叉熵 (Cross-Entropy)

- Cross-entropy (CE) loss, or log-loss, measures the performance of a classification model whose output has a probability value between 0 and 1.
- The Cross-Entropy H(p, q) between two distributions p and q, quantifies the difference between the two probability distributions; i.e., how close is the predicted distribution to the true distribution.
- A perfect model, therefore, should have a CE loss of 0.

$$H(y,\hat{y}) = \sum_{i=1}^{n} y_i \log \frac{1}{\hat{y}_i} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

true probability distribution　　the computed probability distribution

In binary classification, the cross-entropy is proportional to the negative log- likelihood, and therefore minimizing the negative log-likelihood is equivalent to maximizing the likelihood.

In **binary classification**, where the number of output class labels are two, CE is calculated as

$$CE_{Loss} = H(y,\hat{y}) = -\left(y \log(\hat{y}) + (1-y) \log(1-\hat{y})\right)$$

In **multi-class classification**, where the number of output class labels are C-labels, CE is calculated as

$$CE_{Loss} = H(y,\hat{y}) = -\sum_{c=1}^{C} y_c \log(\hat{y}_c)$$

交叉熵損失函數 (The Cross-Entropy Loss)

- The cross-entropy loss: how close is the predicted probability distribution to the true distribution.

In Logistic Regression,
logistic function = sigmoid function

$$\phi(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Negative Log-Likelihood

$$\begin{aligned} -\log P(y|x;\theta) &= -\log\left(\prod_{i=1}^{n}(\phi^{(i)})^{y^{(i)}}(1-\phi^{(i)})^{1-y^{(i)}}\right) \\ &= -\sum_{i=1}^{n}\left(\log(\phi^{(i)})^{y^{(i)}}(1-\phi^{(i)})^{1-y^{(i)}}\right) \\ &= -\sum_{i=1}^{n}\left(y^{(i)}\log(\phi^{(i)}) + (1-y^{(i)})\log(1-\phi^{(i)})\right) \end{aligned}$$

Minimizing the **negative log-likelihood** of the data with respect to θ is the same as minimizing the **binary log loss** (binary cross-entropy) between the observed y values and the predicted probabilities thereof.

Machine Learning Datasets in R

mlbench: Machine Learning Benchmark Problems

- Version: 2.1-1; Published: 2012-07-10
- https://cran.r-project.org/web/packages/mlbench/index.html
- A collection of artificial and real-world machine learning benchmark problems.

```
> library(mlbench)
> data(BostonHousing)
> str(BostonHousing)
'data.frame':   506 obs. of  14 variables:
 $ crim    : num  0.00632 0.02731 0.02729 0.03237
 $ zn      : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5
 $ indus   : num  2.31 7.07 7.07 2.18 2.18 2.18 7.
 $ chas    : Factor w/ 2 levels "0","1": 1 1 1 1 1
 $ nox     : num  0.538 0.469 0.469 0.458 0.458 0.
 $ rm      : num  6.58 6.42 7.18 7 7.15 ...
 $ age     : num  65.2 78.9 61.1 45.8 54.2 58.7 66
 $ dis     : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad     : num  1 2 2 3 3 3 5 5 5 5 ...
 $ tax     : num  296 242 242 222 222 222 311 311
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15
 $ b       : num  397 397 393 395 397 ...
 $ lstat   : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv    : num  24 21.6 34.7 33.4 36.2 28.7 22.9
```

Regression:

- Los Angeles ozone pollution data (**Ozone**, 366x13)
- Boston Housing Data (**BostonHousing**, 506x14, y=medv)

Binary Classification:

- Sonar, Mines vs. Rocks (**Sonar**, 208x61)
- Johns Hopkins University Ionosphere database (**Ionosphere**, 351x35)
- House Votes 84, United States Congressional Voting Records 1984 (**HouseVotes84**, 435x17)
- Wisconsin Breast Cancer Database (**BreastCancer**, 699x11, Inputs: Integer (Nominal))
- Pima Indians Diabetes Database (**PimaIndiansDiabetes**, 768x9)

Multi-Class Classification:

- Zoo Data (**Zoo**, 101x17, #y=type(7))
- Glass Identification Database (**Glass**, 214x10, #y=Type(7))
- Soybean Database (**Soybean**, 683x26, #y=Class(19))
- Vehicle Silhouettes (**Vehicle**, 846x19, #y=Class(4))
- Vowel Recognition (**Vowel**, 990x10, #y=Class(11))
- DNA Primate (splice-junction gene sequences) (**DNA**, 3186x181, #y=Class(3))
- Landsat Multi-Spectral Scanner Image Data (**Satellite**, 6435x37, #y=classes(6))
- Letter Image Recognition Data (**LetterRecognition**, 20000x17, #y=lettr(26, A-Z))
- Shuttle Dataset (**Shuttle**, 58000x10, #y=Class(7))

Others: Servo Data (**Servo**, 167x5, y=continuous Class)

Boston Housing Data (BostonHousing, 506x14, y=medv)

```
> keras::dataset_boston_housing()  #Boston housing price regression dataset
> MASS::Boston #Housing Values in Suburbs of Boston
> mlbench::BostonHousing #Boston Housing Data
```

資料原調查目的: 估計波士頓居民為了提高空氣品質而願意支付額外費用的傾向

The original data are 506 observations on 14 variables, medv being the target variable:

- **crim**: per capita crime rate by town (人均犯罪率/鎮)
- **zn**: proportion of residential land zoned for lots over 25,000 sq.ft
- **indus**: proportion of non-retail business acres per town (非零售商業用地所佔的百分比/鎮)
- **chas**: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- **nox**: nitric oxides concentration (parts per 10 million) (氮氧化合物濃度)
- **rm**: average number of rooms per dwelling (平均房間數/寓所)
- **age**: proportion of owner-occupied units built prior to 1940 (1940年前建的自有住房所
- **dis**: weighted distances to five Boston employment centres (到達5個波士就業中心的平
- **rad**: index of accessibility to radial highways
- **tax**: full-value property-tax rate per USD 10,000 (全額不動產稅率)
- **ptratio**: pupil-teacher ratio by town (學生教師比/鎮)
- **b**: 1000(B - 0.63)^2 where B is the proportion of blacks by town
- **lstat**: percentage of lower status of the population (低社會地位人口百分比)
- **medv**: median value of owner-occupied homes in USD 1000's (自有住房數的中位數價格)

```
dataset_boston_housing(
  path = "boston_housing.npz",
  test_split = 0.2,
  seed = 113L
)
```

Value: Lists of training and test data: `train$x, train$y, test$x, test$y`.

The corrected data set (BostonHousing2) has the following additional columns:

- **cmedv**: corrected median value of owner-occupied homes in USD 1000's
- **town**: name of town
- **tract**: census tract (人口普查區)
- **lon**: longitude of census tract
- **lat**: latitude of census tract

```
> BHdata <- dataset_boston_housing()
> data(Boston, package="MASS")
> head(Boston)
```

Pima Indians Diabetes Database (PimaIndiansDiabetes2, 768x9)

```
> data("PimaIndiansDiabetes2", package="mlbench")
> ? mlbench::PimaIndiansDiabetes2
> str(PimaIndiansDiabetes2)
```

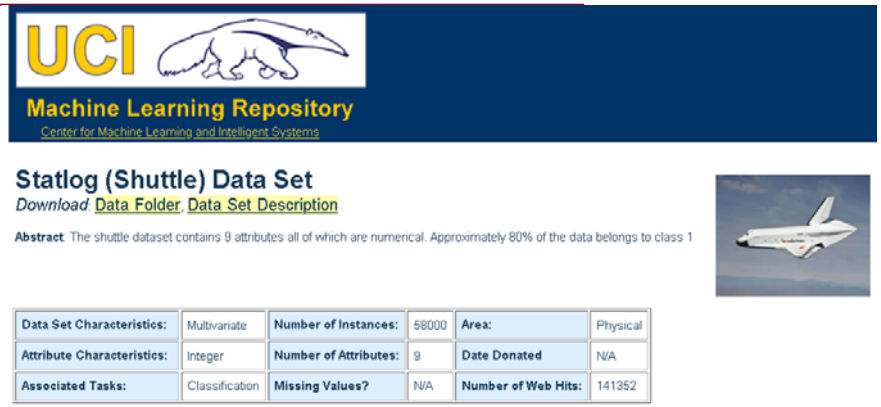
The data set PimaIndiansDiabetes2 (皮馬印地安人血統21歲以上女性) contains a corrected version of the original data set. While the UCI repository index claims that there are no missing values, closer inspection of the data shows several physical impossibilities, e.g., blood pressure or body mass index of 0. In PimaIndiansDiabetes2, all zero values of glucose, pressure, triceps, insulin and mass have been set to NA.

- **pregnant**: Number of times pregnant (懷孕次數)
- **glucose**: Plasma glucose concentration (glucose tolerance test) (血漿葡萄糖濃度)
- **pressure**: Diastolic blood pressure (mm Hg) (舒張壓(毫米汞柱))
- **triceps**: Triceps skin fold thickness (mm) (肱三頭肌皮膚褶厚度)
- **insulin**: 2-Hour serum insulin (mu U/ml) (血清胰島素)
- **mass**: Body mass index (weight in kg/(height in m)\^2)(身體質量指數)
- **pedigree**: Diabetes pedigree function (糖尿病家族函數)
- **age**: Age (years) (年齡)
- **diabetes**: Class variable (test for diabetes) (是否得糖尿病)

Shuttle Dataset (Shuttle, 58000x10, #y=Class(7))

- Space Shuttle Columbia disaster (February 1, 2003)
- Predicting NASA Space Shuttle Part Failure: Shuttle Radiator (散熱器) data trying to classify potential anomalies in the radiator position onboard Space Shuttles.
- **Task**: decide which type of control to use during the landing of the spacecraft depending on the external conditions.
- **Description**: The shuttle dataset contains 9 attributes all of which are numerical with the first one being time. The last column is the class with the following 7 levels: `Rad.Flow, Fpv.Close, Fpv.Open, High, Bypass, Bpv.Close, Bpv.Open`.
- Approximately 80% of the data belongs to class 1. Therefore the default accuracy is about 80%. The aim here is to obtain an accuracy of 99 - 99.9%.
- **Normal instances**: class 1: Rad Flow, class 4: High (94%)
- **Outliers**: rest five categories 6%

```
> # library(mlbench)
> data(Shuttle)
> head(Shuttle)
  V1 V2 V3 V4 V5  V6 V7 V8 V9      Class
1 50 21 77  0 28   0 27 48 22 Fpv.Close
2 55  0 92  0  0  26 36 92 56       High
3 53  0 82  0 52  -5 29 30  2   Rad.Flow
4 37  0 76  0 28  18 40 48  8   Rad.Flow
5 37  0 79  0 34 -26 43 46  2   Rad.Flow
6 85  0 88 -4  6   1  3 83 80     Bypass
> table(Shuttle$Class)
 Rad.Flow Fpv.Close  Fpv.Open      High    Bypass Bpv.Close  Bpv.Open
    45586        50       171      8903      3267        10        13
```



人工神經網路 (Artificial Neural Network，ANN)

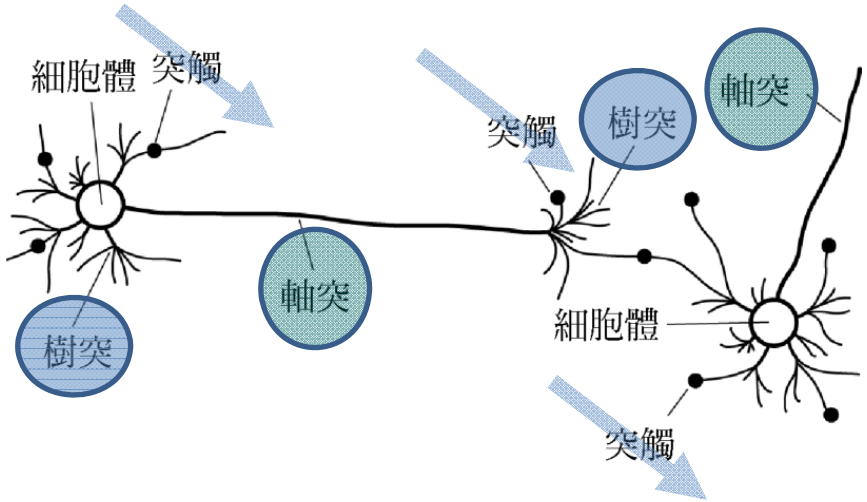
- 人工神經網路(Artificial Neural Network，ANN)，簡稱神經網路(Neural Network，NN)或類神經網路，在機器學習和認知科學領域，是一種**模仿生物神經網路**（動物的中樞神經系統，特別是大腦）的結構和功能的**數學模型**或**計算模型**，用於**對函式進行估計或近似**。(Wiki)

圖書館學與資訊科學大辭典

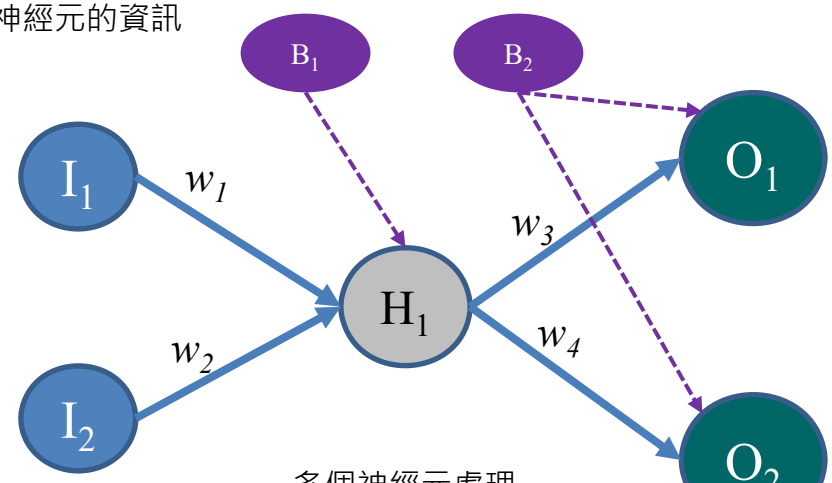
- ANN起源於1943年McCulloch和Pitts所提出之人工神經元運作模型，經過約20年的(初期)發展後，在1960年代中期因為遇到難以突破的技術瓶頸而沉寂了約20年，直到1986年由McClelland和Rumelhart提出Back-Propagation Algorithm突破技術瓶頸後，才又開始蓬勃發展至今。
- **ANN優點與特色**：
  - （一）自我學習能力：能自行從一堆輸入資料中學習到其中隱藏的規則或模式；
  - （二）容錯能力：其所學習到的規則或模式是散佈儲存於整個網路中，因此小部分的類神經網路損壞，並不會對系統的能力造成太大影響；
  - （三）平行處理能力：適合用來處理許多複雜的非線性問題；
  - （四）自我調整能力：能不斷根據新輸入的資料來調整已學習到的規則或模式。
- **ANN主要缺點**是學習速度緩慢，往往須要運用大量的輸入(或訓練)資料來讓類神經網路學習，而模糊邏輯(fuzzy logic)則有學習速度快但學習能力較為受限的特性，因此近年來的研究經常將人工類神經網路和模糊邏輯合併運用來互補長短。

模仿生物神經網路

生物神經網路



外部環境或
其它神經元的資訊



多個神經元處理

輸入層 **隱藏層** **輸出層**

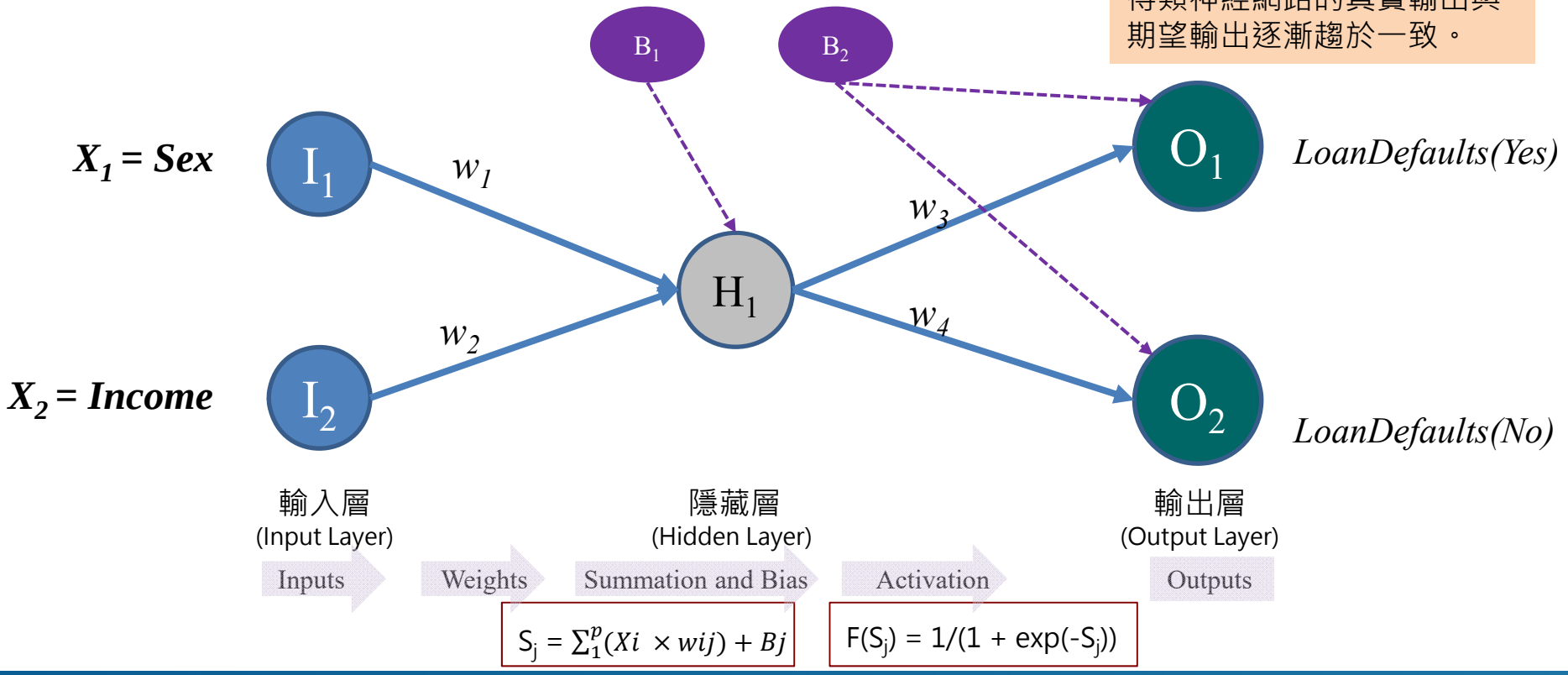
生物神經網路	人工神經網路
細胞體	神經元
樹突	輸入
軸突	輸出
突觸	權重

- W = 模仿生物神經細胞的突觸，稱為鍵結值(weights)，可視為一種加權效果，其值越大，對類神經網路的影響也更大。
- B = 模仿生物神經細胞的細胞核偏權值(bias)，具有偏移的效果。
- f(θ) = 模仿生物神經細胞的細胞核的非線性**轉換函數**，目的是將加權成績和的值做映射得到所需要的輸出。
- O = 輸出傳到外界環境或是其他人工神經元的資料。

人工神經網路主要結構 (Artificial Neural Network, ANN)

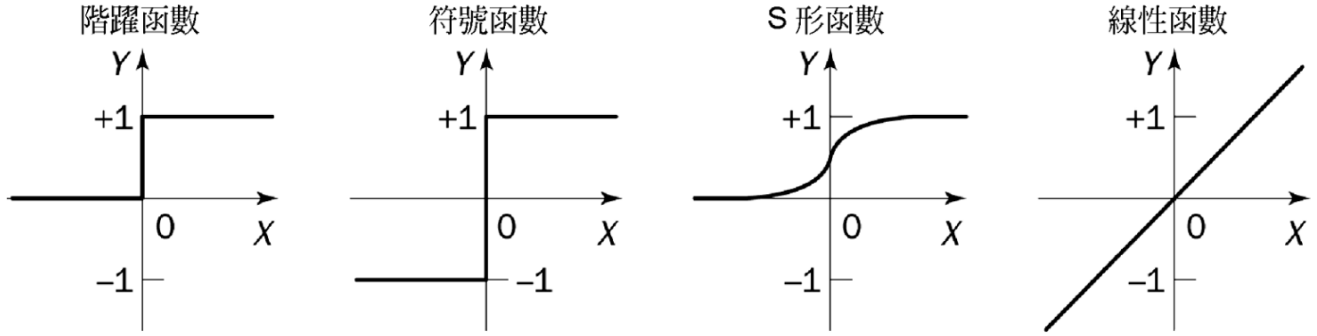
- 輸入層(Input Layer): 輸入的X變數因子。
- 輸出層(Output Layer): 結果值Y的組合。
- 中間的隱藏層(Hidden Layer): 負責運算的神經元，通常會以輸入層加上輸出層數目開根號決定單一層隱藏層中的神經元數目。(The hidden layer carries out a geometric transformation function of the data which goes through that layer.)
- Weights (權重): 代表各個神經元受到刺激後的活躍程度。
- Bias (偏權值)。

機器學習就是逐漸調整weights 和bias的過程，使得類神經網路的真實輸出與期望輸出逐漸趨於一致。



轉換函數: 神經元如何決定輸出

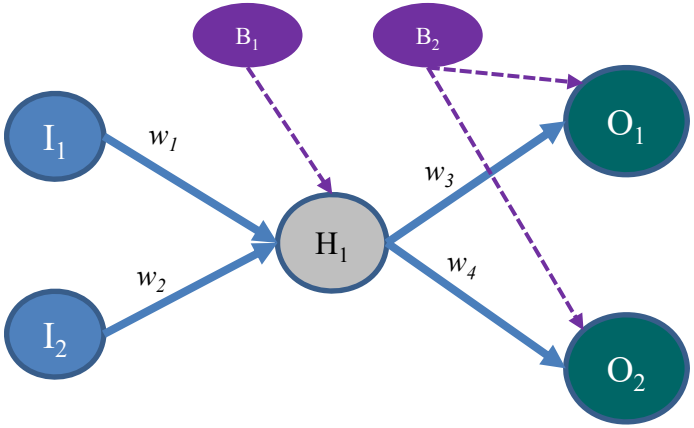
- 神經元一般透過**轉換函數**將最終的計算結果轉換為輸出，我們通常會依照處理問題的特性來選擇不同的轉移函數。
- 常見的轉移函數包含：步階函數、符號函數、線性函數與S型函數。



轉換函數 (transform function)

- 集成函數 (summation function): $S_j = \sum_{i=1}^{p}(X_i \times w_{ij}) + B_j$
- 啟動函數 (activation function): $F(S_j) = \frac{1}{1+\exp(-S_j)}$

X_i: 第i個神經元所輸出的資訊
S_j: 第j個神經元所接收資訊的總和
w_{ij}: 第i個神經元與第j個神經元之間的權重
B_j: 第j個神經元的Bias(偏權值)



啟動函數 (Activation Functions)

```
# Sigmoid
AF_sigmoid <- function(x){
  1/(1+exp(-x))
}
```

$$\phi(z) = \frac{1}{1+\exp(-z)}$$

```
# Softmax
AF_Softmax <- function(x){
  exp(x)/sum(exp(x))
}
```

$$\phi(z_i) = p_i = \frac{\exp z_i}{\sum_{k=1}^{q} \exp z_k}$$

```
# Hyperbolic Tangent
AF_tanh <- function(x){
  (exp(x)-exp(-x))/(exp(x)+exp(-x))
}
```

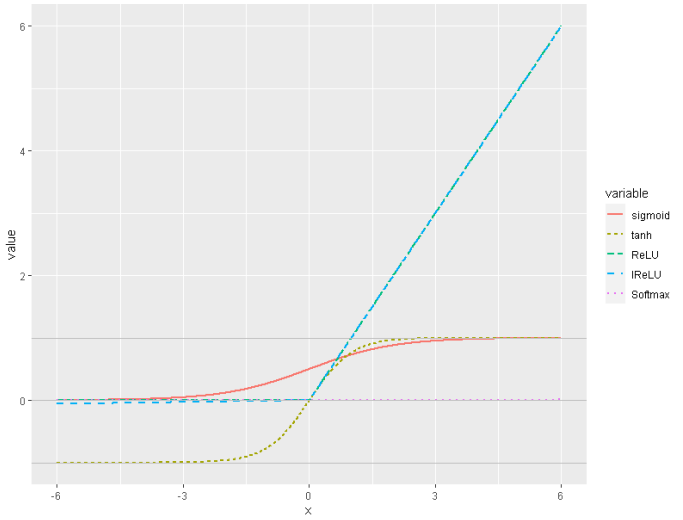
$$\phi(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\phi(z) = \begin{cases} z, & \text{if } z > 0, \\ az, & \text{otherwise} \end{cases}$$

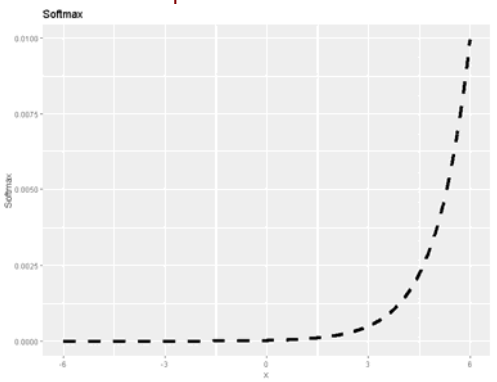
```
# Rectified Linear Unit
AF_ReLU <- function(x){
  ifelse(x > 0, x, 0)
}
```

$$\phi(z) = \max(z, 0)$$

```
# Leaky Rectified Linear Unit
AF_lReLU <- function(x, a=0.01){
  ifelse(x > 0, x, a*x)
}
```




```
library(reshape2)
x <- seq(-6, 6, 0.01)
xyData <- data.frame(x=x, sigmoid=AF_sigmoid(x), tanh=AF_tanh(x),
                     ReLU=AF_ReLU(x), lReLU=AF_lReLU(x), Softmax=AF_Softmax(x))
head(xyData)
xyData.lf <- melt(xyData, id.var="x")
head(xyData.lf)
ggplot(xyData.lf, aes(x=x, y=value, group=variable, colour=variable)) +
  geom_line(aes(linetype=variable), size=1) +
  geom_hline(yintercept=c(-1, 0, 1), color="gray")

ggplot(xyData, aes(x=x, y=Softmax)) +
  geom_line(size=2, linetype="dashed") +
  ggtitle("Softmax")
```

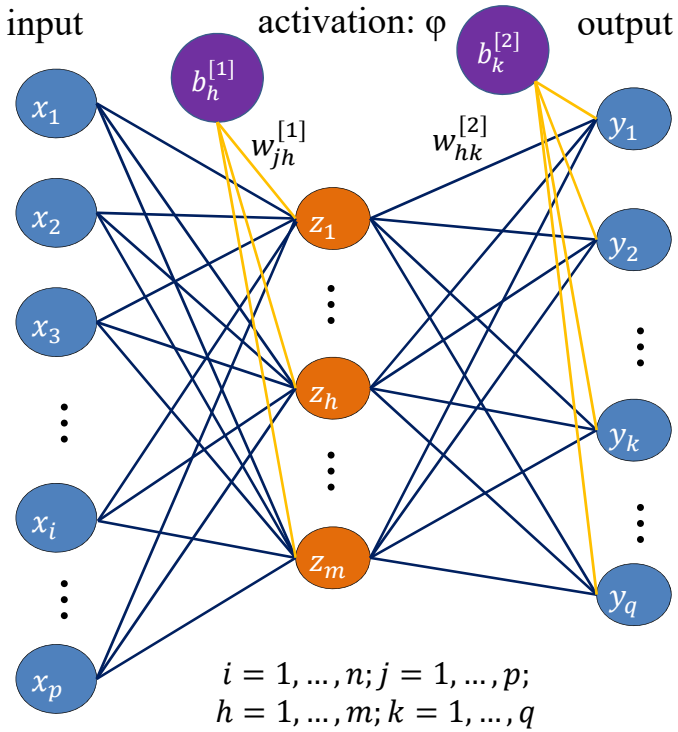


Data Recoding

```
> Loan <- read.table("Loan_training.txt", header=T, sep="\t")
> Loan
   Sex Income LoanDefaults
1    F  23000          Yes
2    F  24000          Yes
3    F  25000           No
4    F  26000           No
...
8    M  27000          Yes
9    M  30000           No
10   M  31000           No
> Loan$LoanDefaults <- as.factor(Loan$LoanDefaults)
> income <- (Loan$Income - min(Loan$Income))/(max(Loan$Income)- min(Loan$Income))
> no.class <- length(unique(Loan$LoanDefaults))
> Target <- data.frame(matrix(0, nrow=length(income), ncol=no.class))
> sapply(1:no.class, function(x){
+     Target[which(Loan$LoanDefaults == levels(Loan$LoanDefaults)[x]), x] <<- 1
+ })
[1] 1 1
> colnames(Target) <- c("NO", "YES")
> data.frame(Loan$Sex, income, Target)
   Loan.Sex income NO YES
1         F  0.000  0   1
2         F  0.125  0   1
3         F  0.250  1   0
4         F  0.375  1   0
...
8         M  0.500  0   1
9         M  0.875  1   0
10        M  1.000  1   0
```

RSNNS

單隱藏層前饋神經網路
Single-hidden-layer Feedforward Neural Networks (FFNNs)



$i = 1, \dots, n; j = 1, \dots, p;$
$h = 1, \dots, m; k = 1, \dots, q$

$$\boldsymbol{w}_h^{[1]} = \begin{bmatrix} w_{1h}^{[1]} \\ w_{2h}^{[1]} \\ \vdots \\ w_{ph}^{[1]} \end{bmatrix}_{p\times 1} \quad \boldsymbol{b}^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_m^{[1]} \end{bmatrix}_{m\times 1} \quad \boldsymbol{w}_k^{[2]} = \begin{bmatrix} w_{1k}^{[2]} \\ w_{2k}^{[2]} \\ \vdots \\ w_{mk}^{[2]} \end{bmatrix}_{m\times 1} \quad \boldsymbol{b}^{[2]} = \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \\ \vdots \\ b_q^{[2]} \end{bmatrix}_{q\times 1}$$

$h = 1, \cdots, m$ $\qquad$ $k = 1, \cdots, q$

$$z_h = \phi^{(1)}\left(\sum_{j=1}^{p} x_j \cdot w_{jh}^{[1]} + b_h^{[1]}\right), \qquad y_k = \phi^{(2)}\left(\sum_{h=1}^{m} z_h \cdot w_{hk}^{[2]} + b_k^{[2]}\right),$$

$$\boldsymbol{z} = \phi^{(1)}(\boldsymbol{W}^{[1]T}\boldsymbol{x} + \boldsymbol{b}^{[1]}) \qquad \boldsymbol{y} = \phi^{(2)}(\boldsymbol{W}^{[2]T}\boldsymbol{z} + \boldsymbol{b}^{[2]})$$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}_{p\times 1} \quad \boldsymbol{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}_{m\times 1} \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_q \end{bmatrix}_{q\times 1}$$

$$\boldsymbol{y} \equiv T(\boldsymbol{x};\boldsymbol{\theta}) = \phi^{(2)}\left(\boldsymbol{W}^{[2]T} \cdot \phi^{(1)}(\boldsymbol{W}^{[1]T}\boldsymbol{x} + \boldsymbol{b}^{[1]}) + \boldsymbol{b}^{[2]}\right)$$

$$\boldsymbol{\theta} = (\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}; \boldsymbol{W}^{[2]}, \boldsymbol{b}^{[2]})$$

$$\boldsymbol{W}^{[1]} = (\boldsymbol{w}_1^{[1]}, \boldsymbol{w}_2^{[1]}, \cdots, \boldsymbol{w}_m^{[1]})_{p\times m}$$

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \left\|\boldsymbol{y}_i - T(\boldsymbol{x}_i;\boldsymbol{\theta})\right\|_F^2 + \lambda \sum_{l=1}^{2} \left\|\boldsymbol{W}^{[l]}\right\|_F^2$$

$$\boldsymbol{W}^{[2]} = (\boldsymbol{w}_1^{[2]}, \boldsymbol{w}_2^{[2]}, \cdots, \boldsymbol{w}_q^{[2]})_{m\times q}$$

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - \alpha \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}(t)} \qquad \lim_{t\to\infty} \boldsymbol{\theta}(t) = \boldsymbol{\theta}^*$$

啟動函數的導數 (Derivatives of Activation Functions)

Derivative of Sigmoid

$$\phi(z) = \frac{1}{1+\exp(-z)}$$

$$\frac{d\phi(z)}{dz} = \phi(z)(1-\phi(z))$$

Derivative of tanh

$$\phi(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\frac{d\phi(z)}{dz} = 1 - tanh^2(z)$$

Derivative of Rectified Linear Unit

$$\phi(z) = \max(z, 0)$$

$$\frac{d\phi(z)}{dz} = I(z > 0)$$

Derivative of Softmax

$$\phi(z_i) = p_i = \frac{\exp z_i}{\sum_{k=1}^{q} \exp z_k}$$

$$\frac{d\phi(z_i)}{dz_j} = \begin{cases} p_i(1-p_j), & \text{if } i = j, \\ -p_i p_j, & \text{if } i \neq j. \end{cases}$$

Derivative of Leaky Rectified Linear Unit

$$\phi(z) = \begin{cases} z, & \text{if } z > 0, \\ az, & \text{otherwise} \end{cases}$$

$$\frac{d\phi(z)}{dz} = \begin{cases} 1, & \text{if } z > 0, \\ a, & \text{otherwise.} \end{cases}$$

Cross-Entropy Loss

- Cross-Entropy (CE) measures the divergence between two probability distributions.
  - CE is large: the difference between the two distributions is large.
  - CE is small: the two distributions are similar to each other.

Suppose there is a neural network model, which predicts K classes $1, 2, \cdots, K$ having probability of occurrences as $\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_K$ and having n_k instances for each of the K classes, i.e., $n_1 + n_2 + \cdots + n_K = n$, where n is the total number of observations.

But, for binary classification, sigmoid activation is the same as softmax.

- The likelihood of this data:

$$P(data|model) = \hat{y}_1^{n_1} \hat{y}_2^{n_2} \cdots \hat{y}_K^{n_K}$$

- The negative log-likelihood:

$$-\log P(data|model) = -\sum_k n_k \log(\hat{y}_k)$$

- The generalized CE loss function ($y_i = n_k/n$):

$$-\frac{1}{n} \log P(data|model) = -\frac{1}{n} \sum_k n_k \log(\hat{y}_k) = -\sum_k y_k \log(\hat{y}_k)$$

- The CE loss for sigmoid (2 classes):

$$J_{sigmoid}(w) = -\frac{1}{n} \sum_{i=1}^{n} (y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

- The CE loss for softmax (K classes):

$$J_{softmax} = -\sum_{k=1}^{K} \sum_{j=1}^{n_k} y_k^{(j)} \log(\hat{y}_k^{(j)})$$

$log(\hat{y})$ when $\hat{y} = 0$ → $log(max(\hat{y}, 1e\text{-}15))$.

Derivative of the Cost Function

Derivative of Cross-Entropy Loss with Sigmoid

$$J_{sigmoid}(w) = -\frac{1}{n}(Y \log(A) + (1 - Y) \log(1 - A))$$

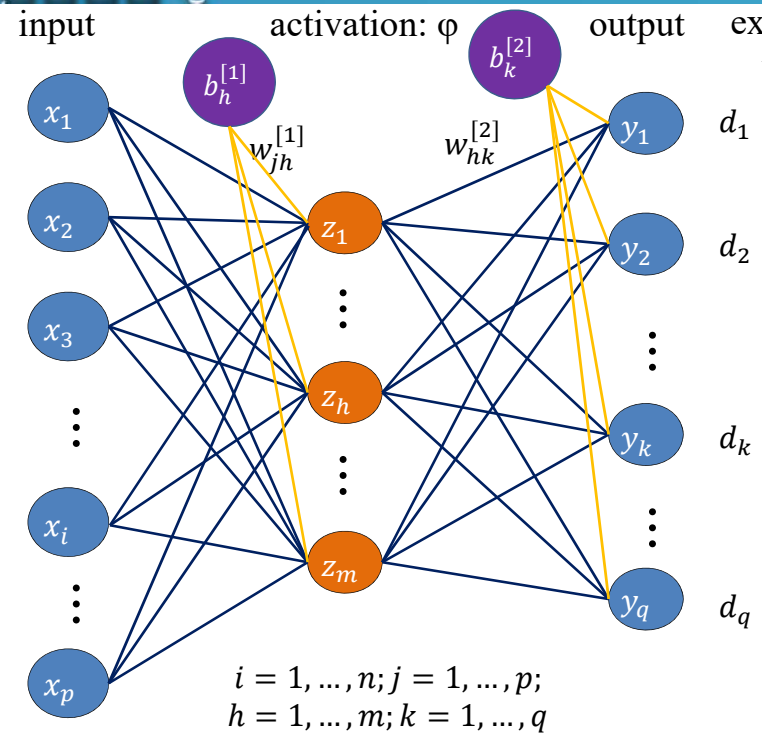
$$\frac{\partial J_{sigmoid}(w)}{\partial w} = -\frac{Y}{A} + \frac{1 - Y}{1 - A}$$

Derivative of Cross-Entropy Loss with Softmax

$$J_{softmax} = -\sum y_k \log(\hat{y}_k)$$

$$\frac{\partial J_{softmax}}{\partial o_i} = \hat{y}_i - y_i$$

單隱藏層前饋神經網路: 反向傳播演算法



$i = 1, \dots, n; j = 1, \dots, p;$
$h = 1, \dots, m; k = 1, \dots, q$

$$z_h = \phi^{(1)}(u_h) = \phi^{(1)}\left(\sum_{j=1}^{p} x_j \cdot w_{jh}^{[1]} + b_h^{[1]}\right), h = 1, \cdots, m.$$

$$y_k = \phi^{(2)}(v_k) = \phi^{(2)}\left(\sum_{h=1}^{m} z_h \cdot w_{hk}^{[2]} + b_k^{[2]}\right), k = 1, \cdots, q.$$

$$\phi^{(1)} = \phi^{(2)} = \phi$$

誤差值

$$E = \frac{1}{2}\sum_{k=1}^{q}(d_k - y_k)^2$$

梯度下降學習法

$$\Delta w = -\alpha\frac{\partial E}{\partial w}$$

啟動函數

$$\phi(x) = \frac{1}{1+\exp(-x)}$$

$$\phi'(x) = \phi(x)(1-\phi(x))$$

隱藏層與輸出層的連結權重調整

$$\frac{\partial E}{\partial w_{hk}^{[2]}} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial v_k}\frac{\partial v_k}{\partial w_{hk}^{[2]}}$$

$$= -(d_k - y_k)\cdot\phi'(v_k)\cdot z_h$$

$$= -(d_k - y_k)\cdot\phi(v_k)(1-\phi(v_k))\cdot z_h$$

$$\Delta w_{hk}^{[2]} = \alpha\delta_k z_h, \text{where } \delta_k = (d_k - y_k)y_k(1-y_k)$$

$$\Delta w_{hk}^{[2](i)} = \alpha\delta_k^{(i)} z_h^{(i)}, \text{where } \delta_k^{(i)} = (d_k^{(i)} - y_k^{(i)})y_k^{(i)}(1-y_k^{(i)})$$

$$w_{hk}^{[2](i+1)} = w_{hk}^{[2](i)} + \Delta w_{hk}^{[2](i)}$$

www.nature.com › letters - 翻譯這個網頁
Learning representations by back-propagating errors | Nature
1986年10月9日 - We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of ...
由 DE Rumelhart 著作 - 1986 - 被引用 22470 次 - 相關文章

反向傳播(Backpropagation,BP)是「誤差反向傳播」的簡稱，是一種與最優化方法(如梯度下降法)結合使用的，用來訓練人工神經網絡的常見方法。該方法對網絡中所有權重計算損失函數的梯度。這個梯度會反饋給最優化方法，用來更新權值以最小化損失函數。

維基百科 自由的百科全書

輸入層與隱藏層的連結權重調整

$$\frac{\partial E}{\partial w_{jh}^{[1]}} = \frac{\partial E}{\partial z_h}\frac{\partial z_h}{\partial u_h}\frac{\partial u_h}{\partial w_{jh}^{[1]}}$$

$$= \sum_k\left(\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial v_k}\frac{\partial v_k}{\partial z_h}\right)\cdot\phi'(u_h)\cdot x_j$$

$$= -\left(\sum_k(d_k - y_k)\cdot\phi'(v_k)\cdot w_{hk}^{[2]}\right)\cdot\phi'(u_h)\cdot x_j$$

$$= -\left(\sum_k \delta_k \cdot w_{hk}^{[2]}\right)\cdot\phi(u_h)(1-\phi(u_h))\cdot x_j$$

$$\Delta w_{jh}^{[1]} = \alpha\delta_h x_j, \text{where } \delta_h = \left(\sum_k \delta_k \cdot w_{hk}^{[2]}\right) z_h(1-z_h)$$

$$\Delta w_{jh}^{[1](i)} = \alpha\delta_j^{(i)} x_j^{(i)}, \text{where } \delta_h^{(i)} = \left(\sum_k \delta_k^{(i)} \cdot w_{hk}^{[2](i)}\right) z_h^{(i)}(1-z_h^{(i)})$$

$$w_{jh}^{[1](i+1)} = w_{jh}^{[1](i)} + \Delta w_{jh}^{[1](i)}$$

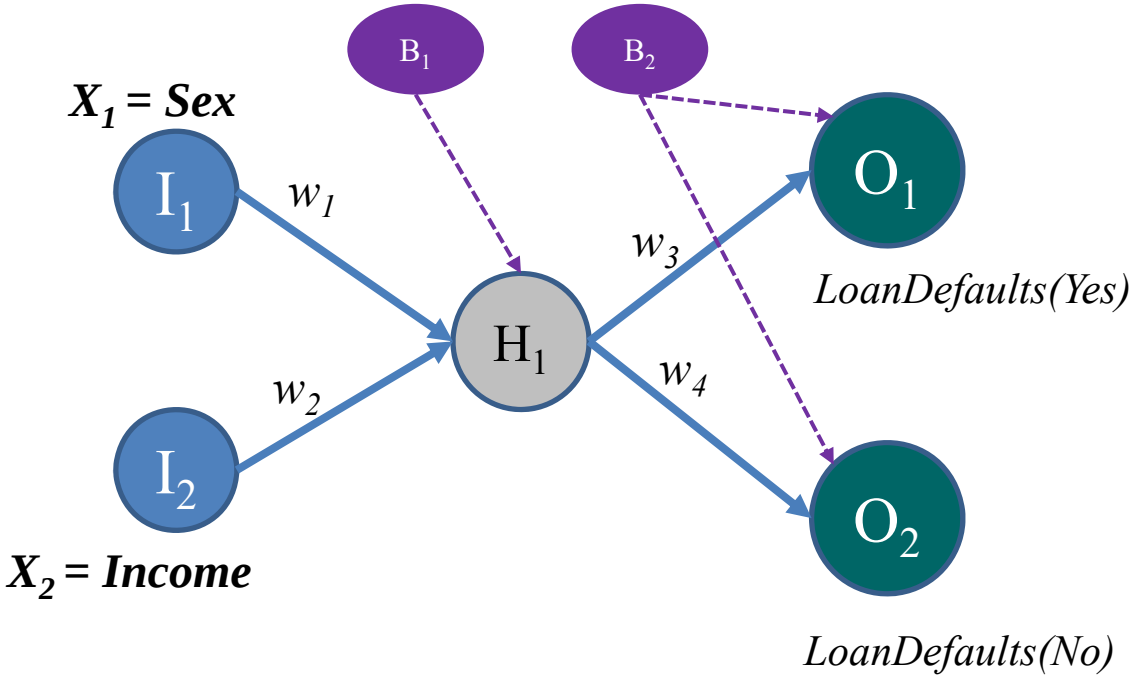
單隱藏層前饋神經網路演算過程

初始值

輸入值I_1: 0
輸入值I_2: 1
期望輸出值O_1: 0 (O_1Ex)
期望輸出值O_2: 1 (O_2Ex)

$I_1 \rightarrow H_1$權重值w_1: 0.01
$I_2 \rightarrow H_1$權重值w_2: 0.01
$H_1 \rightarrow O_1$權重值w_3: 0.01
$H_1 \rightarrow O_2$權重值w_4: 0.01
偏誤權值B_1H_1: 0.01
偏誤權值B_2O_1: 0.01
偏誤權值B_2O_2: 0.01

學習率α : 0.1



真實輸出值H1:
0.01
0.5025

真實輸出值O1:
0.015025
0.5037562

真實輸出O2:
0.015025
0.5037562

O1誤差: 0.1268851
O2誤差: 0.123129
平均誤差: 0.5000141

$H_1 = I_1 \times w_1 + I_2 \times w_2 + B_1H_1$

$\varphi(H_1) = 1/(1 + \exp(-H_1))$

$O_1 = \varphi(H_1) \times w_3 + B_2O_1$

$\varphi(O_1) = 1/(1 + \exp(-O_1))$

$O_2 = \varphi(H_1) \times w_4 + B_2O_2$

$\varphi(O_2) = 1/(1 + \exp(-O_2))$

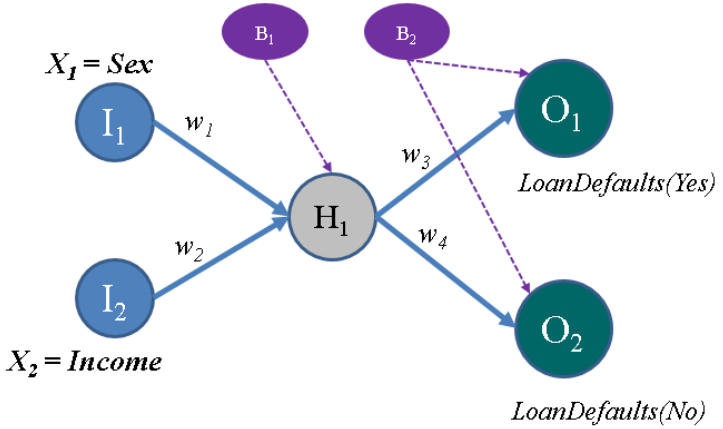
$O_1Err = (O_1Ex - F(O_1))^2/2$

$O_2Err = (O_2Ex - F(O_2))^2/2$

$AveErr = \sqrt{O_1Err + O_2Err}$

單隱藏層前饋神經網路: 權重更新過程 (反向傳播演算法+gradient descent method)

- 誤差項: 節點對誤差的影響
- 輸出層節點對誤差的影響，可看成真實output與期望output的差。
- 隱藏層節點對誤差的影響，可由後面一層的誤差項算出。



The gradient of the sigmoid function:

$$\phi(x) = \frac{1}{1+\exp(-x)}$$

$$\phi'(x) = \phi(x)(1-\phi(x))$$

O1真實輸出X (1 - O1真實輸出) X (O1期望輸出 - O1真實輸出)

O_1誤差項: -0.1259319

$$O_1\text{ErrTerm} = \varphi(O_1) \times (1 - \varphi(O_1)) \times (O_1\text{Ex} - \varphi(O_1))$$

O_2誤差項: 0.124054

$$O_2\text{ErrTerm} = \varphi(O_2) \times (1 - \varphi(O_2)) \times (O_2\text{Ex} - \varphi(O_2))$$

H_1誤差項: -4.695E-06

$$H_1\text{ErrTerm} = \varphi(H_1) \times (1 - \varphi(H_1)) \times (O_1\text{ErrTerm} \times w_3 + O_2\text{ErrTerm} \times w_4)$$

更新權重值w_3: 0.0036719

$$w_3(t+1) = w_3(t) + \alpha \times O_1\text{ErrTerm} \times \varphi(H_1)(t)$$

更新權重值w_4: 0.016337

$$w_4(t+1) = w_4(t) + \alpha \times O_2\text{ErrTerm} \times \varphi(H_1)(t)$$

更新權重值w_1: 0.01

$$w_1(t+1) = w_1(t) + \alpha \times H_1\text{ErrTerm} \times I_1$$

更新權重值w_2: 0.01

$$w_2(t+1) = w_2(t) + \alpha \times H_1\text{ErrTerm} \times I_2$$

更新偏誤權值B_1H_1: 0.009995

$$B_1H_1(t+1) = B_1H_1(t) + \alpha \times H_1\text{ErrTerm}(t)$$

更新偏誤權值B_2O_1: -0.0025932

$$B_2O_1(t+1) = B_2O_1(t) + \alpha \times O_1\text{ErrTerm}(t)$$

更新偏誤權值B_2O_2: 0.0224054

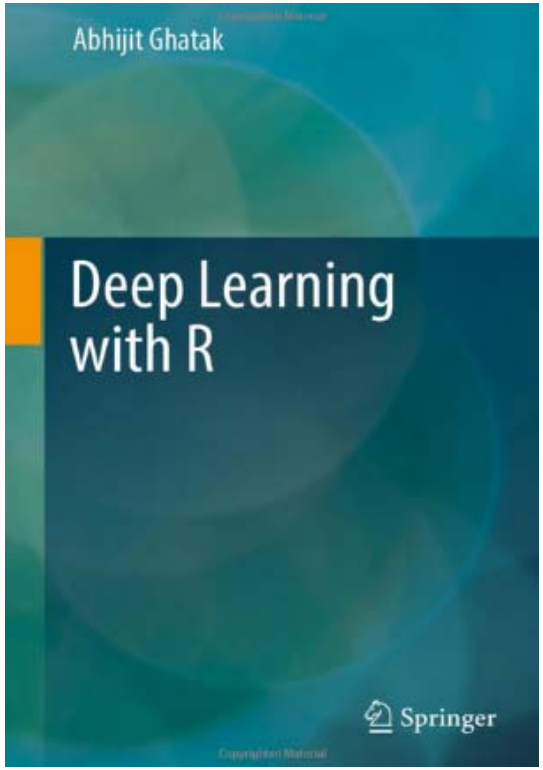
$$B_2O_2(t+1) = B_2O_2(t) + \alpha \times O_2\text{ErrTerm}(t)$$

R程式演示 (`nnet` package) 見23/141頁。

Writing a Simple Neural Network Application

Write R code to:

- define the architecture of a neural network (無隱藏層),
- initialize its parameters,
- calculate the the predictions using forward propagation,
- calculate the loss,
- correct the loss using a backpropagation algorithm (we will use the simple gradient descent method),
- iterate the same procedure, till we arrive at a negligible cost.



Prepare Data: Example (1): iris data

```
> library(caret)
> set.seed(12345)
> iris.two.species <- iris[51:150, ]
> iris.two.species[,5] <- ifelse(iris.two.species[,5] == "setosa", 1, 0)
> id <- createDataPartition(y = 1:nrow(iris.two.species), p = 0.8)
> # note that the dim of the data is p by n
> train.data <- iris.two.species[id$Resample1, ]
> myTrain.X <- t(train.data[, -5])
> myTrain.y <- t(train.data[, 5])
> head(myTrain.X)
             51  52  53  54  55  56  57  58  60  61  62  63  64  65  66  68
Sepal.Length 7.0 6.4 6.9 5.5 6.5 5.7 6.3 4.9 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.8
Sepal.Width  3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.7 2.0 3.0 2.2 2.9 2.9 3.1 2.7
Petal.Length 4.7 4.5 4.9 4.0 4.6 4.5 4.7 3.3 3.9 3.5 4.2 4.0 4.7 3.6 4.4 4.1
Petal.Width  1.4 1.5 1.5 1.3 1.5 1.3 1.6 1.0 1.4 1.0 1.5 1.0 1.4 1.3 1.4 1.0
...
> dim(myTrain.X)
[1]  4 80
> length(myTrain.y)
[1] 80
> test.data <- iris.two.species[-id$Resample1, ]
> myTest.X <- t(test.data[, -5])
> myTest.y <- t(test.data[, 5])
> dim(myTest.X)
[1]  4 20
> length(myTest.y)
[1] 20
> num.iter <- 1000
> learning.rate <- 0.01
```

Prepare Data: Example (2): dogs-vs-cats image data

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
+   install.packages("BiocManager")
> BiocManager::install("EBImage")
Bioconductor version 3.11 (BiocManager 1.30.10), R 4.0.2 (2020-06-22)
Installing package(s) 'EBImage'
...
> library(EBImage)
> library(pbapply)
```

https://www.kaggle.com/c/dogs-vs-cats/data

```
> ########################################
> # read and show original images        #
> ########################################
> #file.path.train <- "data/Dogs_vs_Cats/train"  #25000 files
> #file.path.test <- "data/Dogs_vs_Cats/test1"  #12500 files
>
> file.path.train <- "data/Dogs_vs_Cats/train_subset" #1250 files
> file.path.test <- "data/Dogs_vs_Cats/test1_subset" #50 files
>
> par(mfrow = c(1, 2))
> show.train.id <- 1
> images.train <- list.files(file.path.train)
> img.train <- readImage(file.path(file.path.train, images.train[show.train.id]))
> EBImage::display(img.train, method = "raster")
>
> show.test.id <- 20
> images.test <- list.files(file.path.test)
> img.test <- readImage(file.path(file.path.test, images.test[show.test.id]))
> EBImage::display(img.test, method = "raster")
```

Extract image features by resize

```
extract_feature <- function(file.dir, width, height) {

  img.size <- width * height
  images <- list.files(file.dir)

  print(paste("Processing", length(images), "images@", file.dir))
  label <- ifelse(grepl("dog", images) == T, 1, 0)

  feature.list <- pblapply(images, function(img.name) {

    # img.name <- "cat.9993.jpg"
    img <- readImage(file.path(file.dir, img.name))
    img.resized <- EBImage::resize(img, w = width, h = height)

    ## show images
    # display(img, method = "raster")
    # display(img.resized, method = "raster")

    img.matrix <- matrix(reticulate::array_reshape(img.resized,
                                                   (width*height*channels)),
                         nrow = width*height*channels)
    img.vector <- as.vector(t(img.matrix))
    img.vector
  })

  feature.matrix <- do.call(rbind, feature.list)
  list(t(feature.matrix), label)
}
```

```
# pbapply {pbapply}: Adding
Progress Bar to '*apply'
Functions
```

Extract image features

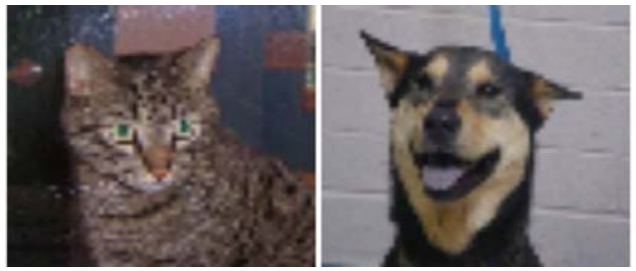
```
> height <- 64
> width <- 64
> channels <- 3
>
> train.data <- extract_feature(file.path.train, width, height)
[1] "Processing 1250 images@ data/Dogs_vs_Cats/train_subset"
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=10s
> myTrain.X <- train.data[[1]]
> myTrain.y <- train.data[[2]]
> dim(myTrain.X)
[1] 12288  1250
> length(myTrain.y)
[1] 1250
>
> test.data <- extract_feature(file.path.test, width, height)
[1] "Processing 50 images@ data/Dogs_vs_Cats/test1_subset"
  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100% elapsed=00s
> str(test.data)
List of 2
 $ : num [1:12288, 1:50] 0.137 0.239 0.376 0.084 0.186 ...
 $ : num [1:50] 0 0 0 0 0 0 0 0 0 0 ...
> myTest.X <- test.data[[1]]
> myTest.y <- test.data[[2]]
> length(myTest.y)
[1] 50
> dim(myTest.X)
[1] 12288    50
```

64*64*3

Show resized images

```
data2Image <- function(xdata, width, height){
  ar <- array(0, dim=c(width, height, 3))
  lapply(1:3, function(i){
    ar[,,i] <<- matrix(xdata[seq(i, length(xdata), 3)],
                       width, height, byrow=T)
  })

  # return an Image class
  Image(ar, dim=c(width, height, 3), colormode=2)
}
```

```
> images.train.resized <- Image(ar, dim=c(width, height, 3), colormode=2)
> images.train.resized <- data2Image(myTrain.X[, show.train.id], width, height)
> images.test.resized <- data2Image(myTest.X[, show.test.id], width, height)
> par(mfrow = c(1, 2))
> EBImage::display(images.train.resized, method = "raster")
> EBImage::display(images.test.resized, method = "raster")
>
>
> # nomalization for each columns?
> myTrain.X <- scale(myTrain.X)
> myTest.X <- scale(myTest.X)
>
> num.iter <- 1000
> learning.rate <- 0.01
```



Simple Neural Network

```r
# activation function:sigmoid
sigmoid <- function(x){
  1/(1+exp(-x))
}
```

```r
# initializing our weight vector and
# bias scalars to zero.
initialize_with_zeros <- function(dim){
  W <- matrix(0, nrow = dim, ncol = 1)
  b <- 0
  list(W = W, b = b)
}
```

```r
# carry out the forward/backward propagation of the network to learn our parameters.
propagate <- function(W, b, X, y){

  n <- ncol(X)  # number of observations of the training data

  # Forward Propagation
  A <- sigmoid((t(W) %*% X) + b)

  # cost is the negative log-likelihood cost of the logistic regression
  cost <- (-1/n) * sum(y * log(A) + (1-y) * log(1 - A))

  ## Backward Propagation
  # dw: the gradient of the cost with respect to w
  dW <- (1/n) * (X %*% t(A-y))
  # db: the gradient of the cost with respect to b
  db <- (1/n) * rowSums(A-y)

  list(gradient = list(dW, db), cost = cost)
}
```

Optimization and prediction functions

```
optimize <- function(W, b, X, y, num.iter, learning.rate, print.cost = FALSE) {
  cost <- numeric(num.iter)

  for (i in 1:num.iter) {

    # calculate gradient and cost
    grad.cost <- propagate(W, b, X, y)
    gradient <- grad.cost$gradient
    cost[i] <- grad.cost$cost

    # Retrieve the derivatives
    dW <- matrix(gradient$dW)
    db <- gradient$db

    # Update the parameters
    W <- W - learning.rate * dW
    b <- b - learning.rate * db

    # Print the cost every 100th iteration
    if ((print.cost == T) & (i%%100 == 0)) {
      cat(sprintf("Cost after iteration %d: %06f\n", i, cost[i]))
    }

    params <- list(W=W, b=b)
    gradient <- list(dW=dW, db=db)
  }

  list(params = params, gradient = gradient, cost = cost)
}
```

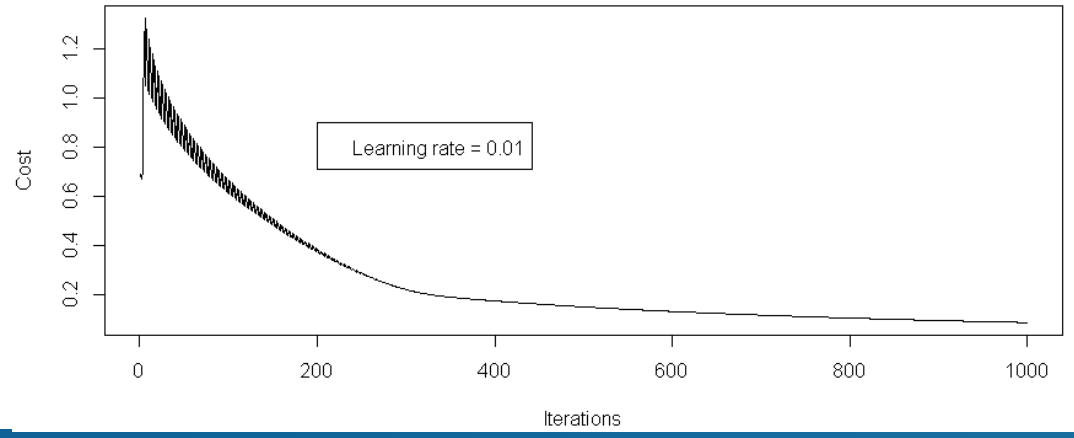
```
# Activation vector A to predict
# the probability of a dog/cat
pred <- function(W, b, X) {
  n <- ncol(X)
  Y.prediction <- numeric(n)
  A <- sigmoid((t(W) %*% X) + b)
  for (i in 1:n) {
    if (A[1, i] > 0.5) {
      Y.prediction[i] <- 1
    } else{
      Y.prediction[i] <- 0
    }
  }
  Y.prediction
}
```

Simple Neural Network

```
simple_model <- function(train.X, train.y, test.X, test.y,
                         num.iter, learning.rate, print.cost = FALSE){

  Wb <- initialize_with_zeros(nrow(train.X))
  W <- Wb$W
  b <- Wb$b

  optFn.output <- optimize(W, b, train.X, train.y,
                           num.iter, learning.rate, print.cost)

  W <- as.matrix(optFn.output$params$W)
  b <- optFn.output$params$b

  pred.train <- pred(W, b, train.X)
  pred.test <- pred(W, b, test.X)

  cat(sprintf("train accuracy: %#.2f \n", mean(pred.train == train.y) * 100))
  cat(sprintf("test accuracy: %#.2f \n", mean(pred.test == test.y) * 100))
  res <- list(cost = optFn.output$cost,
              pred.train = pred.train, pred.test = pred.test, W = W, b = b)
  res
}
```

Run the simple neural network model

```
> SNN.model <- simple_model(myTrain.X, myTrain.y, myTest.X, myTest.y,
+                           num.iter, learning.rate, print.cost = T)
Cost after iteration 100: 0.667832
Cost after iteration 200: 0.386553
Cost after iteration 300: 0.223154
Cost after iteration 400: 0.176600
Cost after iteration 500: 0.152448
Cost after iteration 600: 0.133868
Cost after iteration 700: 0.119117
Cost after iteration 800: 0.107133
Cost after iteration 900: 0.097220
Cost after iteration 1000: 0.088897
train accuracy: 99.76
test accuracy: 56.00
>
> plot(1:length(SNN.model$cost), SNN.model$cost,
+      type="l", xlab = "Iterations", ylab = "Cost")
> legend(200, 0.9, c("Learning rate = 0.01"))
```



Writing a Deep Neural Network (DNN) Algorithm

Neural Networks and Deep Learning in R

CRAN Task View: Machine Learning & Statistical Learning

Maintainer: Torsten Hothorn
Contact: Torsten.Hothorn at R-project.org
Version: 2020-02-20
URL: https://CRAN.R-project.org/view=MachineLearning

Several add-on packages implement ideas and methods developed at the borderline between computer science and statistics - this field of research is usually referred to as machine learning. The packages can be roughly structured into the following topics:

- *Neural Networks and Deep Learning* : Single-hidden-layer neural network are implemented in package nnet (shipped with base R). Package RSNNS offers an interface to the Stuttgart Neural Network Simulator (SNNS). Packages implementing deep learning flavours of neural networks include deepnet (feed-forward neural network, restricted Boltzmann machine, deep belief network, stacked autoencoders), RcppDL (denoising autoencoder, stacked denoising autoencoder, restricted Boltzmann machine, deep belief network) and h2o (feed-forward neural network, deep autoencoders). An interface to tensorflow is available in tensorflow.

Neural Networks and Deep Learning, Recursive Partitioning, Random Forests , Regularized and Shrinkage Methods, Boosting and Gradient Descent :, Support Vector Machines and Kernel Methods, Bayesian Methods , Optimization using Genetic Algorithms , Association Rules, Fuzzy Rule-based Systems, Model selection and validation, Other procedures, Meta packages , GUI , Visualisation,

https://cran.r-project.org/web/views/MachineLearning.html

Deep Learning in R: Short Overview of Packages

- **SAENET**: A Stacked Autoencoder Implementation with Interface to 'neuralnet'
- **darch**: Package for Deep Architectures and Restricted Boltzmann Machines

> Package 'SAENET' was removed from the CRAN repository.
>
> Formerly available versions can be obtained from the archive.
>
> Archived on 2018-05-03 as check problems were not corrected despite reminders.
>
> Please use the canonical form https://CRAN.R-project.org/package=SAENET to link to this page.

- **AMORE**: Artificial Neural Network Training and Simulating
  - Version: 0.2-16; Published: 2020-02-12
  - https://cran.r-project.org/web/packages/AMORE/index.html
  - Commands for training a simulating an artificial neural network.

- **autoencoder**: Sparse Autoencoder for Automatic Learning of Representative Features from Unlabeled Data
  - Version: 1.1; Published: 2015-07-02
  - https://cran.r-project.org/web/packages/autoencoder/index.html
  - Implementation of the sparse autoencoder in R environment, following the notes of Andrew Ng (http://www.stanford.edu/class/archive/cs/cs294a/cs294a.1104/sparseAutoencoder.pdf). The features learned by the hidden layer of the autoencoder (through unsupervised learning of unlabeled data) can be used in constructing deep belief neural networks.

Deep Learning in R: Short Overview of Packages

- **deepnet: deep learning toolkit in R** (Version: 0.2; Published: 2014-03-20)
  - https://cran.r-project.org/web/packages/deepnet/index.html
  - Implement some deep learning architectures and neural network algorithms, including **BP**, **RBM**, **DBN**, **Deep autoencoder** and so on.
- **deepNN**: **Deep Learning** (Version: 1.0; Published: 2020-03-05)
  - https://cran.r-project.org/web/packages/deepNN/index.html
  - Implementation of some Deep Learning methods. Includes multilayer perceptron, different activation functions, regularisation strategies, stochastic gradient descent and dropout.
- **deepr** (Version: 0.0.8; Published: 2015-05-04)
  - https://github.com/woobe/deepr
  - An R package to streamline the training, fine-tuning and predicting processes for deep learning. It aims to further simplify the functions in packages such as 'h2o' and 'deepnet'.
- **FCNN4R**: **Fast Compressed Neural Networks for R** (Version: 0.6.2; Published: 2016-03-09)
  - http://cran.nexr.com/web/packages/FCNN4R/index.html
  - Provides an interface to kernel routines from the **FCNN C++ library**.
- **kerasR**: **R Interface to the Keras Deep Learning Library** (Version: 0.6.1, Published: 2017-06-01)
  - https://cran.r-project.org/web/packages/kerasR/index.html
  - Provides a consistent interface to the 'Keras' Deep Learning Library directly from within R. 'Keras' provides specifications for describing dense neural networks, convolution neural networks (CNN) and recurrent neural networks (RNN) running on top of either 'TensorFlow' or 'Theano'.

https://srdas.github.io/DLBook/DeepLearningWithR.html

Deep Learning in R: Short Overview of Packages

- **neuralnet**: Training of Neural Networks
  - Version: 1.44.2; Published: 2019-02-07
  - https://cran.r-project.org/web/packages/neuralnet/index.html
  - Training of neural networks using backpropagation, resilient backpropagation with (Riedmiller, 1994) or without weight backtracking (Riedmiller and Braun, 1993) or the modified globally convergent version by Anastasiadis et al. (2005). The package allows flexible settings through custom-choice of error and activation function. Furthermore, the calculation of generalized weights (Intrator O & Intrator N, 1993) is implemented.
- **NeuralNetTools**: Visualization and Analysis Tools for Neural Networks
  - Version: 1.5.2; Published: 2018-07-26
  - https://cran.r-project.org/web/packages/NeuralNetTools/index.html
  - Visualization and analysis tools to aid in the interpretation of neural network models. Functions are available for plotting, quantifying variable importance, conducting a sensitivity analysis, and obtaining a simple list of model weights.
- **nnet**: Feed-Forward Neural Networks and Multinomial Log-Linear Models
  - Version: 7.3-14; Published: 2020-04-26
  - https://cran.r-project.org/web/packages/nnet/index.html
  - Software for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models.

Deep Learning in R: Short Overview of Packages

- **RcppDL**: Deep Learning Methods via Rcpp
  - Version: 0.0.5; Published: 2014-12-17
  - https://cran.r-project.org/web/packages/RcppDL/index.html
  - This package is based on the C++ code from Yusuke Sugomori, which implements basic machine learning methods with many layers (deep learning), including **dA** (Denoising Autoencoder), **SdA** (Stacked Denoising Autoencoder), **RBM** (Restricted Boltzmann machine) and **DBN** (Deep Belief Nets).

- **rnn**: Recurrent Neural Network
  - Version: 1.4.0; Published: 2020-07-03 (Vignettes)
  - https://cran.r-project.org/web/packages/rnn/index.html
  - Implementation of a Recurrent Neural Network architectures in native R, including Long Short-Term Memory (Hochreiter and Schmidhuber, <doi:10.1162/neco.1997.9.8.1735>), Gated Recurrent Unit (Chung et al., <arXiv:1412.3555>) and vanilla RNN.

- **RSNNS**: Neural Networks using the Stuttgart Neural Network Simulator (SNNS)
  - Version: 0.4-12; Published: 2019-09-17
  - https://cran.r-project.org/web/packages/RSNNS/index.html
  - The **Stuttgart Neural Network Simulator (SNNS)** is a library containing many standard implementations of neural networks. This package wraps the SNNS functionality to make it available from within R. Using the 'RSNNS' low-level interface, all of the algorithmic functionality and flexibility of SNNS can be accessed. Furthermore, the package contains a convenient high-level interface, so that the most common neural network topologies and learning algorithms integrate seamlessly into R.

Summary of NN methods

R Packages	DNN Methods
AMORE	Multilayer Feedforward NN (adaptative BP with momentum method)
autoencoder	Autoencoder NN (unsupervised) + BP
darch	Hinton, G. E., S. Osindero, Y. W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18(7), S. 1527-1554, DOI: 10.1162/neco.2006.18.7.1527, 2006.
deepnet	Restricted Boltzmann Machine, DNN with weights initialized by StackedAutoEncoder, Mutiple hidden layers NN by BP
deepNN	BP for a multilayer perceptron
FCNN4R	Fast Compressed NN
kerasR	convolution neural networks (CNN) and recurrent neural networks (RNN)
neuralnet	NN with BP, resilient BP
NeuralNetTools	Plot a neural interpretation diagram for a neural network object (nnet, neuralnet)
nnet	Feedforward single-hidden-layer NN and Multinomial Log-Linear Model
RcppDL	dA (Denoising Autoencoder), SdA (Stacked Denoising Autoencoder), RBM (Restricted Boltzmann machine) and DBN (Deep Belief Nets).
rnn	Recurrent NN, Long Short-Term Memory (LSTM), Gated Recurrent Unit and vanilla RNN.
RSNNS	multilayer perceptron (MLP), Jordan networks/Elman networks (partially recurrent networks), radial basis function (RBF) network.
SAENET	Stacked Autoencoder

The R interface to the deep-learning framework

- **h2o**: R Interface for the 'H2O' Scalable Machine Learning Platform
  - Version: 3.30.0.1; Published: 2020-04-09
  - https://cran.r-project.org/web/packages/h2o/index.html
  - R interface for 'H2O', the scalable open source machine learning platform that offers parallelized implementations of many supervised and unsupervised machine learning algorithms such as Generalized Linear Models, Gradient Boosting Machines (including XGBoost), Random Forests, Deep Neural Networks (Deep Learning), Stacked Ensembles, Naive Bayes, Cox Proportional Hazards, K-Means, PCA, Word2Vec, as well as a fully automatic machine learning algorithm (AutoML).
  - https://www.h2o.ai/products/h2o/

- **MXNet** - R API: A Flexible and Efficient Library for Deep Learning
  - https://mxnet.apache.org/versions/1.6/api/r
  - MXNet supports the R programming language. The MXNet R package brings flexible and efficient GPU computing and state-of-art deep learning to R. It enables you to write seamless tensor/matrix computation with multiple GPUs in R. It also lets you construct and customize the state-of-art deep learning models in R, and apply them to tasks, such as image classification and data science challenges.

The R interface to the deep-learning framework

- **tfestimators**: Interface to 'TensorFlow' Estimators
  - Version: 1.9.1; Published: 2018-11-07
  - https://cran.r-project.org/web/packages/tfestimators/index.html
  - Interface to 'TensorFlow' Estimators <https://www.tensorflow.org/programmers_guide/estimators>, a high-level API that provides implementations of many different model types including linear models and deep neural networks.

核心是C++開發的，封裝是用Python。R的TensorFlow套件調用Python TensorFlow API來執行計算模型。

- **TensorFlow**: R Interface to 'TensorFlow'
  - Version: 2.2.0; Published: 2020-05-11
  - https://cran.r-project.org/web/packages/tensorflow/index.html
  - Interface to 'TensorFlow' <https://www.tensorflow.org/>, an open source software library for numerical computation using data flow graphs. **Nodes** in the graph represent mathematical operations, while the **graph edges** represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more 'CPUs' or 'GPUs' in a desktop, server, or mobile device with a single 'API'. 'TensorFlow' was originally developed by researchers and engineers working on the **Google Brain Team** within Google's Machine Intelligence re search organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

The R interface to the deep-learning framework

- **keras: R Interface to "Keras"**
  - Version: 2.3.0.0; Published: 2020-05-19
  - https://cran.r-project.org/web/packages/keras/index.html
  - Interface to 'Keras' <https://keras.io>, a high-level neural networks 'API'. 'Keras' was developed with a focus on enabling fast experimentation, supports both convolution based networks and recurrent networks (as well as combinations of the two), and runs seamlessly on both 'CPU' and 'GPU' devices.

範例: 利用"nnet"構建一個人工神經網路來預測iris資料集之品種

```
> # install.packages("nnet")
> library(nnet)
> xdata <- iris[, 1:4]
> ydata <- iris[, 5]
> no.class <- length(unique(ydata))
> T <- matrix(0, nrow=nrow(xdata), ncol=no.class)
> sapply(1:no.class, function(x){
+     T[which(ydata == levels(ydata)[x]), x] <<- 1
+ })
> id <- sample(1:nrow(iris), 100)
> training.x <- xdata[id,]
> training.y <- T[id, ]
> testing.x <- xdata[-id,]
> testing.y <- T[-id, ]
> nn.iris <- nnet(training.x, training.y, size = 2, rang = 0.1,
+             decay = 5e-4, maxit = 200)
# weights:  19
initial  value 73.240851
iter  10 value 37.694980
iter  20 value 32.664756
iter  30 value 31.946509
...
> pred <- max.col(predict(nn.iris, testing.x))
> testing.true <- apply(T[-id, ], 1, which.max)
> table(testing.true, pred)
             pred
testing.true  1  2  3
           1 14  0  0
           2  0 13  2
           3  0  0 21
```

nnet: 單一隱藏層的feed-forward neural networks套件

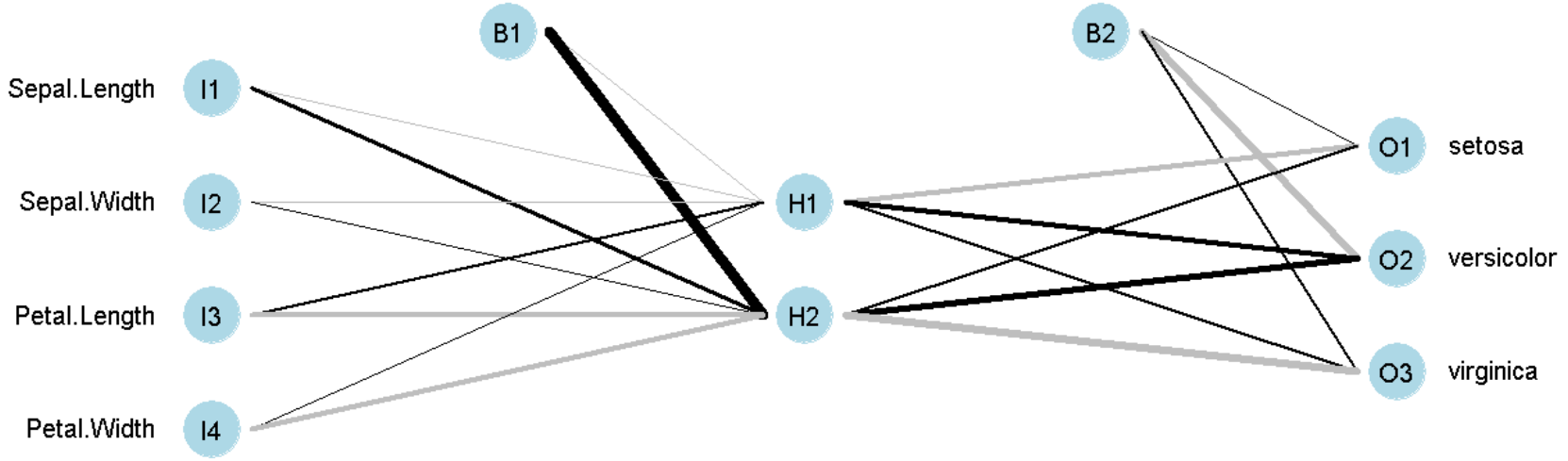
```
> T
        [,1] [,2] [,3]
  [1,]    1    0    0
  [2,]    1    0    0
  [3,]    1    0    0
  [4,]    1    0    0
...
[148,]    0    0    1
[149,]    0    0    1
[150,]    0    0    1
```

- **size**: number of units in the hidden layer.
- **rang**: initial random weights on [-rang, rang]. Value about 0.5 unless the inputs are large, in which case it should be chosen so that rang * max(|x|) is about 1.
- **decay**: parameter for weight decay. Default 0. (It's regularization to avoid over-fitting.)
- **maxit**: maximum number of iterations. Default 100.

範例: 利用"nnet"構建一個人工神經網路來預測iris資料集之品種

```
> str(nn.iris)
List of 15
 $ n             : num [1:3] 4 2 3
 ...
$ wts           : num [1:19] -0.209 -0.309 -1.996 2.634 1.349 ...
 $ convergence  : int 1
 $ fitted.values: num [1:100, 1:3] 0.99229 0.009657 0.993522 0.011416 0.000118 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:100] "13" "93" "23" "94" ...
  .. ..$ : NULL
...
> summary(nn.iris)
a 4-2-3 network with 19 weights
options were - decay=5e-04
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1
 -0.21  -0.31  -2.00   2.63   1.35
...
 b->o3 h1->o3 h2->o3
  3.26   3.61 -13.76
> print(nn.iris)
a 4-2-3 network with 19 weights
options were - decay=5e-04
> coef(nn.iris)
      b->h1       i1->h1       i2->h1       i3->h1       i4->h1        b->h2
 -0.2088855   -0.3088326   -1.9963053    2.6339182    1.3487065   20.6682103
     i1->h2       i2->h2       i3->h2       i4->h2        b->o1       h1->o1
```

範例: 利用"nnet"構建一個人工神經網路來預測iris資料集之品種




```
# install.packages("NeuralNetTools")
library(NeuralNetTools)
plotnet(nn.iris)
```

```
# or
nn.iris.2 <- nnet(Species ~ ., data = iris, subset = id,
                  size = 2, rang = 0.1,
                  decay = 5e-4, maxit = 200)
table(iris$Species[-id], predict(nn.iris.2, iris[-id,], type = "class"))
plotnet(nn.iris.2)
```

See also: https://rpubs.com/skydome20/R-Note8-ANN

範例: 利用"neuralnet "構建一個深度神經網路來逼近一個函數 y=x^2

neuralnet: 多個隱藏層的類神經網路R套件

利用" **neuralnet** "構建一個深度神經網路來逼近一個函數 y=x^2

neuralnet {neuralnet} R Documentation

Training of neural networks

Description

Train neural networks using backpropagation, resilient backpropagation (RPROP) with (Riedmiller, 1994) or without weight backtracking (Riedmiller and Braun, 1993) or the modified globally convergent version (GRPROP) by Anastasiadis et al. (2005). The function allows flexible settings through custom-choice of error and activation function. Furthermore, the calculation of generalized weights (Intrator O. and Intrator N., 1993) is implemented.

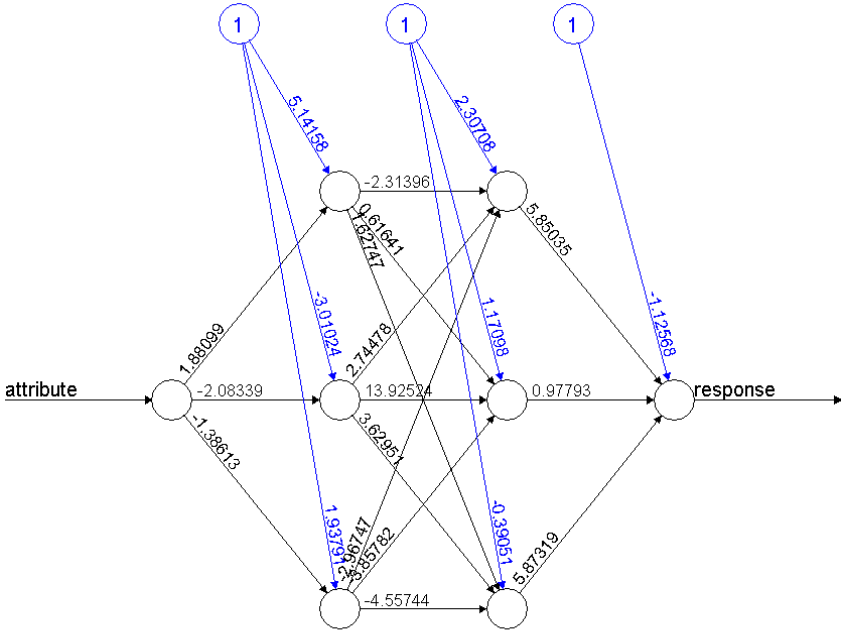
```
# install.packages("neuralnet")
library(neuralnet)
set.seed(12345)
x <- sample(seq(-2, 2, length.out=50), 50, replace=F)
f <- function(x){
  x^2
}
y <- f(x)
train.data <- data.frame(attribute=x, response=y)
head(train.data)
plot(train.data, main="f(x)=x^2")

fit <- neuralnet(response ~ attribute,
                 data=train.data,
                 hidden=c(3, 3),
                 threshold=0.01)
fit
plot(fit)
```

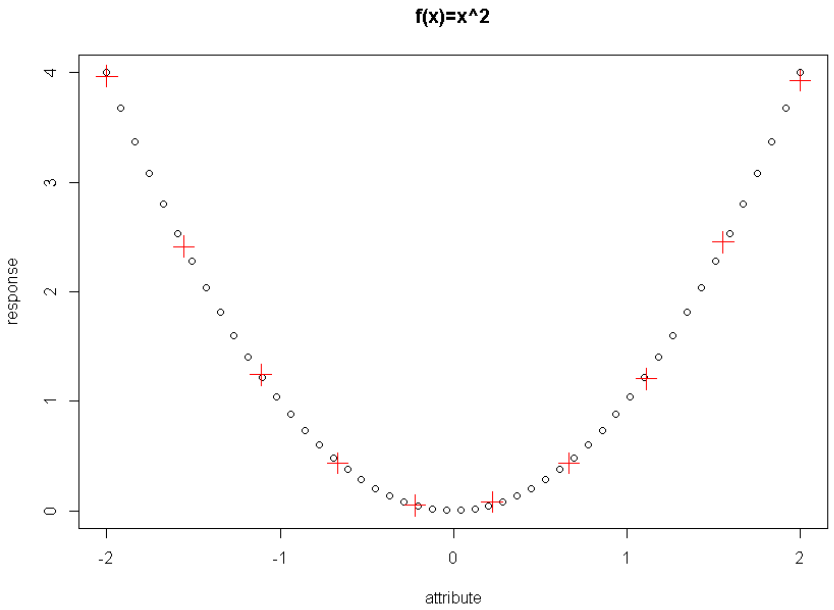
Usage

```
neuralnet(formula, data, hidden = 1, threshold = 0.01,
  stepmax = 1e+05, rep = 1, startweights = NULL,
  learningrate.limit = NULL, learningrate.factor = list(minus = 0.5,
  plus = 1.2), learningrate = NULL, lifesign = "none",
  lifesign.step = 1000, algorithm = "rprop+", err.fct = "sse",
  act.fct = "logistic", linear.output = TRUE, exclude = NULL,
  constant.weights = NULL, likelihood = FALSE)
```

範例: 利用"neuralnet "構建一個深度神經網路來逼近一個函數 y=x^2

```
test.data <- as.matrix(sample(seq(-2, 2, length.out=10), 10, replace=F), ncol=1)
head(test.data)
pred <- compute(fit, test.data)

data.frame(Test.attribute=test.data, Prediction=pred$net.result,
           Actual.response=f(test.data))

dev.set(dev.prev())
points(test.data, pred$net.result, col="red", cex=2, pch=3)
```

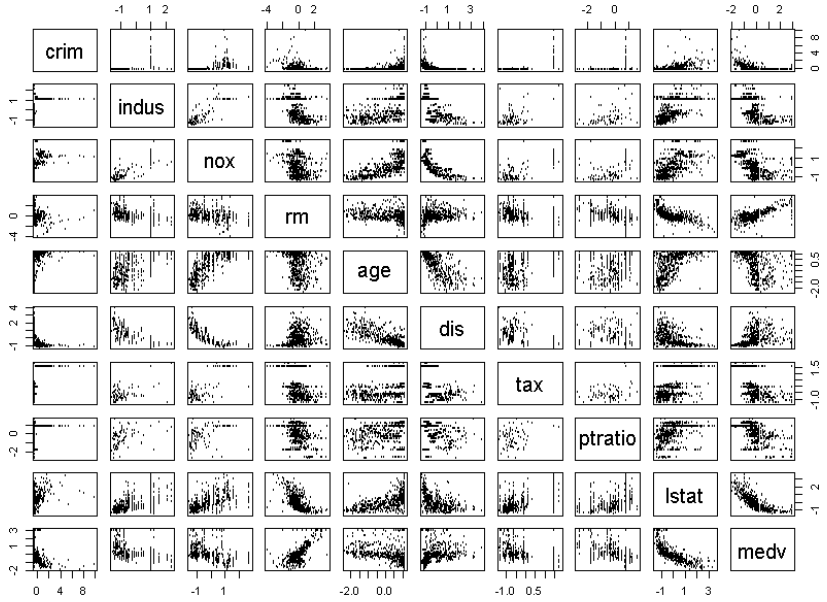


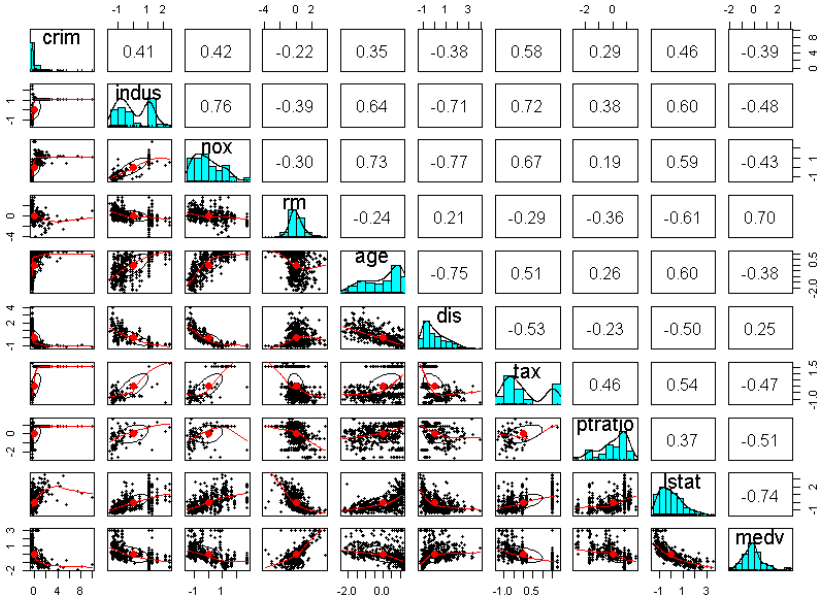


範例: BostonHousing
預測波士頓郊區的房價 ("lm", "neuralnet", "deepnet")

```
data(Boston, package="MASS")
head(Boston)
str(Boston)
names(Boston)
used.variables <- c("crim", "indus", "nox", "rm", "age",
                    "dis", "tax", "ptratio", "lstat", "medv")
Boston.used <- as.data.frame(scale(Boston[used.variables]))
str(Boston.used)

apply(Boston.used, 2, function(x) sum(is.na(x)))
pairs(Boston.used, cex=0.4)

library(psych)
pairs.panels(Boston.used)
```

```
# data(BostonHousing, package="mlbench")
# head(BostonHousing)
# str(BostonHousing)
```





範例: BostonHousing
預測波士頓郊區的房價 ("lm", "neuralnet", "deepnet")

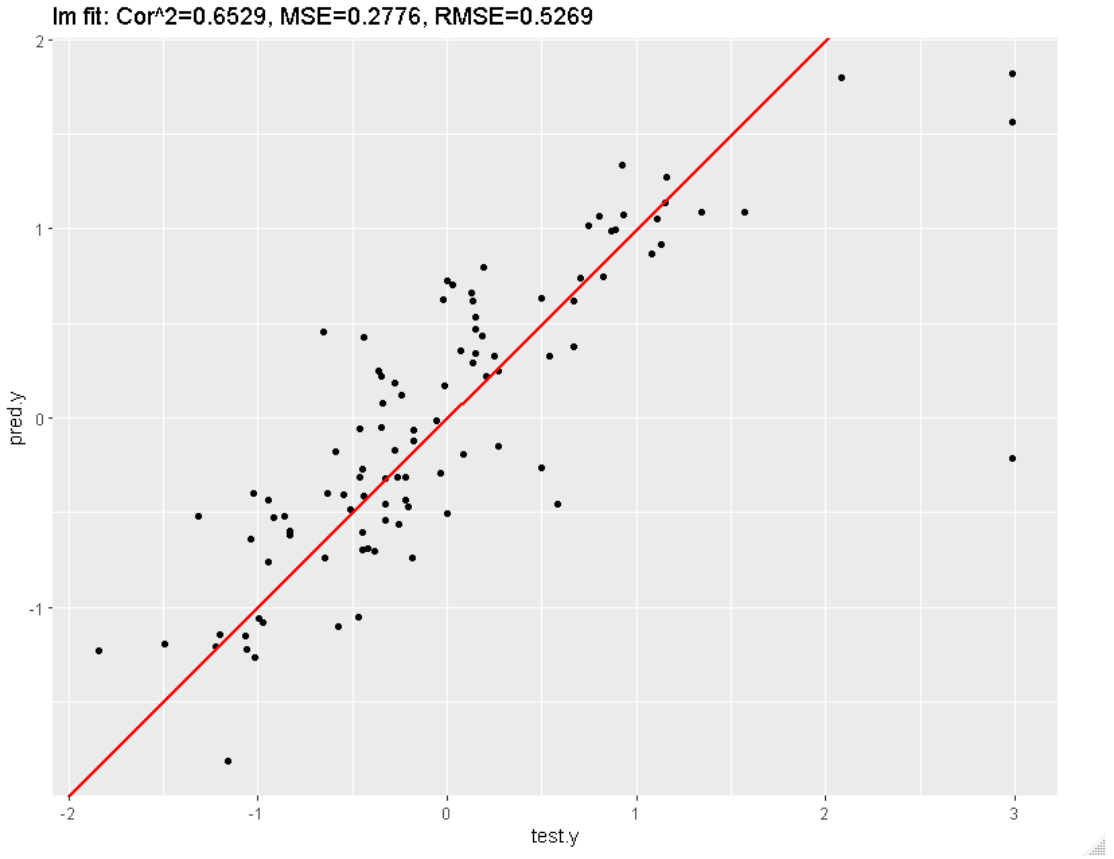
```
library(caret)
set.seed(12345)
id <- createDataPartition(y=1:nrow(Boston.used), p=0.8)
train.data <- Boston.used[id$Resample1, ]
head(train.data)
train.X <- train.data[, -10]
train.y <- train.data[, 10]
test.data <- Boston.used[-id$Resample1, ]
head(test.data)
test.X <- test.data[, -10]
test.y <- test.data[, 10]

validation.index <- function(pred.value, real.value){
  require(Metrics) # 計算各種模型的擬合標準
  cor.v <- cor(pred.value, real.value)^2
  mse.v <- mse(pred.value, real.value)
  rmse.v <- rmse(pred.value, real.value)
  index <- paste0("Cor^2=", round(cor.v, 4),
                  ", MSE=", round(mse.v, 4),
                  ", RMSE=", round(rmse.v, 4))
  index
}
```

BostonHousing:
利用 "lm"構建一個多重回歸模型預測波士頓郊區的房價

```
> fit.lm <- lm(formula=medv ~ ., data=train.data)
> summary(fit.lm)

Call:
lm(formula = medv ~ ., data = train.data)

Residuals:
     Min       1Q   Median       3Q      Max
-1.42038 -0.33885 -0.08678  0.22445  3.09712

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.003829   0.027202   0.141 0.888124
crim        -0.048178   0.042629  -1.130 0.259085
indus       -0.040613   0.052182  -0.778 0.436860
nox         -0.210235   0.054846  -3.833 0.000147 ***
rm           0.346899   0.036120   9.604  < 2e-16 ***
age         -0.034690   0.047231  -0.734 0.463092
dis         -0.295494   0.048465  -6.097 2.57e-09 ***
tax          0.012265   0.049680   0.247 0.805122
ptratio     -0.258568   0.033750  -7.661 1.43e-13 ***
lstat       -0.382526   0.046014  -8.313 1.51e-15 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5473 on 396 degrees of freedom
Multiple R-squared:  0.7213,  Adjusted R-squared:  0.7149
F-statistic: 113.9 on 9 and 396 DF,  p-value: < 2.2e-16
```

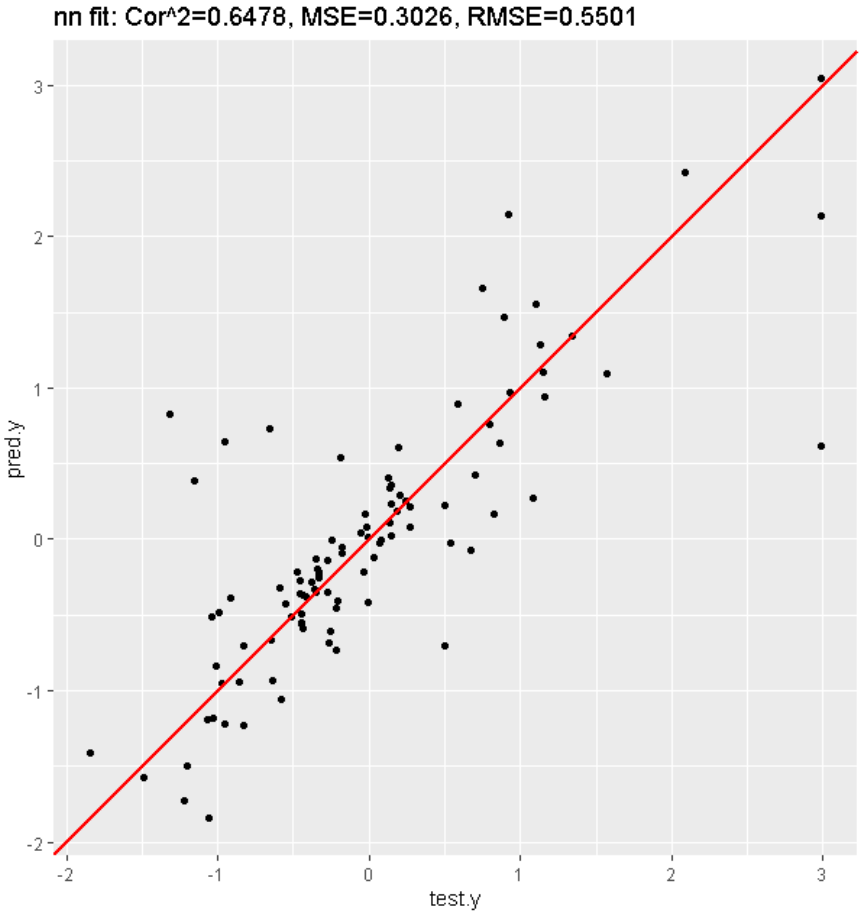
BostonHousing:
利用 "lm"構建一個多重回歸模型預測波士頓郊區的房價

```
> pred.lm <- predict(fit.lm, test.X)
> lm.pred.data <- data.frame(pred.y=pred.lm, test.y=test.y)
> ggplot(data=lm.pred.data, aes(x=test.y, y=pred.y)) +
+   geom_point() +
+   ggtitle(paste("lm fit:", validation.index(pred.lm, test.y))) +
+   geom_abline(intercept=0, slope=1, color="red", size=1) +
+   coord_fixed(ratio=1)
```

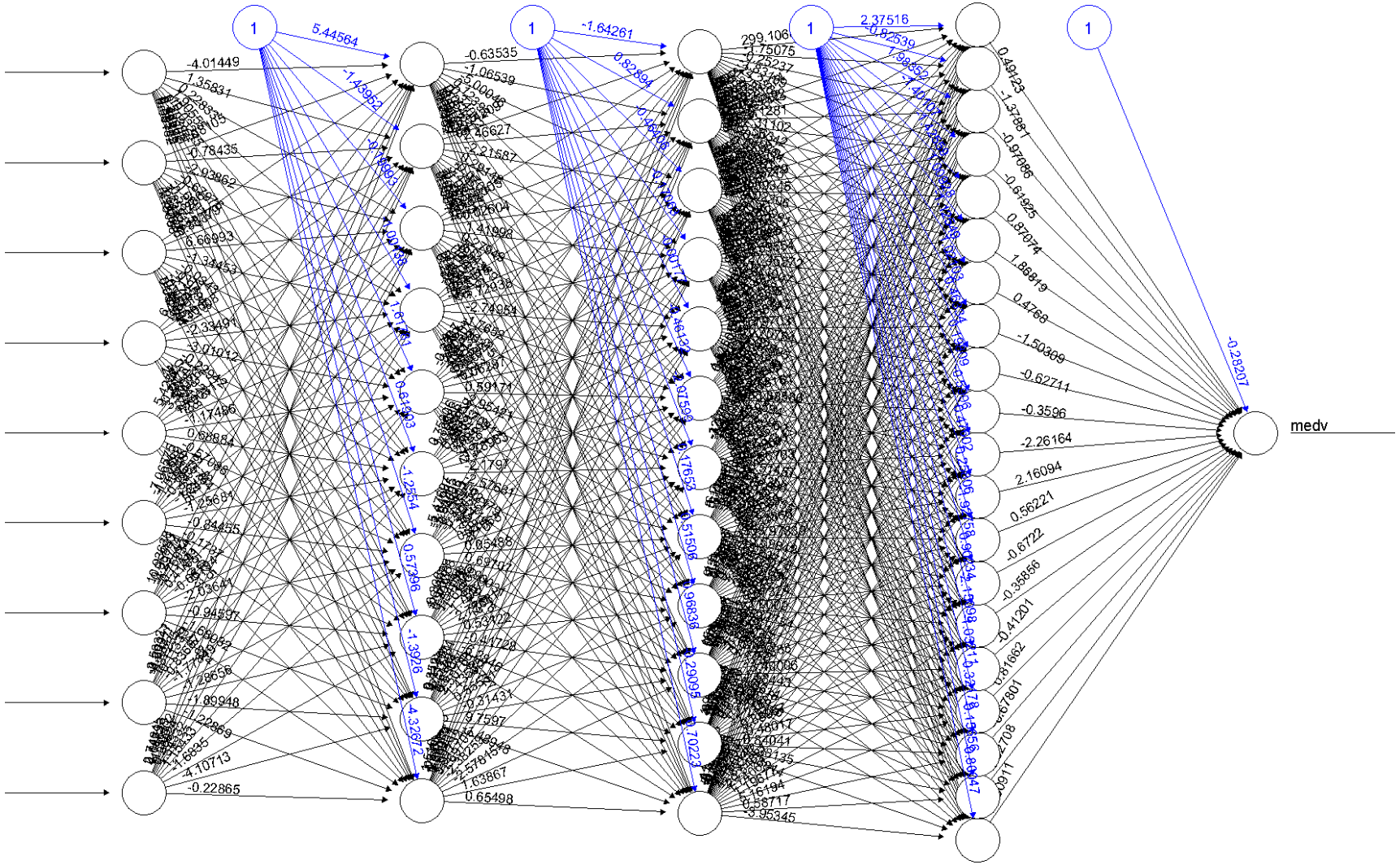


BostonHousing:
利用 "neuralnet"構建一個深度神經網路回歸模型
預測波士頓郊區的房價

```
> fit.nn <- neuralnet(formula=medv ~ .,
+                     data=train.data,
+                     hidden=c(10, 12, 20),
+                     threshold=0.1,
+                     algorithm="rprop+",
+                     err.fct="sse",
+                     act.fct="logistic",
+                     linear.output=T)
> str(fit.nn)
List of 14
...




- attr(*, "class")= chr "nn"
> summary(fit.nn)
                  Length Class      Mode
call                 9   -none-     call
...
result.matrix      516   -none-     numeric
>
> pred.nn <- compute(fit.nn,  test.X)
> nn.pred.data <- data.frame(pred.y=pred.nn$net.result, test.y=test.y)
> ggplot(data=nn.pred.data, aes(x=test.y, y=pred.y)) +
+   geom_point() +
+   ggtitle(paste("nn fit:", validation.index(pred.nn$net.result, test.y))) +
+   geom_abline(intercept=0, slope=1, color="red", size=1) +
+   coord_fixed(ratio=1)
```



plot(fit.nn)


1
5.44564
-1.43952
-1.64261
0.82894
2.37516
0.82539
1.98852
-0.28207
medv

BostonHousing:
利用"deepnet"構建一個反向傳播深度神經網路模型
預測波士頓郊區的房價

Usage

```
nn.train(x, y, initW = NULL, initB = NULL, hidden = c(10), activationfun = "sigm",
    learningrate = 0.8, momentum = 0.5, learningrate_scale = 1, output = "sigm",
    numepochs = 3, batchsize = 100, hidden_dropout = 0, visible_dropout = 0)
```

nn.train {deepnet} R Documentation

Training Neural Network

Description

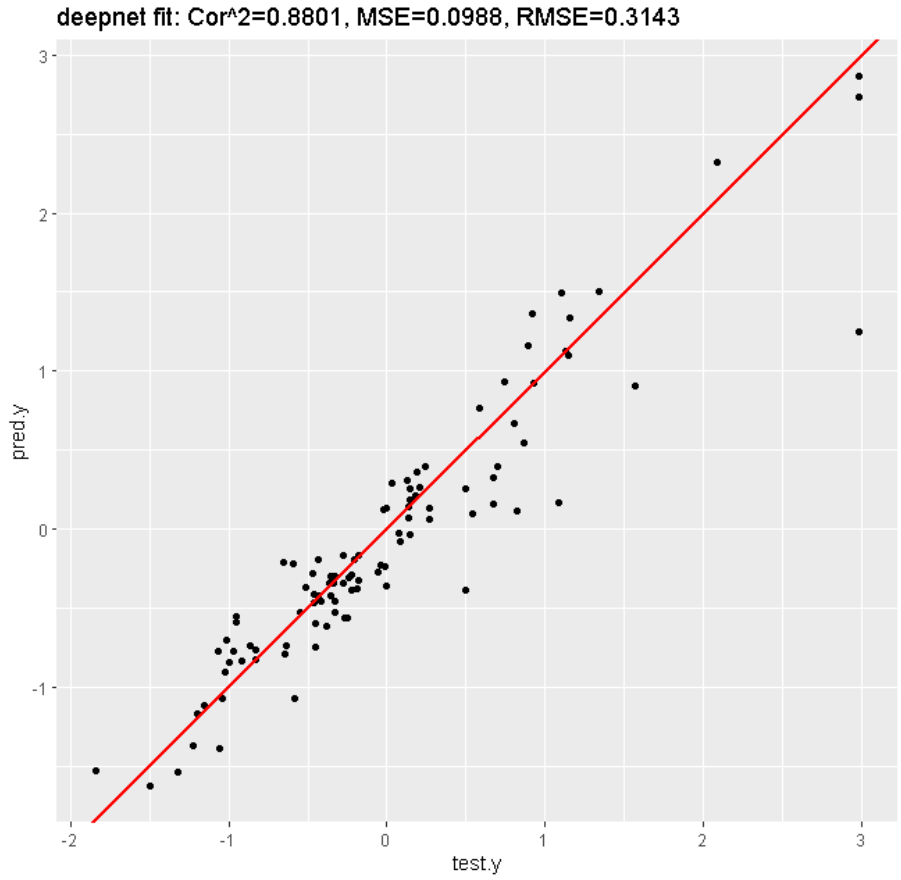
Training single or mutiple hidden layers neural network by BP

```
> require(deepnet)
> train.X <- as.matrix(train.X)
> test.X <- as.matrix(test.X)
> fit.dn <- nn.train(x=train.X, y=train.y,
+                    initW=NULL, initB=NULL,
+                    hidden=c(10, 12, 20),
+                    learningrate=0.58,
+                    momentum=0.74,
+                    learningrate_scale=1,
+                    activationfun="sigm",  #linear, tanh
+                    output="linear", #sigm, softmax
+                    numepochs=970,
+                    batchsize=60,
+                    hidden_dropout=0,
+                    visible_dropout=0)
>
> pred.dn <- nn.predict(fit.dn, test.X)
> dn.pred.data <- data.frame(pred.y=pred.dn, test.y=test.y)
```

dbn.dnn.train: Training a Deep NN with weights initialized by DBN
nn.train: Training NN by BP
rbm.train: Training a RBM (restricted Boltzmann Machine)
sae.dnn.train: Training a Deep NN with weights initialized by Stacked AutoEncoder

BostonHousing:
利用"deepnet"構建一個反向傳播深度神經網路模型
預測波士頓郊區的房價

```
> ggplot(data=dn.pred.data, aes(x=test.y, y=pred.y)) +
+   geom_point() +
+   ggtitle(paste("deepnet fit:", validation.index(pred.dn, test.y))) +
+   geom_abline(intercept=0, slope=1, color="red", size=1) +
+   coord_fixed(ratio=1)
```



範例: PimaIndiansDiabetes2:
分類問題: 預測糖尿病檢測陽陰性

```
> data("PimaIndiansDiabetes2", package="mlbench")
> # missing values
> sapply(PimaIndiansDiabetes2, function(x) sum(is.na(x)))
pregnant  glucose pressure  triceps  insulin     mass pedigree      age diabetes
       0        5       35      227      374       11        0        0        0
> pima <- PimaIndiansDiabetes2
> dim(pima)
[1] 768   9
> pima$insulin <- NULL
> pima$triceps <- NULL
> dim(pima)
[1] 768   7
> pima <- na.omit(pima)
> dim(pima)
[1] 724   7
> names(pima)
[1] "pregnant" "glucose"  "pressure" "mass"     "pedigree" "age"      "diabetes"
> pima$diabetes
[1] pos neg pos neg pos neg pos pos neg pos neg pos pos pos pos neg pos neg neg
 ...
Levels: neg pos
> pima <- data.frame(scale(pima[,-7]), pima[,7])
>
> set.seed(12345)
> train.id <- sample(1:nrow(pima), 600)
> train.X <- pima[train.id, -7]
> train.y <- as.integer(pima[train.id, 7])
> test.X <- pima[-train.id, -7]
> test.y <- as.integer(pima[-train.id, 7])
```

PimaIndiansDiabetes2: 利用"RSNNS"構建一個深度神經網路分類器模型

mlp {RSNNS} R Documentation

Create and train a multi-layer perceptron (MLP)

Description

This function creates a multilayer perceptron (MLP) and trains it. MLPs are fully connected feedforward networks, and probably the most common network architecture in use. Training is usually performed by error backpropagation or a related procedure.

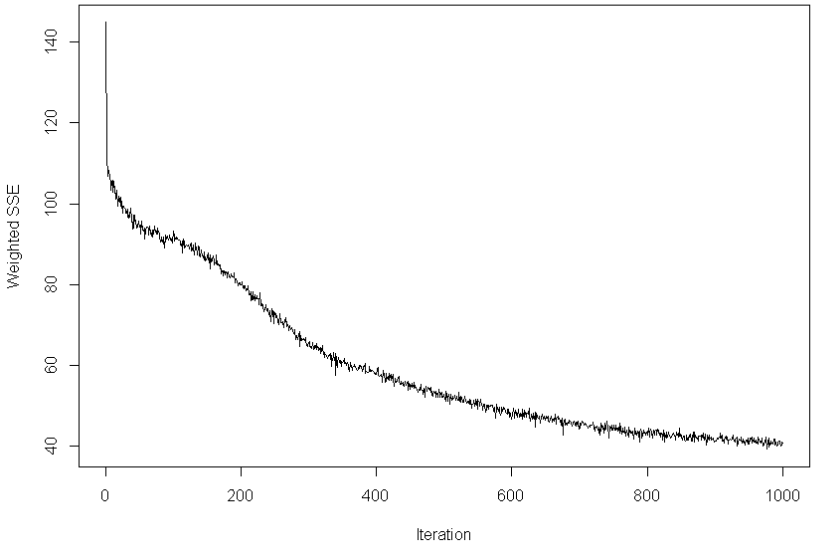
There are a lot of different learning functions present in SNNS that can be used together with this function, e.g., `Std_Backpropagation`, `BackpropBatch`, `BackpropChunk`, `BackpropMomentum`, `BackpropWeightDecay`, `Rprop`, `Quickprop`, `SCG` (scaled conjugate gradient), ...

```
mlp(x, y, size = c(5), maxit = 100,
  initFunc = "Randomize_Weights", initFuncParams = c(-0.3, 0.3),
  learnFunc = "Std_Backpropagation", learnFuncParams = c(0.2, 0),
  updateFunc = "Topological_Order", updateFuncParams = c(0),
  hiddenActFunc = "Act_Logistic", shufflePatterns = TRUE,
  linOut = FALSE, outputActFunc = if (linOut) "Act_Identity" else
  "Act_Logistic", inputsTest = NULL, targetsTest = NULL,
  pruneFunc = NULL, pruneFuncParams = NULL, ...)
```

PimaIndiansDiabetes2:
利用"RSNNS"構建一個深度神經網路分類器模型

```
> require(RSNNS)
> fit.mlp <- mlp(x=train.X, y=train.y,
+                size=c(12, 8),
+                maxit=1000,
+                initFun="Randomiza_Weights",
+                initFuncParams=c(-0.3, 0.3),
+                learnFunc="Std_Backpropagation",
+                learnFuncParams=c(0.2, 0),
+                updateFunc="Topological_Order",
+                updateFuncParams=c(0),
+                hiddenActFunc="Act_Logistic",
+                shufflePatterns=T,
+                linOut=T)
>
> pred.mlp <- ifelse(predict(fit.mlp, test.X) >= 1.5, 2, 1)
```

```
summary(fit.mlp)
fit.mlp
weightMatrix(fit.mlp)
extractNetInfo(fit.mlp)
plotIterativeError(fit.mlp)
```



PimaIndiansDiabetes2: 利用"RSNNS"構建一個深度神經網路分類器模型

```
> library(caret)
> caret::confusionMatrix(data=as.factor(pred.mlp),
+                        reference=as.factor(test.y))
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 70 12
         2 17 25
 Accuracy : 0.7661
                 95% CI : (0.6817, 0.8374)
    No Information Rate : 0.7016
    P-Value [Acc > NIR] : 0.06812

                  Kappa : 0.4623

 Mcnemar's Test P-Value : 0.45761

            Sensitivity : 0.8046
            Specificity : 0.6757
         Pos Pred Value : 0.8537
         Neg Pred Value : 0.5952
             Prevalence : 0.7016
         Detection Rate : 0.5645
   Detection Prevalence : 0.6613
      Balanced Accuracy : 0.7401

       'Positive' Class : 1
```

PimaIndiansDiabetes2: 利用"AMORE"構建一個深度神經網路分類器模型

newff {AMORE} R Documentation

Create a Multilayer Feedforward Neural Network

Description

Creates a feedforward artificial neural network according to the structure established by the AMORE package standard.

Usage

```
newff(n.neurons, learning.rate.global, momentum.global, error.criterium, Stao,
        hidden.layer, output.layer, method)
```

train {AMORE} R Documentation

Neural network training function.

Description

For a given data set (training set), this function modifies the neural network weights and biases to approximate the relationships amongst variables present in the training set. These may serve to satisfy several needs, i.e. fitting non-linear functions.

Usage

```
train(net, P, T, Pval=NULL, Tval=NULL, error.criterium="LMS", report=TRUE,
 n.shows, show.step, Stao=NA,prob=NULL,n.threads=0L)
```

PimaIndiansDiabetes2: 利用"AMORE"構建一個深度神經網路分類器模型

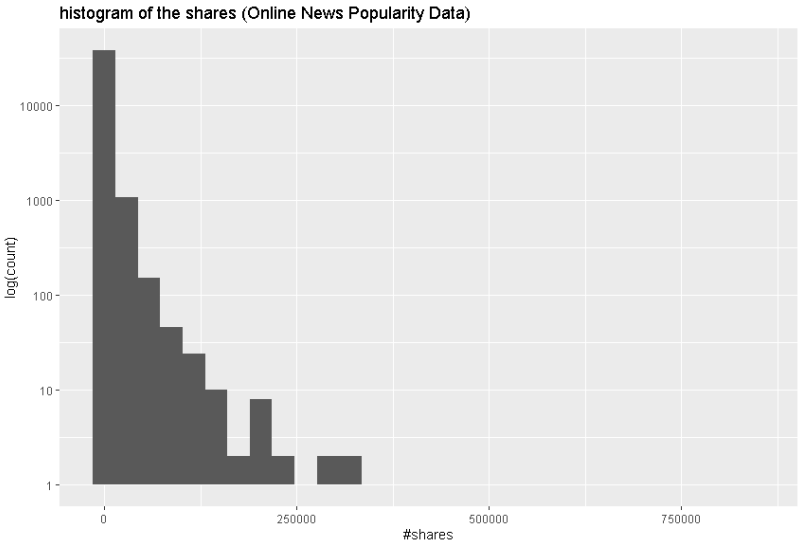
```
> amore.net <- newff(n.neurons=c(6, 12, 8, 1),
+                    learning.rate.global=0.01,
+                    momentum.global=0.5,
+                    error.criterium="LMLS",
+                    Stao=NA,
+                    hidden.layer="sigmoid",
+                    output.layer="purelin",
+                    method="ADAPTgdwm")
>
> fit.amore <- AMORE::train(net=amore.net,
+                    P=as.matrix(train.X), T=as.numeric(train.y),
+                    error.criterium="LMLS",
+                    report=T,
+                    n.shows=5,
+                    show.step=100)
index.show: 1 LMLS 0.0760980749807378
...
index.show: 5 LMLS 0.0729652957354979
>
> pred.amore <- ifelse(sim(fit.amore$net, as.matrix(test.X)) >= 1.5, 2, 1)
```

PimaIndiansDiabetes2:
利用"AMORE"構建一個深度神經網路分類器模型

```
> caret::confusionMatrix(data=as.factor(pred.amore),
+                        reference=as.factor(test.y))
Confusion Matrix and Statistics

          Reference
Prediction  1  2
         1 81 17
         2  6 20

               Accuracy : 0.8145
                 95% CI : (0.7348, 0.8786)
    No Information Rate : 0.7016
    P-Value [Acc > NIR] : 0.002963

                  Kappa : 0.5156

 Mcnemar's Test P-Value : 0.037056

            Sensitivity : 0.9310
            Specificity : 0.5405
         Pos Pred Value : 0.8265
         Neg Pred Value : 0.7692
             Prevalence : 0.7016
         Detection Rate : 0.6532
   Detection Prevalence : 0.7903
      Balanced Accuracy : 0.7358

       'Positive' Class : 1
```

範例: 利用"deepnet"構建一個深度神經網路進行在線熱點新聞分類預測

```
> data.zip <- "OnlineNewsPopularity.zip"
> uci <- "https://archive.ics.uci.edu/ml/machine-learning-databases/00332/"
> url.loc <- paste0(uci, data.zip)
> download.file(url.loc, destfile=data.zip, method="libcurl")
> unzip(data.zip, exdir="data")
> dir()
...
>
> file.loc <- "data/OnlineNewsPopularity/OnlineNewsPopularity.csv"
> ONP.data <- read.csv(file.loc)
> head(ONP.data)
...
> str(ONP.data)
'data.frame':       39644 obs. of  61 variables:
 $ url                          : chr  "http://mashabl
 $ timedelta                    : num  731 731 731 731
...
 $ shares                       : int  593 711 1500 12
>
> summary(ONP.data$shares)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1     946    1400    3395    2800  843300
>
>
> ggplot(data=ONP.data, aes(x=shares)) + geom_histogram()
> ggplot(data=ONP.data, aes(x=shares)) + geom_histogram() +
+   scale_y_log10() +
+   labs(x="#shares", y="log(count)",
+        title="histogram of the shares (Online News Popularity Data)")
>
```



範例: 利用"deepnet"構建一個深度神經網路 進行在線熱點新聞分類預測

```
> response <- as.numeric(ONP.data$shares > median(ONP.data$shares))
> scale_01 <- function(x){
+   (x-min(x))/(max(x)-min(x))
+ }
> names(ONP.data)
 [1] "url"                          "timedelta"                    "n_tokens_title"
 [4] "n_tokens_content"             "n_unique_tokens"              "n_non_stop_words"
 [7] "n_non_stop_unique_tokens"     "num_hrefs"                    "num_self_hrefs"
[10] "num_imgs"                     "num_videos"                   "average_token_length"
[13] "num_keywords"                 "data_channel_is_lifestyle"    "data_channel_is_entertainment"
[16] "data_channel_is_bus"          "data_channel_is_socmed"       "data_channel_is_tech"
[19] "data_channel_is_world"        "kw_min_min"                   "kw_max_min"
[22] "kw_avg_min"                   "kw_min_max"                   "kw_max_max"
[25] "kw_avg_max"                   "kw_min_avg"                   "kw_max_avg"
[28] "kw_avg_avg"                   "self_reference_min_shares"    "self_reference_max_shares"
[31] "self_reference_avg_sharess"   "weekday_is_monday"            "weekday_is_tuesday"
[34] "weekday_is_wednesday"         "weekday_is_thursday"          "weekday_is_friday"
[37] "weekday_is_saturday"          "weekday_is_sunday"            "is_weekend"
[40] "LDA_00"                       "LDA_01"                       "LDA_02"
[43] "LDA_03"                       "LDA_04"                       "global_subjectivity"
[46] "global_sentiment_polarity"    "global_rate_positive_words"   "global_rate_negative_words"
[49] "rate_positive_words"          "rate_negative_words"          "avg_positive_polarity"
[52] "min_positive_polarity"        "max_positive_polarity"        "avg_negative_polarity"
[55] "min_negative_polarity"        "max_negative_polarity"        "title_subjectivity"
[58] "title_sentiment_polarity"     "abs_title_subjectivity"       "abs_title_sentiment_polarity"
[61] "shares"
> channel <- ONP.data[, 14:19] #binary
> kword <- apply(ONP.data[, 20:28], 2, scale_01)
> day <- ONP.data[, 32:39] #binary
> lda <- ONP.data[, 40:44] #binary
> ONP.used <- cbind(channel, kword, day, lda, response)
```

Latent Dirichlet allocation

範例: 利用"deepnet"構建一個深度神經網路 進行在線熱點新聞分類預測

```
> library(caret)
> set.seed(12345)
> id <- createDataPartition(y=response, p=0.8)
> response.at <- which(names(train.data) %in% c("response"))
>
> train.data <- ONP.used[id$Resample1, ]
> head(train.data)
  data_channel_is_lifestyle data_channel_is_entertainment data_channel_is_bus
1                         0                              1                   0
...
> train.X <- train.data[, -response.at]
> train.y <- train.data[, response.at]
> dim(train.X)
[1] 31716    28
>
> test.data <- ONP.used[-id$Resample1, ]
> head(test.data)
   data_channel_is_lifestyle data_channel_is_entertainment data_channel_is_bus
7                          1                              0                   0
...
> test.X <- test.data[, -response.at]
> test.y <- test.data[, response.at]
> dim(test.X)
[1] 7928   28
```

範例: 利用"deepnet"構建一個深度神經網路進行在線熱點新聞分類預測

```
> require(deepnet)
> set.seed(12345)
> fit.dn <- nn.train(x=as.matrix(train.X),
                     y=train.y,
+                    hidden=c(60, 30),
+                    numepochs=10,
+                    activationfun="sigm",
+                    output="sigm")
>
> attributes(fit.dn)
$names
 [1] "input_dim"           "output_dim"
 [3] "hidden"              "size"
 [5] "activationfun"       "learningrate"
 [7] "momentum"            "learningrate_scale"
 [9] "hidden_dropout"      "visible_dropout"
[11] "output"              "W"
[13] "vW"                  "B"
[15] "vB"                  "post"
[17] "pre"                 "e"
[19] "L"
>
> pred.dn <- nn.predict(fit.dn, as.matrix(test.X))
> pred.dn.class <- ifelse(pred.dn > 0.5, 1, 0)
```

```
> caret::confusionMatrix(data=as.factor(pred.dn.class),
+                        reference=as.factor(test.y))
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 2413 1446
         1 1581 2488

               Accuracy : 0.6182
                 95% CI : (0.6074, 0.6289)
    No Information Rate : 0.5038
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.2365

 Mcnemar's Test P-Value : 0.01487

            Sensitivity : 0.6042
            Specificity : 0.6324
         Pos Pred Value : 0.6253
         Neg Pred Value : 0.6115
             Prevalence : 0.5038
         Detection Rate : 0.3044
   Detection Prevalence : 0.4868
      Balanced Accuracy : 0.6183

       'Positive' Class : 0
```

卷積神經網路 (Convolutional neural network, CNN)

Source:

vision.stanford.edu › papers › Lecun98 ▾ PDF 翻譯這個網頁

Gradient-Based Learning Applied to Document Recognition

... THE IEEE, NOVEMBER 1998. 1. Gradient-Based Learning Applied to Document. Recognition.

Yann LeCun, L eon Bottou, Yoshua Bengio, and Patrick Haffner.

由 Y LeCun 著作 - 1998 - 被引用 30029 次 - 相關文章

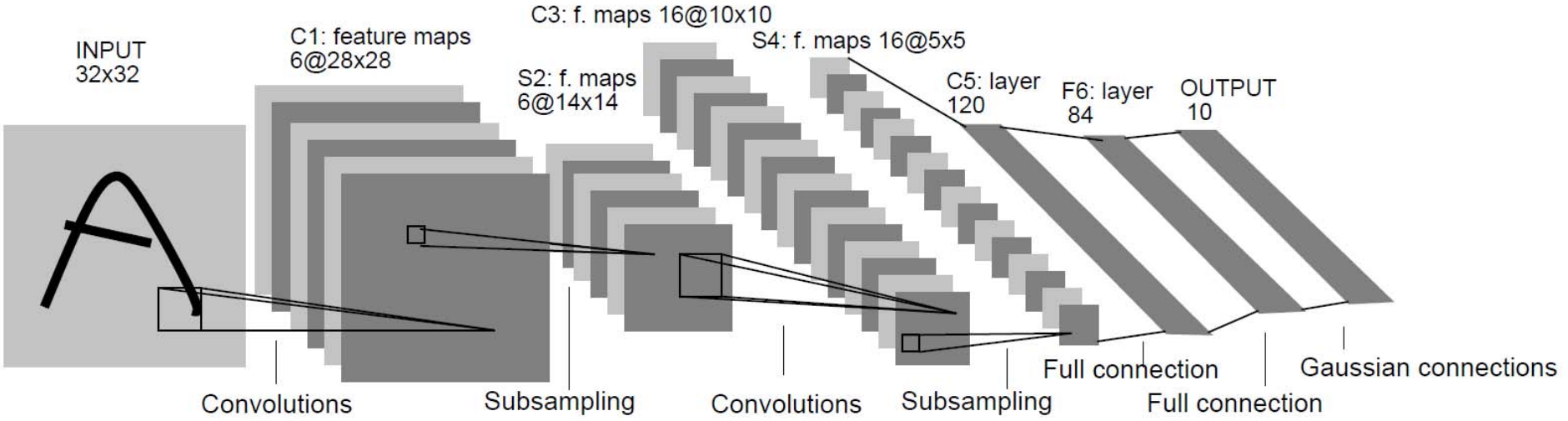
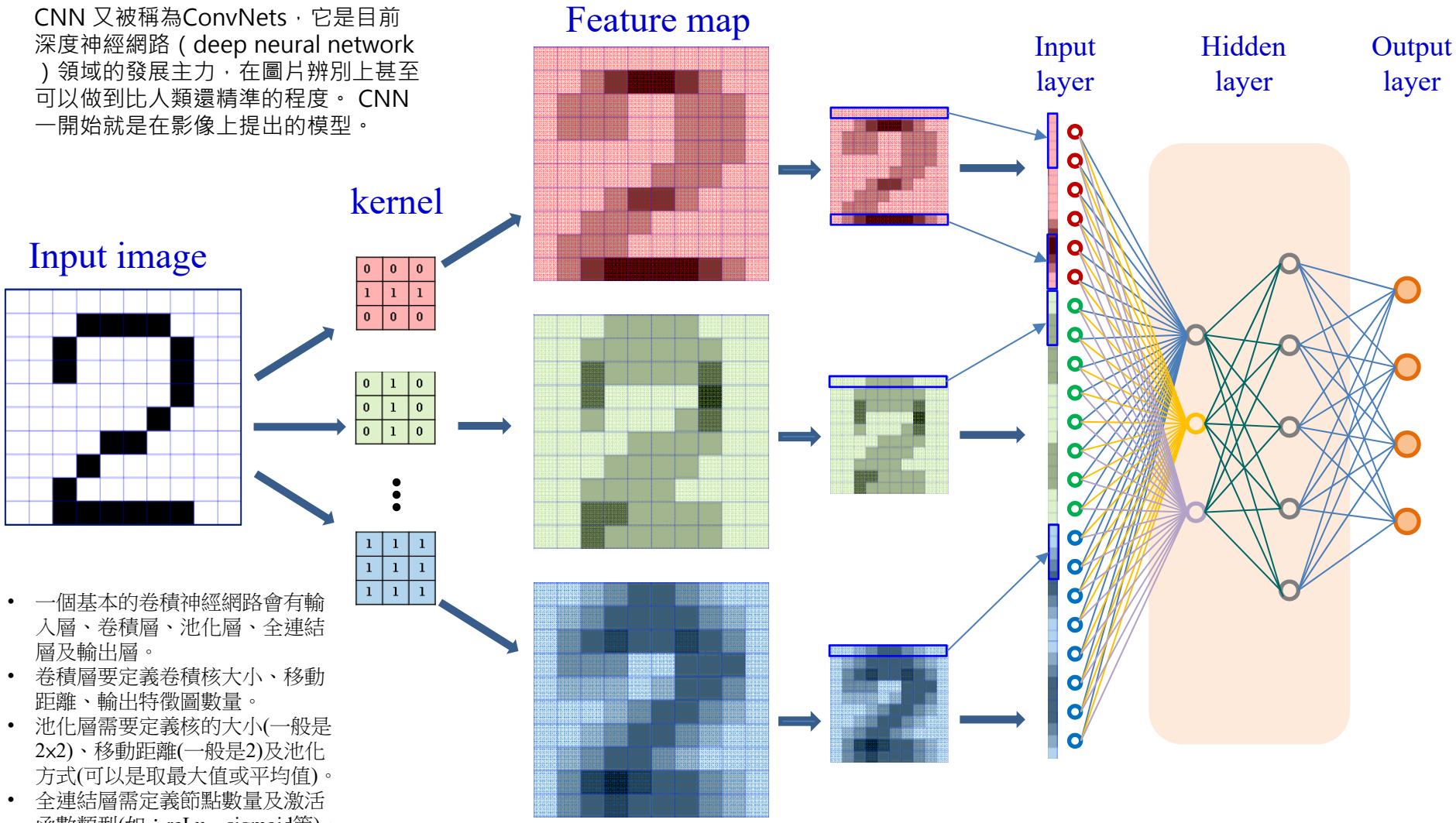


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

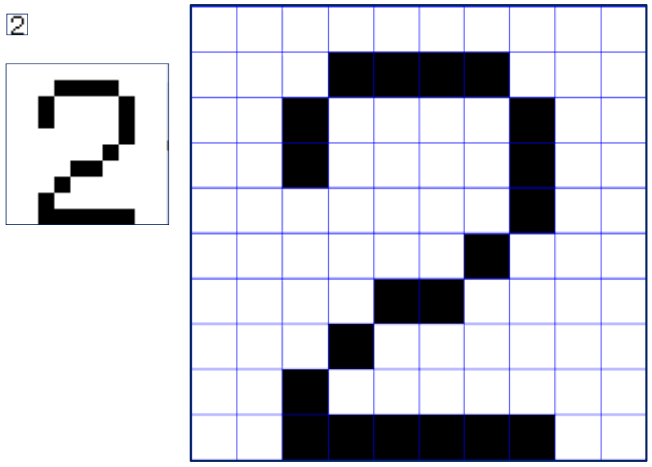
卷積神經網路 (Convolutional neural network, CNN)

CNN 又被稱為ConvNets，它是目前深度神經網路（deep neural network）領域的發展主力，在圖片辨別上甚至可以做到比人類還精準的程度。 CNN一開始就是在影像上提出的模型。



- 一個基本的卷積神經網路會有輸入層、卷積層、池化層、全連結層及輸出層。
- 卷積層要定義卷積核大小、移動距離、輸出特徵圖數量。
- 池化層需要定義核的大小(一般是2x2)、移動距離(一般是2)及池化方式(可以是取最大值或平均值)。
- 全連結層需定義節點數量及激活函數類型(如：reLu、sigmoid等)。
- 輸出層要定義正規化機率值(如：Softmax)外，還要定義損失函數以作為訓練模型用。

Convolution (卷積運算)



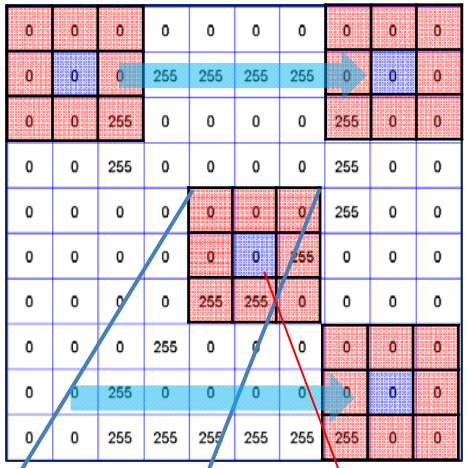
Raw image
10x10 pixels

(filter)
Convolution kernel

$$\frac{1}{3} \times \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Normalization factor

$$\otimes \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 255 \\ 255 & 255 & 0 \end{bmatrix} = 85$$

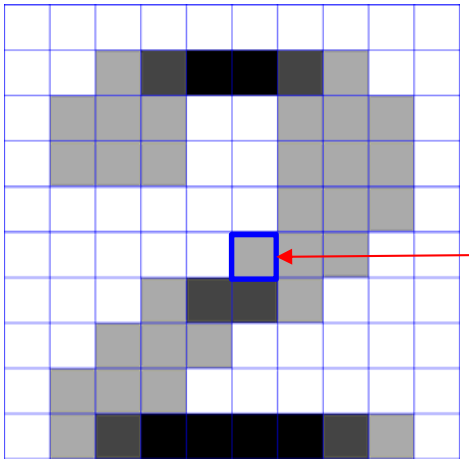


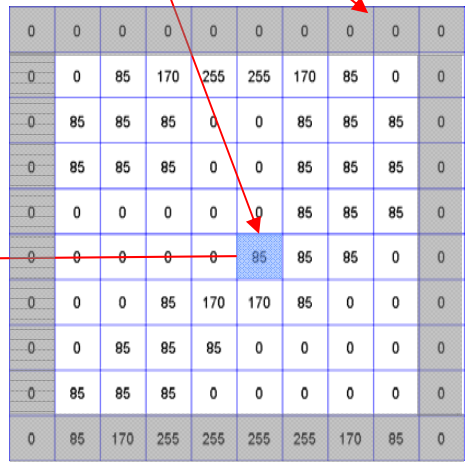
Stride:
kernel移動長度

Data matrix
(image grayscale)
8位元

`convolve {imager}`,
Boundary condition:
Dirichlet (T), Neumann (F)

Feature map





zero padding method:
input image 四周圍加0，再進行Convolution
image size=10x10

Convolution Kernels

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Kernel (image processing)

From Wikipedia, the free encyclopedia

For other uses, see Kernel (disambiguation).

In image processing, a **kernel**, **convolution matrix**, or **mask** is a small matrix. It is used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between a kernel and an image.

Contents [hide]
1 Details

Operation	Kernel ω	Image result g(x,y)
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

The general expression of a convolution is

$$g(x,y) = \omega * f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy) f(x+dx, y+dy),$$

where $g(x,y)$ is the filtered image, $f(x,y)$ is the original image, ω is the filter kernel. Every element of the filter kernel is considered by $-a \leq dx \leq a$ and $-b \leq dy \leq b$.

Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 × 3 (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Unsharp masking 5 × 5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Prewitt Operator

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

Directional Edge Detection

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal lines

-1	2	-1
-1	2	-1
-1	2	-1

Vertical lines

-1	-1	2
-1	2	-1
2	-1	-1

+45° edges

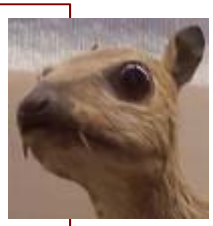
2	-1	-1
-1	2	-1
-1	-1	2

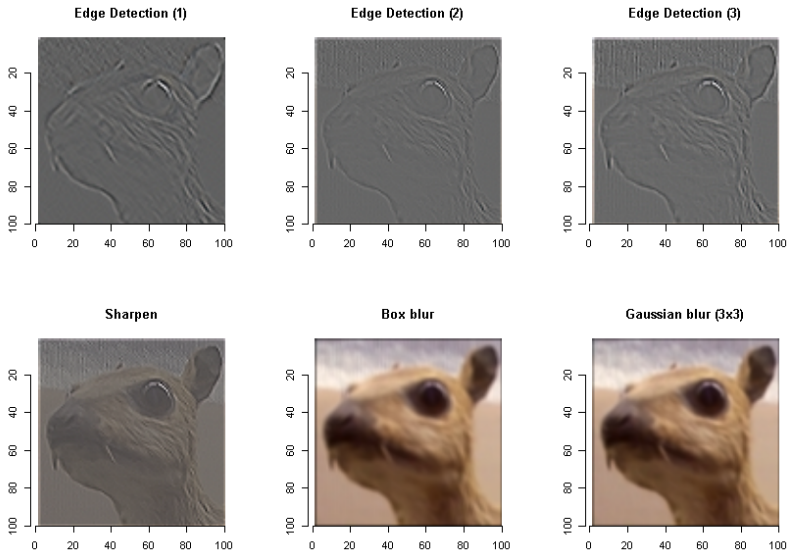
- 45° edges

https://en.wikipedia.org/wiki/Kernel_(image_processing)

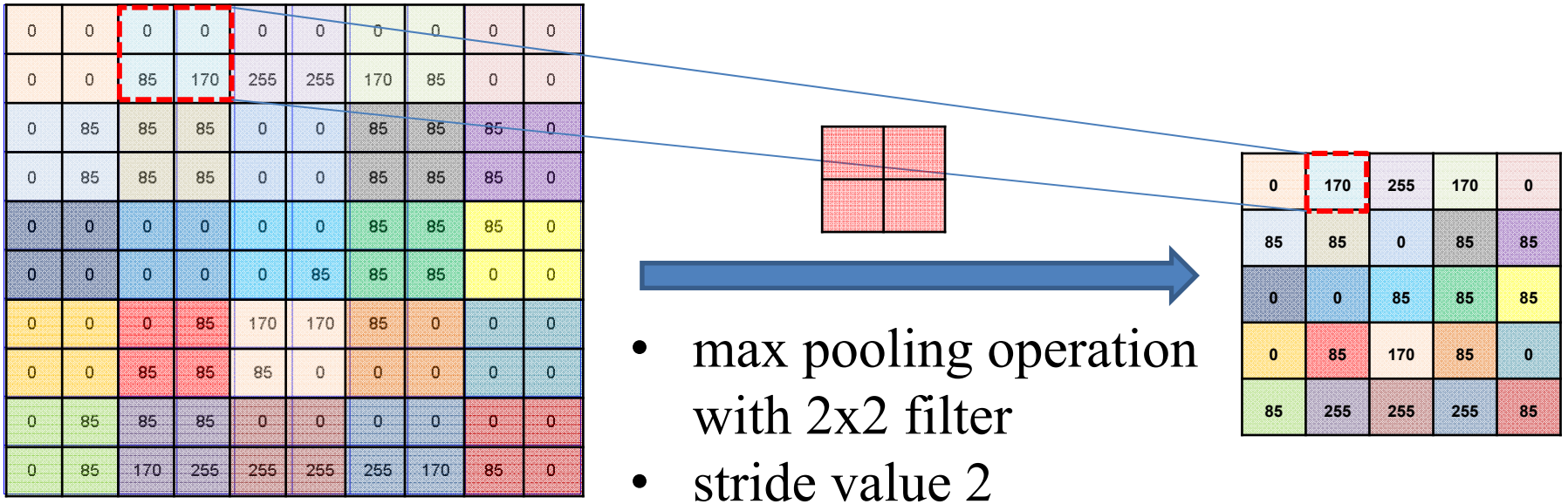
Image Convolution 練習

```
library(imager)
vd.img <- load.image("data/Vd-Orig.png")
dim(vd.img)
plot(vd.img)

Edge.detection.k1 <- as.cimg(matrix(c(1, 0, -1, 0, 0, 0, -1, 0, 1), ncol=3))
Edge.detection.k2 <- as.cimg(matrix(c(0, -1, 0, -1, 4, -1, 0, -1, 0), ncol=3))
Edge.detection.k3 <- as.cimg(matrix(c(-1, -1, -1, -1, 8, -1, -1, -1, -1), ncol=3))
Sharpen.k <- as.cimg(matrix(c(0, -1, 0, -1, 5, -1, 0, -1, 0), ncol=3))
Boxblur.k <- as.cimg(matrix(rep(1, 9), ncol=3)/9)
Gaussianblur.3x3.k <- as.cimg(matrix(c(1, 2, 1, 2, 4, 2, 1, 2, 1), ncol=3)/16)

edk1.img <- imager::convolve(vd.img, Edge.detection.k1)
edk2.img <- imager::convolve(vd.img, Edge.detection.k2)
edk3.img <- imager::convolve(vd.img, Edge.detection.k3)
sk.img <- imager::convolve(vd.img, Sharpen.k)
bk.img <- imager::convolve(vd.img, Boxblur.k)
g3k.img <- imager::convolve(vd.img, Gaussianblur.3x3.k)

par(mfrow=c(2, 3))
plot(edk1.img, main="Edge Detection (1)")
plot(edk2.img, main="Edge Detection (2)")
plot(edk3.img, main="Edge Detection (3)")
plot(sk.img, main="Sharpen")
plot(bk.img, main="Box blur")
plot(g3k.img, main="Gaussian blur (3x3)")
```





Pooling (池化運算)



- max pooling operation with 2x2 filter
- stride value 2

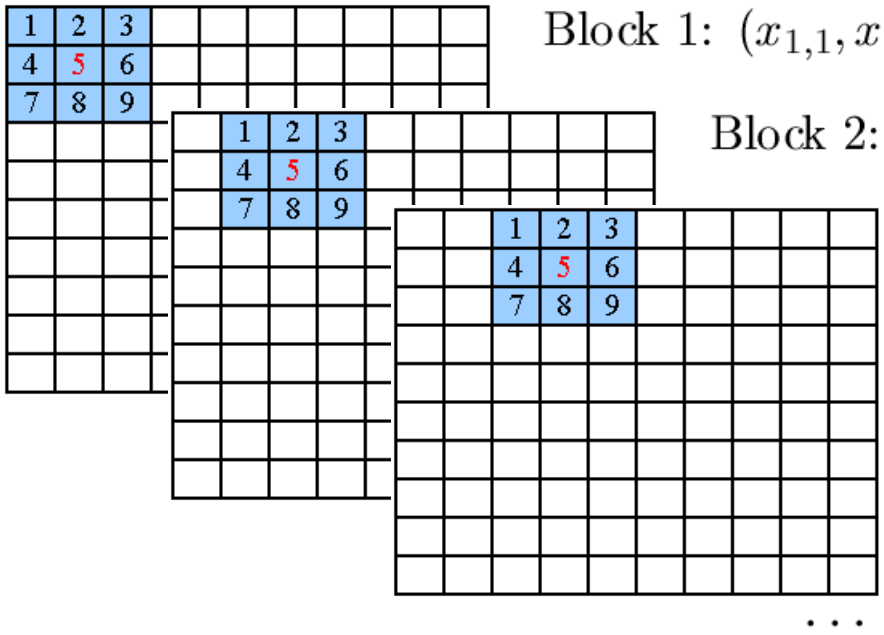
input:
10x10 image
data matrix

output:
5x5 (size-reduced)
pooled image
data matrix

Understanding Max-Pooling of Image Data with R
https://www.datatechnotes.com/2018/11/understanding-max-pooling-image-data.html

Image Feature Extraction: Local Blocks

Block length = 3



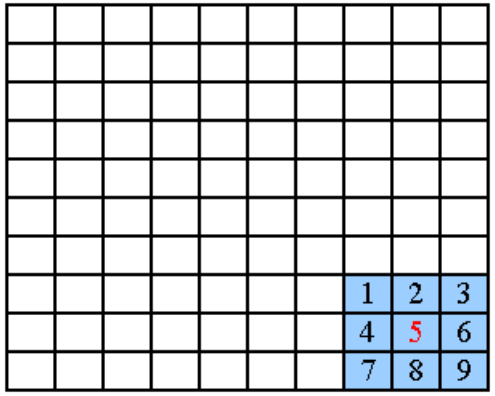
Block 1: $(x_{1,1}, x_{1,2}, \cdots, x_{1,9})$

Block 2: $(x_{2,1}, x_{2,2}, \cdots, x_{2,9})$

Block 3: $(x_{3,1}, x_{3,2}, \cdots, x_{3,9})$

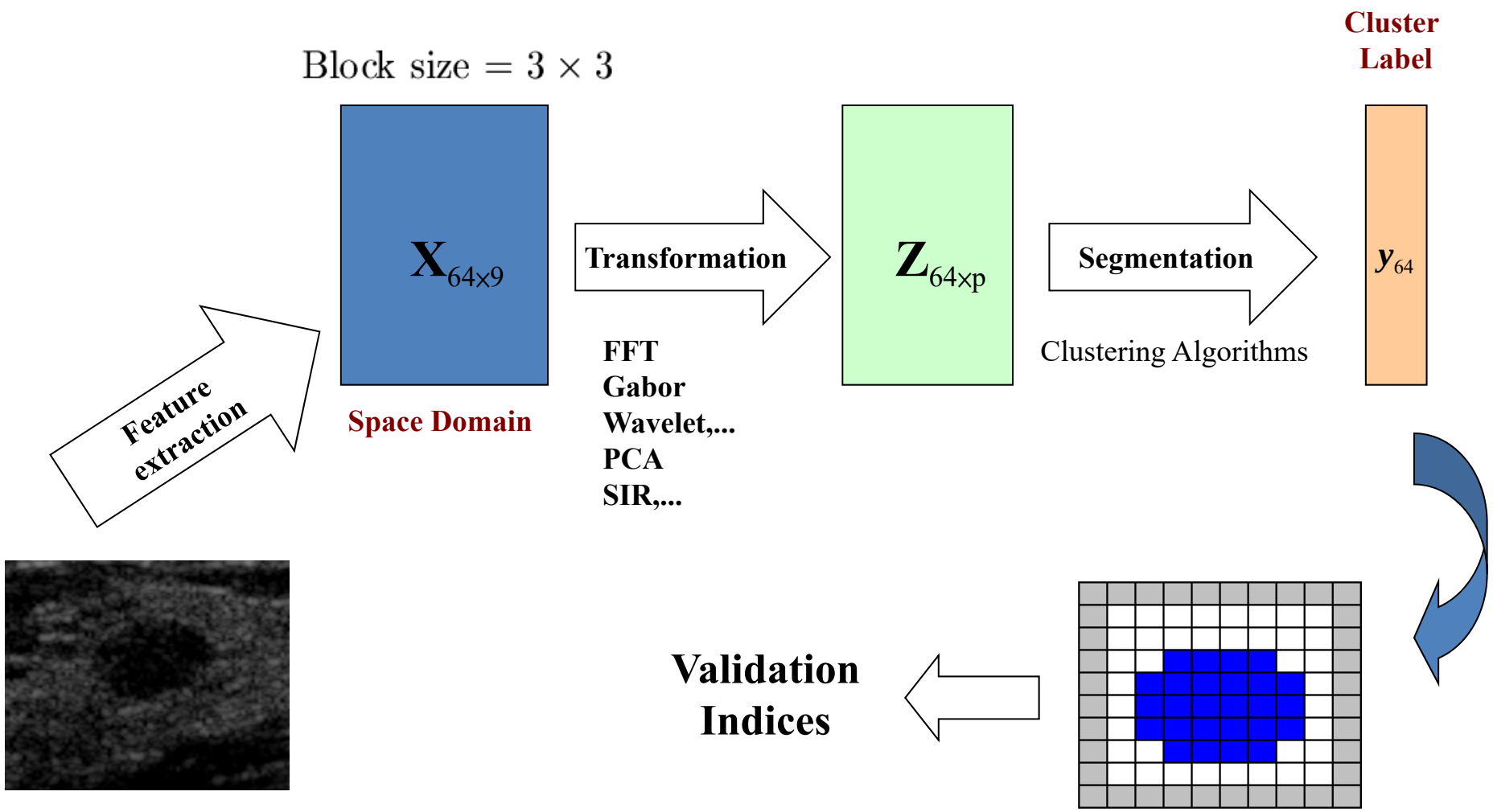
$\cdots$

Block 64: $(x_{64,1}, x_{64,2}, \cdots, x_{64,9})$



grey level: *f(x,y)=0,...,255*

Image Segmentation Procedure



其它深度學習技術

- 循環神經網路(Recurrent Neural Network, RNN): 應用在地震預測、天氣預測、文章閱讀等。
- 長短期記憶 (LSTM): 應用在機器翻譯、語音辨識、語音合成等。
- 深度前饋神經網路
- 深度捲積神經網路
- 深度堆疊自動編碼網路
- 稀疏深度神經網路
- 深度融合網路
- 深度生成網路
- 深度複捲積神經網路與深度二值神經網路
- 深度循環和遞迴神經網路
- 深度強化學習
- 生成對抗網路(Generative Adversarial Network, GA
- ...

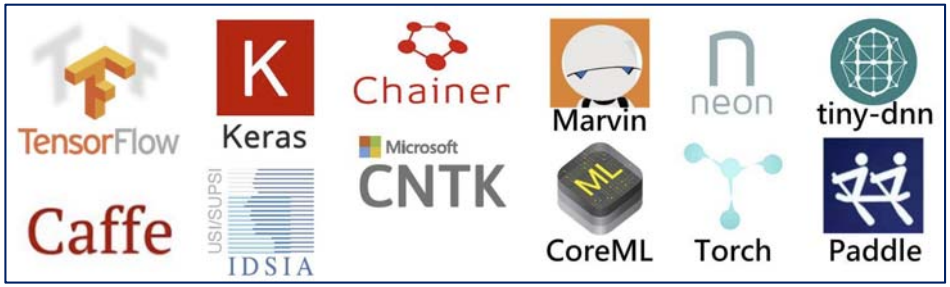
WIKIPEDIA
The Free Encyclopedia

Article Talk

Deep learning

From Wikipedia, the free encyclopedia

深度學習，從「框架」開始學起



Source: https://makerpro.cc/2018/06/deep-learning-frameworks/

Deep learning frameworks offer building blocks for designing, training and validating deep neural networks, through a high level programming interface. Widely used deep learning frameworks such as MXNet, PyTorch, TensorFlow and others rely on GPU-accelerated libraries such as cuDNN, NCCL and DALI to deliver high-performance multi-GPU accelerated training.

選擇深度框架需考慮的面向:

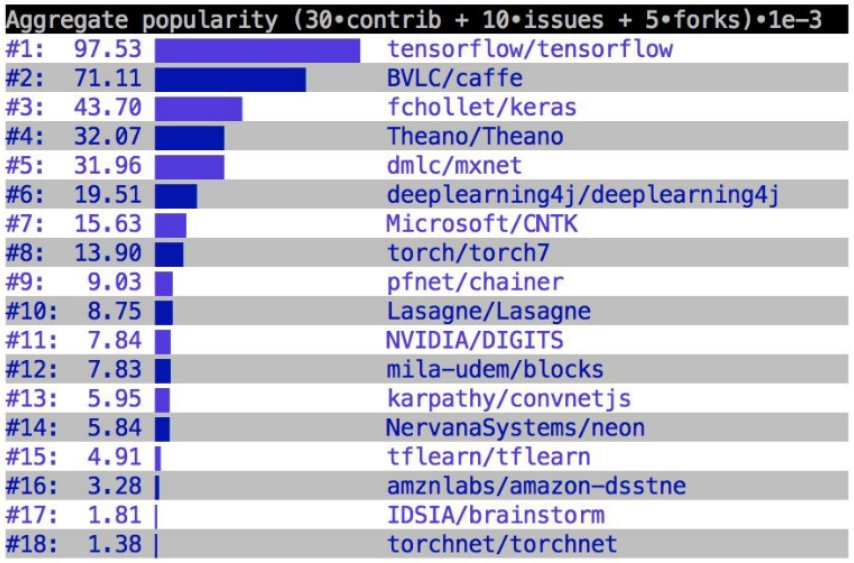
- **程式語言**: C++(效率好)，Python(上手容易、支援性強)，Java(只有TensorFlow、DL4J和MXNet)。
- **執行平台**: Linux+CPU+GPU，Windows+Intel(BigDL)+Microsoft(CNTK)，Mac(CoreML)，是否有支援叢集(Cluster)運算。Google的TensorFlow可支援Linux、Windows、Mac及Android。
- **模型支援**: 是否有支援監督型學習模型(CNN)、時序型(如RNN/LSTM)、增強學習(如Q-Learning)、轉移學習、對抗生成(GAN)等等。
- **框架轉換**: Keras可支援TensorFlow、Theano、MXNet、DL4J、CNTK。ONNX(微軟和臉書聯盟)可支援Caffe2、CNTK、PyTorch等。
- **社群支援**: 框架社群是否活躍，或很久沒有維護/升級，甚至即將淘汰。
- **最多人的選擇**: Python + Keras + TensorFlow。

深度學習，從「框架」開始學起

TensorFlow or Keras? Which one should I learn?

Aakash Nain Follow
May 13, 2017 · 5 min read

Deep learning is everywhere. 2016 was the year where we saw some huge advancements in the field of Deep Learning and 2017 is all set to see many more advanced use cases. With plenty of libraries out there for deep learning, one thing that confuses a beginner in this field the most is which library to choose.



Deep Learning libraries/frameworks as per popularity(Source : Google)

https://medium.com/implodinggradients/tensorflow-or-keras-which-one-should-i-learn-5dd7fa3f9ca0

https://geekflare.com/ai-frameworks/

AI Framework Comparision

Framework	Language	Opensource?	Features of Architecture
TensorFlow	C++ or Python	Yes	Uses data structures
Microsoft CNTK	C++	Yes	GPU/CPU based. It supports RNN, GNN, and CNN.
Caffe	C++	Yes	Its architecture supports CNN
Theano	Python	Yes	Flexible architecture allowing it to deploy in any GPU or CPU
Amazon Machine Learning	Multiple languages	Yes	Hailing from Amazon, it uses AWS.
Torch	Lua	Yes	Its architecture allows powerful computations.
Accord.Net	C#	Yes	Capable of scientific computations and pattern recognition.
Apache Mahout	Java, Scala	Yes	Capable of making machines learn without having to program
Spark MLib	R, Scala, Java, and Python	Yes	Drivers, and executors run in their processors—horizontal or vertical clusters.

Comparison of deep-learning software

Deep-learning software by name [edit]

Software	Creator	Initial Release	Software license[b]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)	Actively Developed
BigDL	Jason Dai (Intel)	2016	Apache 2.0	Yes	Apache Spark	Scala	Scala, Python			No		Yes	Yes	Yes			
Caffe	Berkeley Vision and Learning Center	2013	BSD	Yes	Linux, macOS, Windows[2]	C++	Python, MATLAB, C++	Yes	Under development[3]	Yes	Yes	Yes[4]	Yes	Yes	No	?	No[5]
Chainer	Preferred Networks	2015	BSD	Yes	Linux, macOS	Python	Python	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No[6]
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	2014	Apache 2.0	Yes	Linux, macOS, Windows, Android (Cross-platform)	C++, Java	Java, Scala, Clojure, Python (Keras), Kotlin	Yes	No[7]	Yes[8][9]	Computational Graph	Yes[10]	Yes	Yes	Yes	Yes[11]	
Dlib	Davis King	2002	Boost Software License	Yes	Cross-platform	C++	C++	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	
Flux	Mike Innes	2017	MIT license	Yes	Linux, MacOS, Windows (Cross-platform)	Julia	Julia			Yes	Yes	Yes[12]	Yes	Yes	No	Yes	Yes
Intel Data Analytics Acceleration Library	Intel	2015	Apache License 2.0	Yes	Linux, macOS, Windows on Intel CPU[13]	C++, Python, Java	C++, Python, Java[13]	Yes	No	No	Yes	No		Yes		Yes	
Intel Math Kernel Library	Intel		Proprietary	No	Linux, macOS, Windows on Intel CPU[14]		C[15]	Yes[16]	No	No	Yes	No	Yes[17]	Yes[17]		No	
Keras	François Chollet	2015	MIT license	Yes	Linux, macOS, Windows	Python	Python, R	Only if using Theano as backend	Can use Theano, Tensorflow or PlaidML as backends	Yes	Yes	Yes[18]	Yes	Yes	No[19]	Yes[20]	Yes
MATLAB + Deep Learning Toolbox	MathWorks		Proprietary	No	Linux, macOS, Windows	C, C++, Java, MATLAB	MATLAB	No	No	Train with Parallel Computing Toolbox and generate CUDA code with GPU Coder[21]	Yes[22]	Yes[23][24]	Yes[23]	Yes[23]	Yes	With Parallel Computing Toolbox[25]	Yes
Microsoft Cognitive Toolkit (CNTK)	Microsoft Research	2016	MIT license[26]	Yes	Windows, Linux[27] (macOS via Docker on roadmap)	C++	Python (Keras), C++, Command line,[28] BrainScript[29] (.NET on roadmap[30])	Yes[31]	No	Yes	Yes	Yes[32]	Yes[33]	Yes[33]	No[34]	Yes[35]	No[36]
Apache MXNet	Apache Software Foundation	2015	Apache 2.0	Yes	Linux, macOS, Windows,[37][38] AWS, Android,[39] iOS, JavaScript[40]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl, Clojure	Yes	On roadmap[41]	Yes	Yes[42]	Yes[43]	Yes	Yes	Yes	Yes[44]	Yes
Neural Designer	Artelnics		Proprietary	No	Linux, macOS, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?	
OpenNN	Artelnics	2003	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	Yes	?	?	No	No	No	?	
PlaidML	Vertex.AI,Intel	2017	AGPL	Yes	Linux, macOS, Windows	Python, C++, OpenCL	Python, C++	?	Some OpenCL ICDs are not recognized	No	Yes	Yes	Yes	Yes		Yes	Yes
PyTorch	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan (Facebook)	2016	BSD	Yes	Linux, macOS, Windows	Python, C, C++, CUDA	Python, C++, Julia	Yes	Via separately maintained package[45][46][46]	Yes	Yes	Yes	Yes	Yes		Yes	Yes
Apache SINGA	Apache Software Foundation	2015	Apache 2.0	Yes	Linux, macOS, Windows	C++	Python, C++, Java	No	Supported in V1.0	Yes	?	Yes	Yes	Yes	Yes	Yes	
TensorFlow	Google Brain	2015	Apache 2.0	Yes	Linux, macOS, Windows,[47] Android	C++, Python, CUDA	Python (Keras), C/C++, Java, Go, JavaScript, R,[48] Julia, Swift	No	On roadmap[49] but already with SYCL[50] support	Yes	Yes[51]	Yes[52]	Yes	Yes	Yes	Yes	Yes
Theano	Université de Montréal	2007	BSD	Yes	Cross-platform	Python	Python (Keras)	Yes	Under development[53]	Yes	Yes[54][55]	Through Lasagne's model zoo[56]	Yes	Yes	Yes	Yes[57]	No
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	2002	BSD	Yes	Linux, macOS, Windows,[58] Android,[59] iOS	C, Lua	Lua, LuaJIT,[60] C, utility library for C++/OpenCL[61]	Yes	Third party implementations[62][63]	Yes[64][65]	Through Twitter's Autograd[66]	Yes[67]	Yes	Yes	Yes	Yes[68]	No
Wolfram Mathematica	Wolfram Research	1988	Proprietary	No	Windows, macOS, Linux, Cloud computing	C++, Wolfram Language, CUDA	Wolfram Language	Yes	No	Yes	Yes	Yes[69]	Yes	Yes	Yes	Yes[69]	Yes
Software	Creator	Initial Release	Software license[b]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[71]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)	Actively Developed

https://en.wikipedia.org/wiki/Comparison_of_deep-learning_software

- TensorFlow 堪稱是目前最容易上手深度學習的工具之一。它最大的競爭對手是 PyTorch，由 Facebook 和 Miscrosoft 主導的機器學習框架。
- **Tensorflow vs PyTorch – Comparison, Features & Applications**
  - https://www.upgrad.com/blog/tensorflow-vs-pytorch-comparison/
- **rTorch: R Bindings to 'PyTorch'**
  - 'R' implementation and interface of the Machine Learning platform 'PyTorch' <https://pytorch.org/> developed in 'Python'.
  - Published: 2020-10-12
  - https://cran.r-project.org/web/packages/rTorch/index.html
- **Introducing Torch for R. Use Torch natively from R!**
  - https://blog.rstudio.com/2020/09/29/torch/

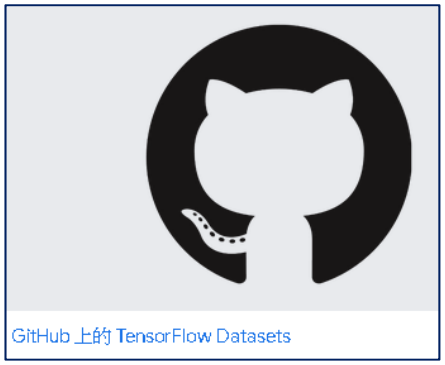
TensorFlow 官網

https://www.tensorflow.org/?hl=zh-tw

Tensorflow是一套由Google發佈的的機器學習框架，使用python及C++語言開發而成，支援多種程式語言，業界主要用於文字、語音、圖片、影片等媒體媒體格式的辨識處理。

TensorFlow 官網的教學課程

TensorFlow Datasets：一組可立即使用的資料集



https://www.tensorflow.org/datasets

30 Largest TensorFlow Datasets for Machine Learning

TensorFlow Image Datasets (影像/圖像):

- CelebA (明星臉屬性數據集)、Downsampling Imagenet(物體、場景、車輛、人物等圖像)、Lsun (場景影像，例如臥室、教室和餐廳)、Bigearthnet(衛星航空影像)、Places 365 (場景圖片，包括辦公室、碼頭和別墅)、Quickdraw(Quickdraw玩家社區繪製的圖像集合)、SVHN Cropped (街景房號)、VGGFace2 (人臉圖像)、COCO (標籤圖像，爲物體偵測、分割和圖像字幕任務而建立)、Open Images Challenge 2019 (9百萬張圖像，世界線上最大的標籤圖像資料庫)、Open Images V4、AFLW2K3D (面部圖像，均有3D面部真實標註)

TensorFlow Video Datasets (影片/動態影像/視頻):

- UCF101 (爲訓練動作識別模型而建立的視頻數據集，有101個動作類別的13320個視頻)、BAIR Robot Pushing (44000個機器人推的動作的示例視頻)、 Moving MNIST (MNIST的變體。包含10,000個視頻)、EMNIST (擴展的MNIST數據集)、

TensorFlow Audio Datasets (音頻):

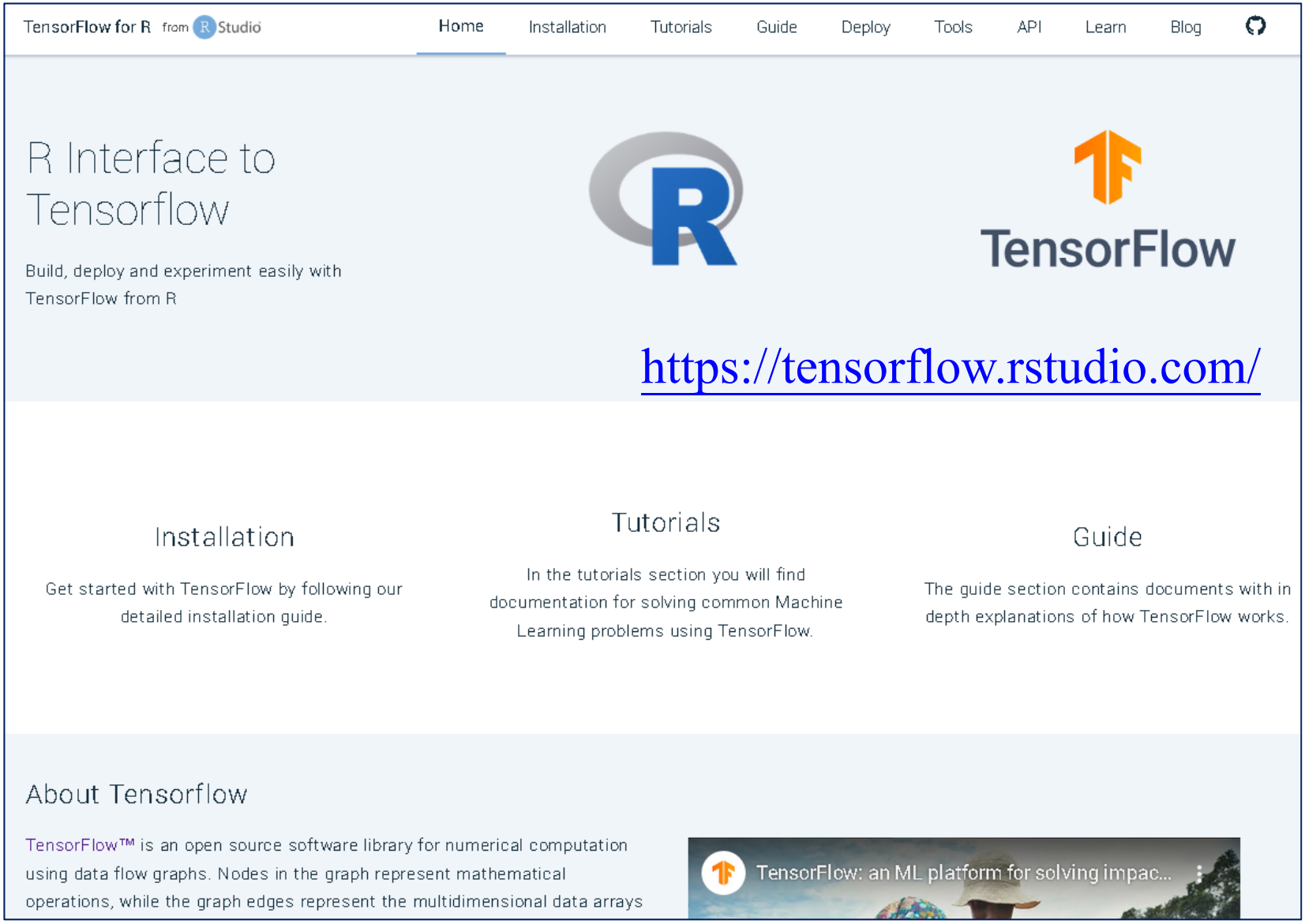
- CREMA-D (爲情感識別任務而創建，由年齡，種族和性別不同的91位演員表達的7,442個音頻剪輯)、Librispeech (1000小時的英語語音)、Libritts (585小時的英語語音，用於各種語音識別任務)、TED-LIUM (110多個小時的英語TED演講)、 VoxCeleb (1,251位演講者的150,000多個音頻樣本)、

TensorFlow Text Datasets (文本):

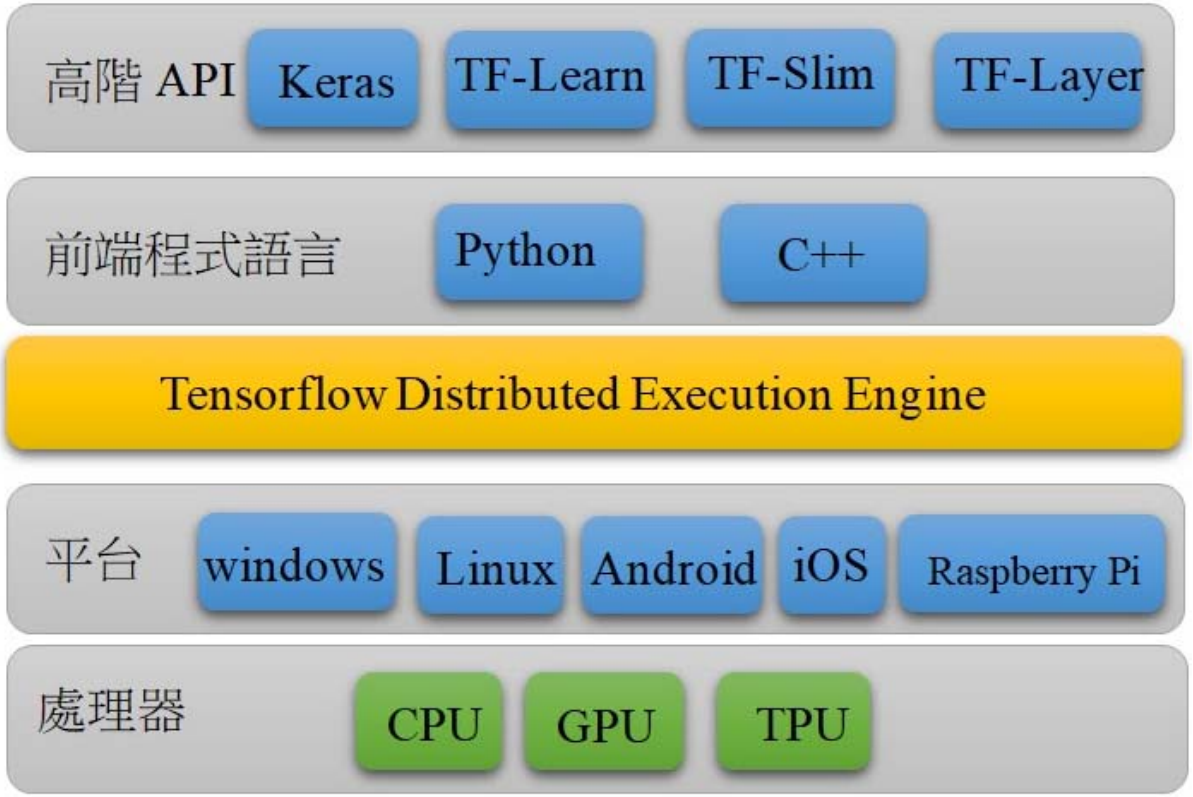
- C4 (Common Crawl's Web Crawl Corpus) (網頁數據庫, 包含了超過40種語言、跨越7年的數據)' Civil Comments (50個英文新聞網站的180多萬條公衆評論)' IRC Disentanglement (Ubuntu IRC頻道的77000多條評論)、Lm1b (語言模型基準，包含10億個單詞)、SNLI (斯坦福自然語言推理數據集)、e-SNLI (SNLI的擴展)、MultiNLI (仿照SNLI數據集，包含433,000個句子對)、Wiki40b (40種不同語言的維基百科文章)、Yelp (包含598,000條高度極性的Yelp評論)

https://lionbridge.ai/datasets/tensorflow-datasets-machine-learning/
https://bangqu.com/vHC39P.html

學習資源:
RStudio 「TensorFlow for R」 官網



TensorFlow 架構圖



瞭解 TensorFlow 和 Keras 之間的關係

TensorFlow 的高階 API，是按照 Keras API 用以定義和訓練類神經網路的標準而建立。Keras 可快速建立原型、進行最先進的研究，以及實際生產，同時還提供容易使用的 API。

閱讀 TensorFlow 2 的 Keras 指南 →

Keras

- Keras是一個開放原始碼，基於Python高階深度學習的程式庫。
- Keras可以快速有方便運算的主要原因是，它已經將訓練模型的輸入層、隱藏層、輸出層，做好架構，使用者只需要加入並且填寫正確的參數ex.神經元個數、activation function的函式...等。
- Keras與TensorFlow，CNTK和Theano不同，Keras並不是一個端到端的機器學習框架。相反，它是作為一個接口，提供高層次的抽象，讓神經網絡的配置變得簡單。目前Google TensorFlow已經支持Keras作為後端，不久之後，微軟 CNTK也會支持。缺點：不能作為獨立框架有效使用。

https://keras.rstudio.com

https://keras.io/

Keras [編輯]

維基百科，自由的百科全書

Keras是一個用Python編寫的開源神經網路庫，能夠在TensorFlow、Microsoft Cognitive Toolkit、Theano或PlaidML之上執行[1][2]。Keras旨在快速實現深度神經網路，專注於用戶友好、模組化和可延伸性，是ONEIROS（開放式神經電子智慧機器人作業系統）專案研究工作的部分產物[3]，主要作者和維護者是Google工程師弗朗索瓦·肖萊。肖萊也是XCeption深度神經網路模型的作者[4]。

2017年，Google的TensorFlow團隊決定在TensorFlow核心庫中支援Keras[5]。 Chollet解釋道，Keras被認為是一個介面，而非獨立的機器學習框架。它提供了更進階別、更直觀的抽象集，無論使用何種計算後端，用戶都可以輕鬆地開發深度學習模型[6]。微軟也向Keras添加了CNTK後端，自CNTK v2.0開始[7][8]。

Installation of TensorFlow for R: 前置作業 (設定路徑的方法)

 # 設定環境變數(Windows 8 之前)

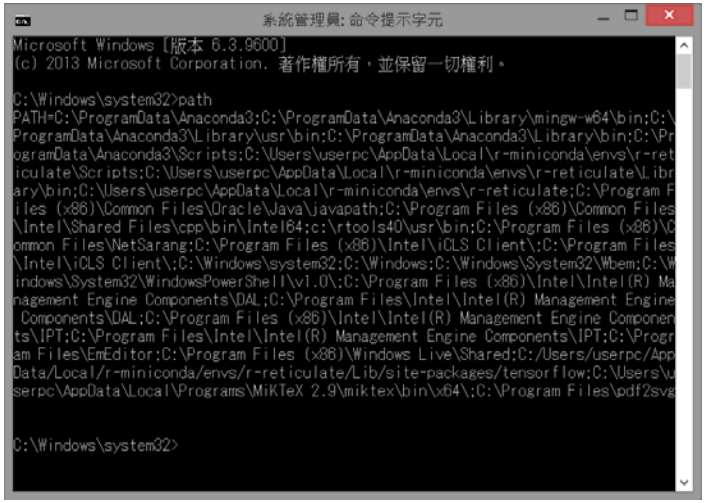
- - (Windows鍵+X) 系統(Y) => (左側) 進階系統設定
- - 進階(頁籤) => 環境變數 => 系統變數 => 找到「Path」=> 編輯 => 確定

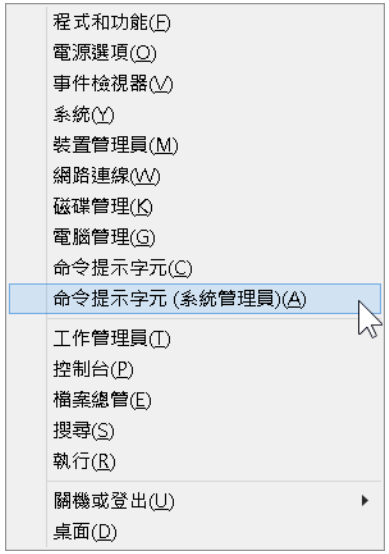
 # process to refresh environment variables without reboot windows

- - 開啟「命令提示字元」視窗。輸入「set PATH=...」
- - 關閉「命令提示字元」視窗，再重新開啟。輸入「echo %PATH%」測試。

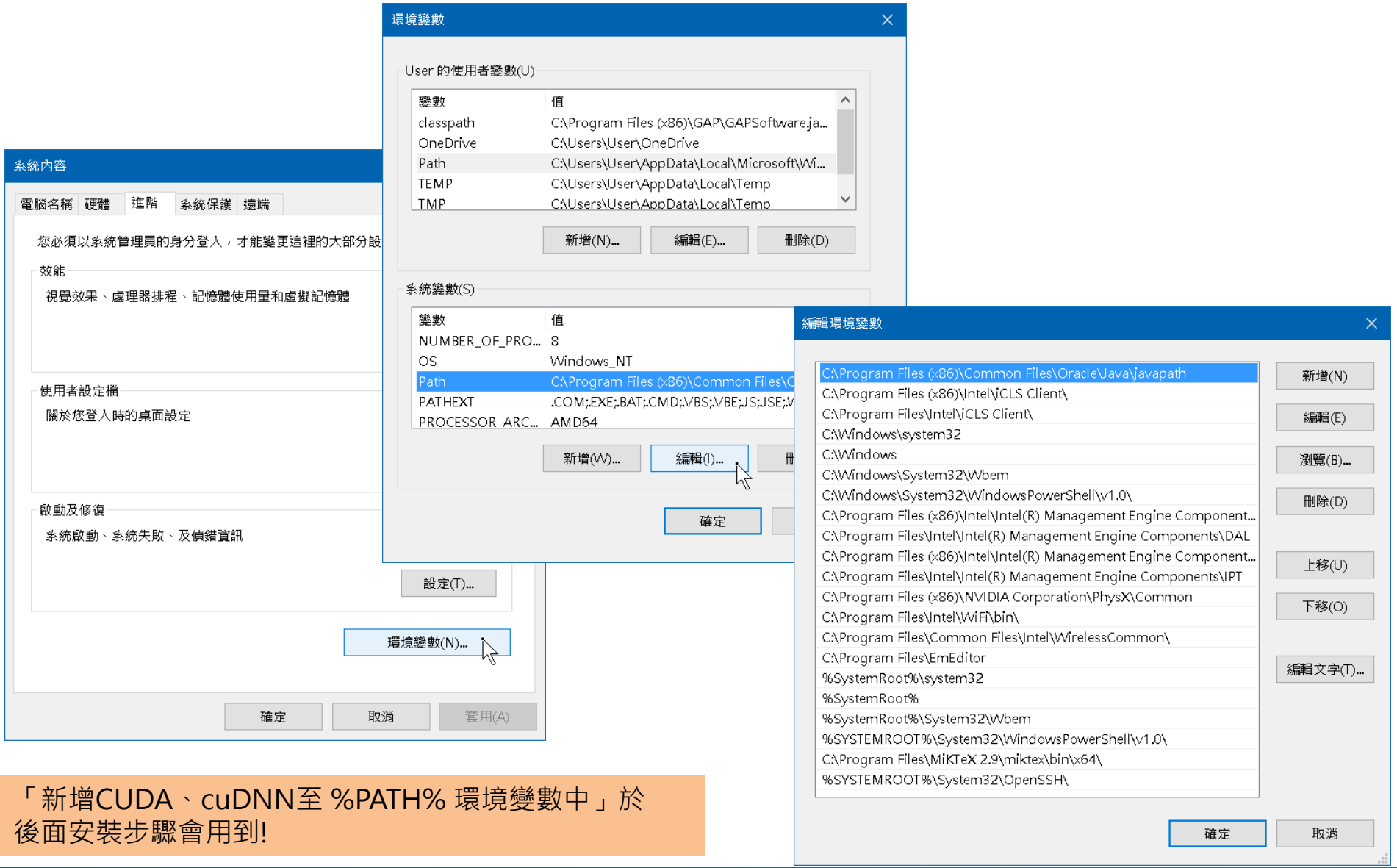
 # 設定環境變數(Windows 10)

- - (Windows鍵+X) 系統(Y) => 「關於」頁面(右側) => 「系統資訊」 => (左側) 進階系統設定
- - 進階(頁籤) => 環境變數 => 系統變數 => 找到「Path」=> 編輯 => 確定





Installation of TensorFlow for R:
前置作業 (設定路徑的方法)



「新增CUDA、cuDNN至 %PATH% 環境變數中」於後面安裝步驟會用到!

Installation of TensorFlow for R: 前置作業 (安裝Rtools)

下載並安裝 Rtools40 (rtools40-x86_64.exe):
https://cran.r-project.org/bin/windows/Rtools/


```
> # 設定路徑
> writeLines('PATH="${RTOOLS40_HOME}\\usr\\bin;${PATH}"', con = "~/.Renviron")
> # Restart R within Rstudio
> .rs.restartR()
> # 測試: show the path to your Rtools installation.
> Sys.which("make")
                                make
"C:\\rtools40\\usr\\bin\\make.exe"
```

Installation of TensorFlow for R: 快速安裝並測試

- https://tensorflow.rstudio.com/installation/
- TensorFlow is tested and supported on the following 64-bit systems: Ubuntu 16.04 or later/**Windows 7 or later**/macOS 10.12.6 (Sierra) or later (no GPU support)。

快速安裝: Quick start

```
> # 於Ggui或Rstudio中，執行下列指令
> install.packages("tensorflow")
> # 載入tensorflow
> library(tensorflow)
> # 安裝 tensorflow、keras、python等等環境
> install_tensorflow()
> # 安裝順利後，R Session會重啟，再次載入tensorflow
> library(tensorflow)
> # 測試安裝結果是否成功
> # 若成功會印出"tf.Tensor(b'Hellow Tensorflow', shape=(), dtype=string)"
> tf$constant("Hellow Tensorflow")
```

```
install_tensorflow(
  method = c("auto", "virtualenv", "conda"),
  conda = "auto",
  version = "default",
  envname = NULL,
  extra_packages = NULL,
  restart_session = TRUE,
  conda_python_version = "3.6",
  ...)
```

```
# Installation methods
# conda: install an Anaconda Python environment
# named r-reticulate
install_tensorflow(method="conda")
# Alternate Versions
install_tensorflow(version = "2.0.0")
install_tensorflow(version = "nightly")
install_tensorflow(version = "nightly-gpu")
```

Installation of TensorFlow for R: 安裝步驟的細節

```
> install.packages("tensorflow")
> library(tensorflow)
> install_tensorflow()
No non-system installation of Python could be found.
Would you like to download and install Miniconda?
Miniconda is an open source environment management system for Python.
See https://docs.conda.io/en/latest/miniconda.html for more details.

Would you like to install Miniconda? [Y/n]:
* Downloading "https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe" ...
嘗試 URL 'https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe'
Content type 'application/octet-stream' length 58431368 bytes (55.7 MB)
downloaded 55.7 MB

* Installing Miniconda -- please wait a moment ...
...
> Sys.setenv(TENSORFLOW_PYTHON='C:/Users/userpc/AppData/Local/r-miniconda/envs/r-
reticulate/Lib/site-packages/tensorflow')
> library(tensorflow)
> tf_config()
TensorFlow v2.0.0-beta1 ()
Python v3.6 (C:/Users/userpc/AppData/Local/r-miniconda/envs/r-reticulate/python.exe)
> tf$constant("Hellow Tensorflow")
tf.Tensor(b'Hellow Tensorflow', shape=(), dtype=string)
```

安裝步驟

換成你的Windows使者名(帳戶)

或 `> Sys.setenv(RETICULATE_PYTHON="/usr/local/bin/python")`

若出現
「錯誤: Installation of TensorFlow not found.」或「錯誤: Python module tensorflow was not found.」

```
> install_tensorflow(version = "2.0.0b1", method = "conda", envname = "r-reticulate")
```

啟動RStudio/tensorflow，並測試是否安裝成功

```
> require(tensorflow)
> # 印出一行字
> # TF2.0: use tf$compat$v1$Session() instead of tf$Session()
> # sess <- tf$Session()
> tf$compat$v1$disable_eager_execution()
> sess <- tf$compat$v1$Session()
> sess
<tensorflow.python.client.session.Session>
> hello <- tf$constant('Hello, TensorFlow!')
> hello
Tensor("Const:0", shape=(), dtype=string)
> sess$run(hello)
b'Hello, TensorFlow!'
>
> # 運算
> alpha <- tf$constant(5)
> alpha
Tensor("Const_1:0", shape=(), dtype=float32)
> beta <- tf$constant(4)
> beta
Tensor("Const_2:0", shape=(), dtype=float32)
> prod <- tf$multiply(alpha, beta)
> prod
Tensor("Mul:0", shape=(), dtype=float32)
> sess$run(prod)
[1] 20
> sess$close()
```

tf: Main TensorFlow module, Interface to main TensorFlow module. Provides access to top level classes and functions as well assub-modules (e.g., **tf$nn**,**tf$contrib$learn**, etc.).

啟動RStudio/keras，並建立一模型，測試是否可成功運作

```
# install R Interface to 'keras'
> install.packages("keras")
> library(keras)
> install_keras()
```

Keras 提供的Layer包括：全連階層(Dense)、Activation layer、Dropout、Flatten、Reshape、Permute、RepeatVector、Lambda、ActivityRegularization、Masking。我們目前只使用到全連階層(Dense)， 它的運算就是 output = activation(dot(input, kernel) + bias)，即前面提到的 y = g(x * W + b)

```
> # define the a Keras model using the sequential API.
> model <- keras_model_sequential()
> model %>%
+     layer_flatten(input_shape = c(28, 28)) %>%
+     layer_dense(units = 128, activation = "relu") %>%
+     layer_dropout(0.2) %>%
+     layer_dense(10, activation = "softmax")
> summary(model)
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_8 (Dense)              (None, 128)               100480
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
>
```

`dropout` layer: randomly sets some fraction of the neurons in each layer to zero during each training step, which can help to avoid overfitting.

Build a neural network that classifies MNIST images dataset.
https://tensorflow.rstudio.com/tutorials/beginners/

安裝R Interface to 'Python': reticulate

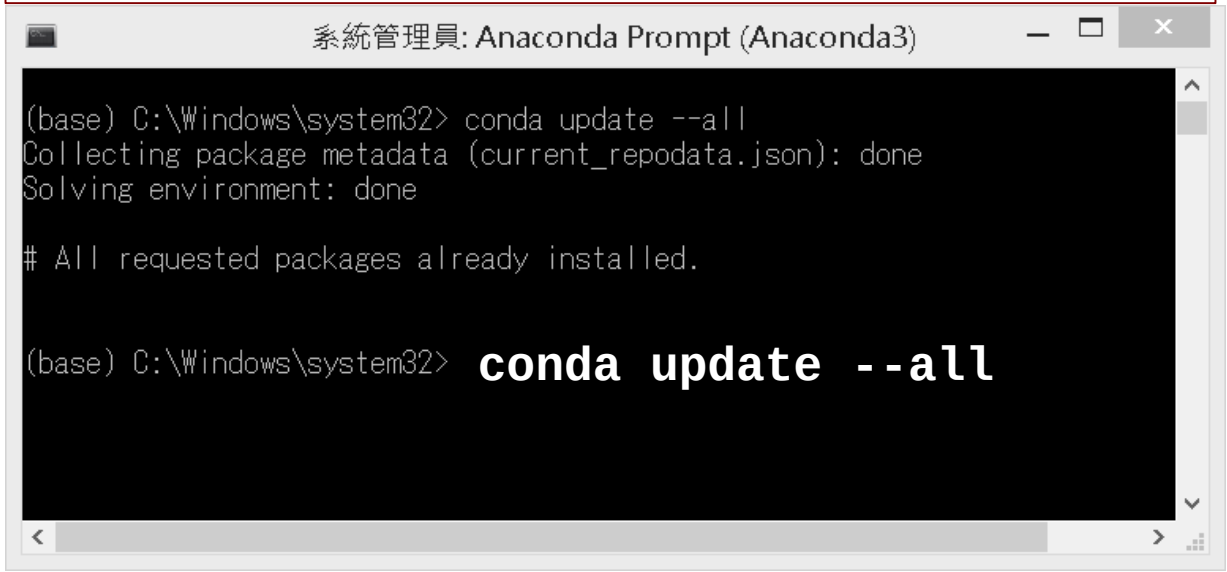
```
> install.packages("reticulate")
> library(reticulate)
> conda_version()
[1] "conda 4.8.4"
> py_available() # Check if Python is available on this system
[1] TRUE
> py_module_available("tensorflow")
[1] TRUE
> py_config() # Python configuration
python:       C:/Users/userpc/AppData/Local/r-miniconda/envs/r-reticulate/python.exe
libpython:    C:/Users/userpc/AppData/Local/r-miniconda/envs/r-reticulate/python36.dll
pythonhome:   C:/Users/userpc/AppData/Local/r-miniconda/envs/r-reticulate
version:      3.6.10|Anaconda, Inc.|(default, May 7 2020, 19:46:08)[MSC v.1916 64 bit (AMD64)]
Architecture: 64bit
numpy:      C:/Users/userpc/AppData/Local/r-miniconda/envs/r-reticulate/Lib/site-packages/numpy
numpy_version:1.19.1
>
> conda_list() # names and paths to the respective python binaries of available environments
name                                                                    python
1           tf                                C:\\ProgramData\\Anaconda3\\envs\\tf\\python.exe
2    Anaconda3                                          C:\\ProgramData\\Anaconda3\\python.exe
3 r-reticulate C:\\Users\\userpc\\AppData\\Local\\r-miniconda\\envs\\r-reticulate\\python.exe
>
> conda_binary() # the location of the main conda binary
[1] "C:/Users/userpc/AppData/Local/r-miniconda/condabin/conda.bat"
```

其它

```
> ## 卸載套件
> detach("package:tensorflow", unload=TRUE)
> ## 移除套件
> remove.packages("tensorflow")
```

```
> sessionInfo()
> Sys.getenv()  # 系統參數
> packageVersion("tensorflow")
> packageVersion("keras")
```

```
## 從Anaconda更新Python套件
# if "Anaconda3-2020.07-Windows-x86_64.exe" is installed
# opened the Anaconda Prompt and updated the conda packages with
conda update --all    (系統管理員身份)
```



R筆記 – (15)Windows安裝深度學習套件：Tensorflow/Keras(R版本)
skydome20 (2017/10/13)
https://rpubs.com/skydome20/R-Note15-R_tensorflow_keras_install_windows

可能的錯誤訊息(1)

```
> library(tensorflow)
> install_tensorflow()
No non-system installation of Python could be found.
Would you like to download and install Miniconda?
Miniconda is an open source environment management system for Python.
See https://docs.conda.io/en/latest/miniconda.html for more details.

Would you like to install Miniconda? [Y/n]: y
* Downloading "https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe" ...
嘗試 URL 'https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe'
Content type 'application/octet-stream' length 58431368 bytes (55.7 MB)
downloaded 55.7 MB

* Installing Miniconda -- please wait a moment ..

Installation complete.

Restarting R session...
Error in gzfile(file, "wb") : 無法開啟連結
Error saving session (options): R code execution
WARNING: Forcing suspend of process in spite of a
```

```
> sessionInfo()
R version 4.0.2 (2020-06-22)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 8.1 x64 (build 9600)

Matrix products: default

locale:
[1] LC_COLLATE=Chinese (Traditional)_Taiwan.950
[2] LC_CTYPE=Chinese (Traditional)_Taiwan.950
[3] LC_MONETARY=Chinese (Traditional)_Taiwan.950
[4] LC_NUMERIC=C
[5] LC_TIME=Chinese (Traditional)_Taiwan.950
```

```
> library(tensorflow)
> tf$constant("Hellow Tensorflow")
錯誤: Installation of TensorFlow not found.

Python environments searched for 'tensorflow' package:
 C:\Users\userpc\AppData\Local\r-miniconda\envs\r-reticulate\python.exe
You can install TensorFlow using the install_tensorflow() function.
```

可能的錯誤訊息(2)

```
> tf$Session()
錯誤: Python module tensorflow was not found.

Detected Python configuration:

python:         C:/Users/userpc/AppData/Local/r-miniconda/envs/r-reticulate/python.exe
libpython:      C:/Users/userpc/AppData/Local/r-miniconda/envs/r-reticulate/python36.dll
pythonhome:     C:/Users/userpc/AppData/Local/r-miniconda/envs/r-reticulate
version:        3.6.10 |Anaconda, Inc.| (default, May  7 2020, 19:46:08) [MSC v.1916 64 bit
(AMD64)]
Architecture:   64bit
numpy:          C:/Users/userpc/AppData/Local/r-miniconda/envs/r-reticulate/Lib/site-
packages/numpy
numpy_version:  1.19.1
```

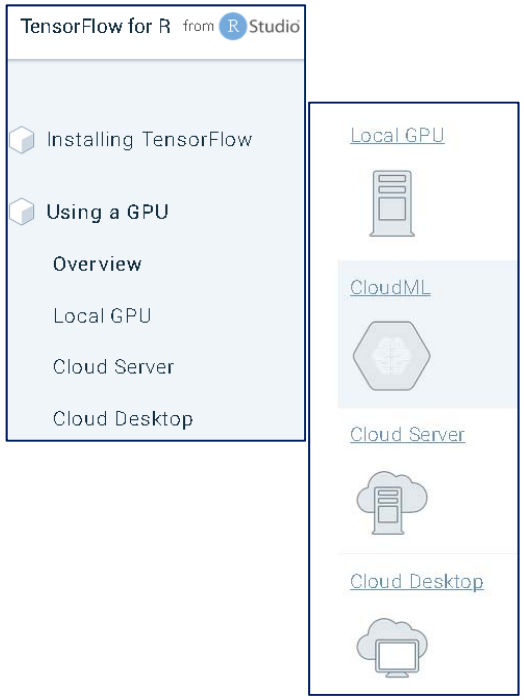
Solved:

```
> library(tensorflow)
> install_tensorflow(version = "2.0.0b1", method = "conda", envname = "r-reticulate")
```

使用 NVIDIA GPU 運算

硬體需求:

- For systems that have a recent, high-end **NVIDIA® GPU**, TensorFlow is available in a GPU version that takes advantage of the CUDA and cuDNN libraries to accelerate training performance.
- 支援搭載 CUDA® Compute Capability 3.5 以上版本的 NVIDIA® GPU 顯示卡。請參閱採用 CUDA 的 GPU 顯示卡清單。https://developer.nvidia.com/cuda-gpus
- Note that the GPU version of TensorFlow is currently only supported on **Windows and Linux** (there is no GPU version available for **Mac OS X** since NVIDIA GPUs are not commonly available on that platform).



軟體需求 (注意版本搭配):

- NVIDIA® GPU 驅動程式：CUDA 10.1 需要 418.x 以上版本。
  https://www.nvidia.com/drivers
  452.06-desktop-win10-64bit-international-nsd-dch-whql.exe

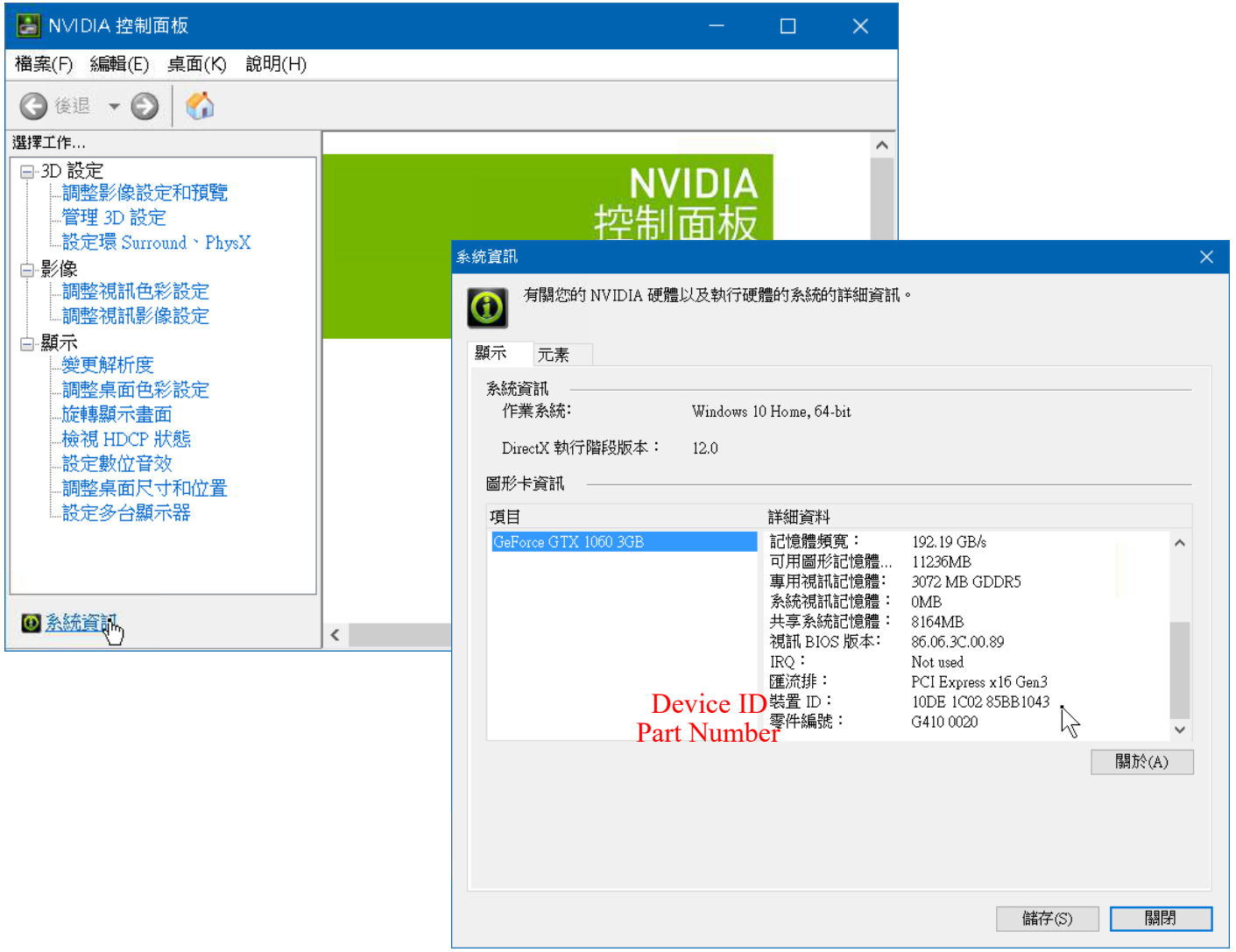
- CUDA® Toolkit：TensorFlow 支援 CUDA 10.1 (TensorFlow 2.1.0 以上版本)
  https://developer.nvidia.com/cuda-toolkit-archive
  cuda_10.1.243_426.00_win10.exe (選擇合適的版本)
  cuda_11.0.3_451.82_win10.exe

- cuDNN (CUDA Deep Neural Network library) SDK (7.6 以上版本):
  https://developer.nvidia.com/cudnn
  cudnn-10.1-windows10-x64-v8.0.2.39.zip (選擇合適的版本)
  cudnn-10.2-windows10-x64-v8.0.2.39.zip
  cudnn-11.0-windows-x64-v8.0.2.39.zip

Identifying the (NVIDIA®) Graphics Card Model and Device ID in a PC

檢查顯示卡是否使用NVIDIA GPU，並找出Device ID

桌面，按滑鼠右鍵



Windows 作業系統需求: 下載並安裝Visual Studio

https://visualstudio.microsoft.com/zh-hant/downloads/

System Requirements

To use CUDA on your system, you will need the following installed:

- A CUDA-capable GPU
- A supported version of Microsoft Windows
- A supported version of Microsoft Visual Studio
- the NVIDIA CUDA Toolkit (available at http://developer.nvidia.com/cuda-downloads)

The next two tables list the currently supported Windows operating systems and compilers.

Table 1. Windows Operating System Support in CUDA 11.0

Operating System	Native x86_64	Cross (x86_32 on x86_64)
Windows 10	YES	YES
Windows Server 2019	YES	NO
Windows Server 2016	YES	NO

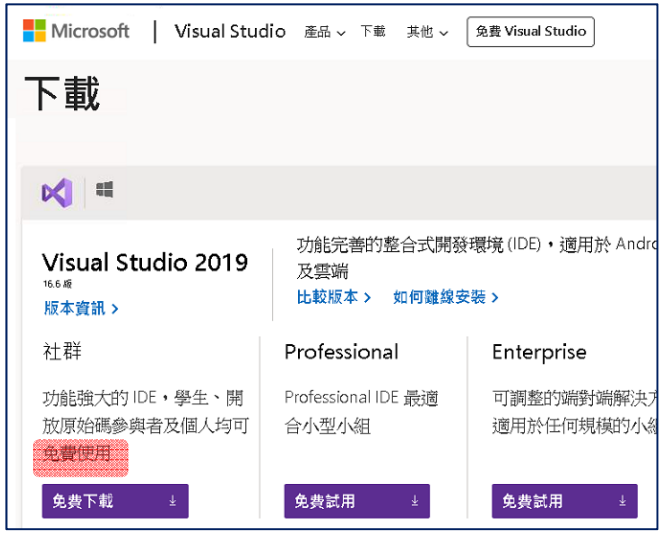


Table 2. Windows Compiler Support in CUDA 11.0

Compiler*	IDE	Native x86_64	Cross (x86_32 on x86_64)
MSVC Version 192x	Visual Studio 2019 16.x (RTW and all updates)	YES	YES
MSVC Version 191x	Visual Studio 2017 15.x (RTW and all updates)	YES	YES
MSVC Version 1900	Visual Studio 2015 14.0 (RTW and updates 1, 2, and 3)	YES	YES
	Visual Studio Community 2015	YES	YES
MSVC Version 1800	Visual Studio 2013 12.0	YES	YES
MSVC Version 1700	Visual Studio 2012 11.0	YES	YES

說明文件:

- CUDA Toolkit 隨附 CUPTI (The API reference guide fir CUPTI, the CUDA Profiling Tools Interface. http://docs.nvidia.com/cuda/cupti/
- 適用於 Windows 的 CUDAR 安裝指南。 https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/

下載/更新 NVIDIA® GPU 驅動程式

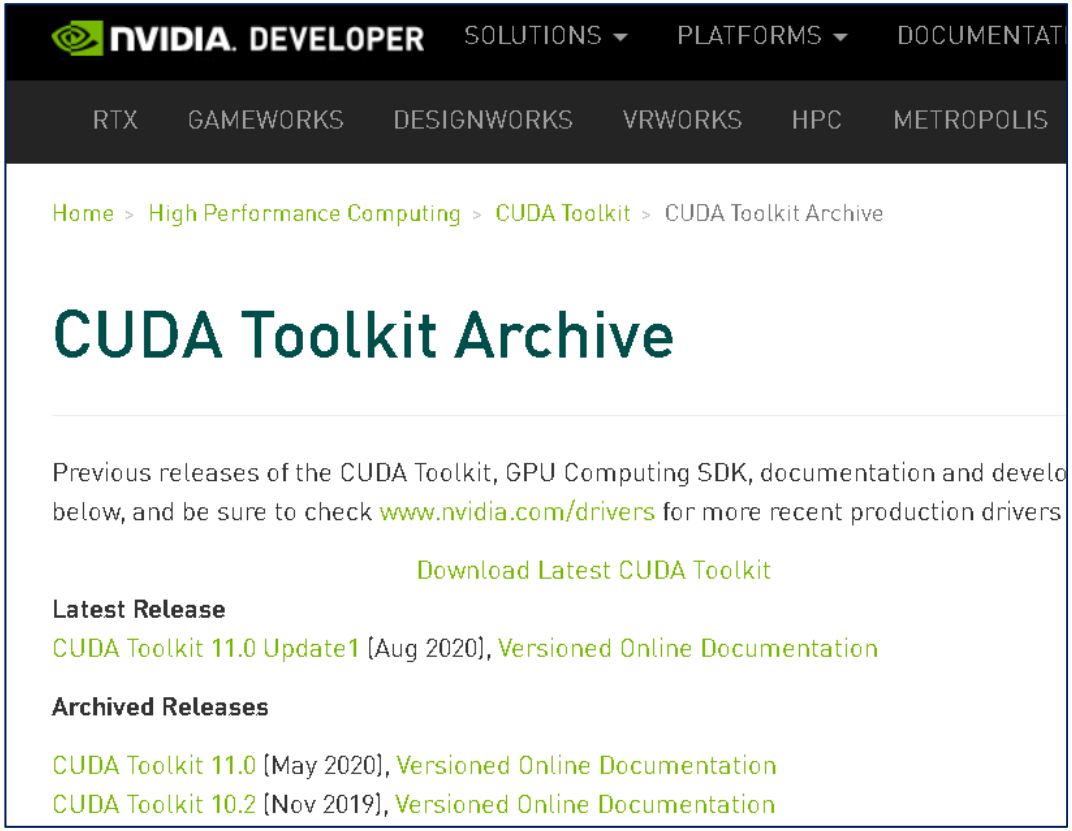
https://www.nvidia.com.tw/Download/index.aspx?lang=tw

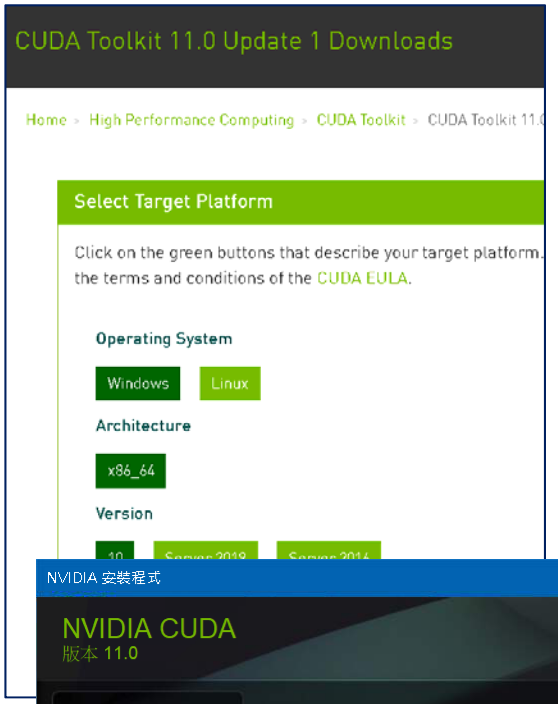
GPU加速運算安裝若需要GPU加速運算，需要Nvidia的顯示卡，並安裝CUDA® Toolkit 8.0及cuDNN v6，且顯示卡的CUDA Compute Capability須高於3分。

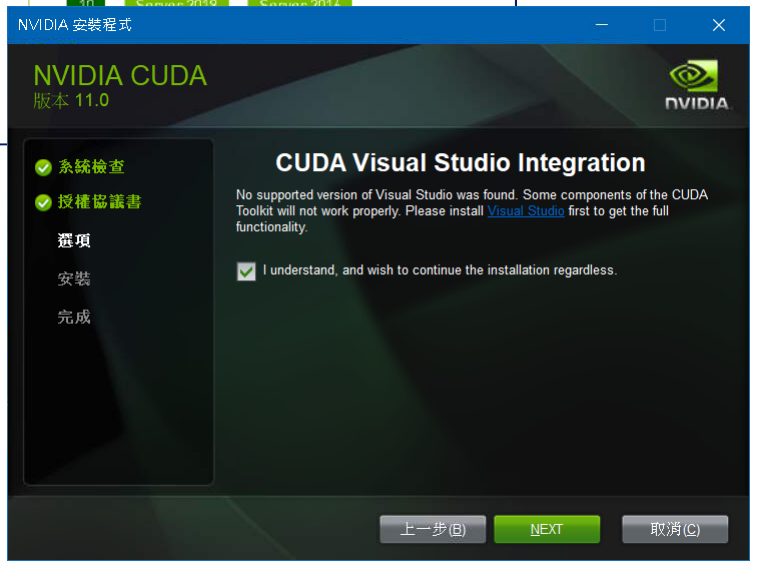
查詢GPU的Compute Capability網址：
https://developer.nvidia.com/cuda-gpus/

下載並安裝 CUDA Toolkit

https://developer.nvidia.com/cuda-toolkit-archive

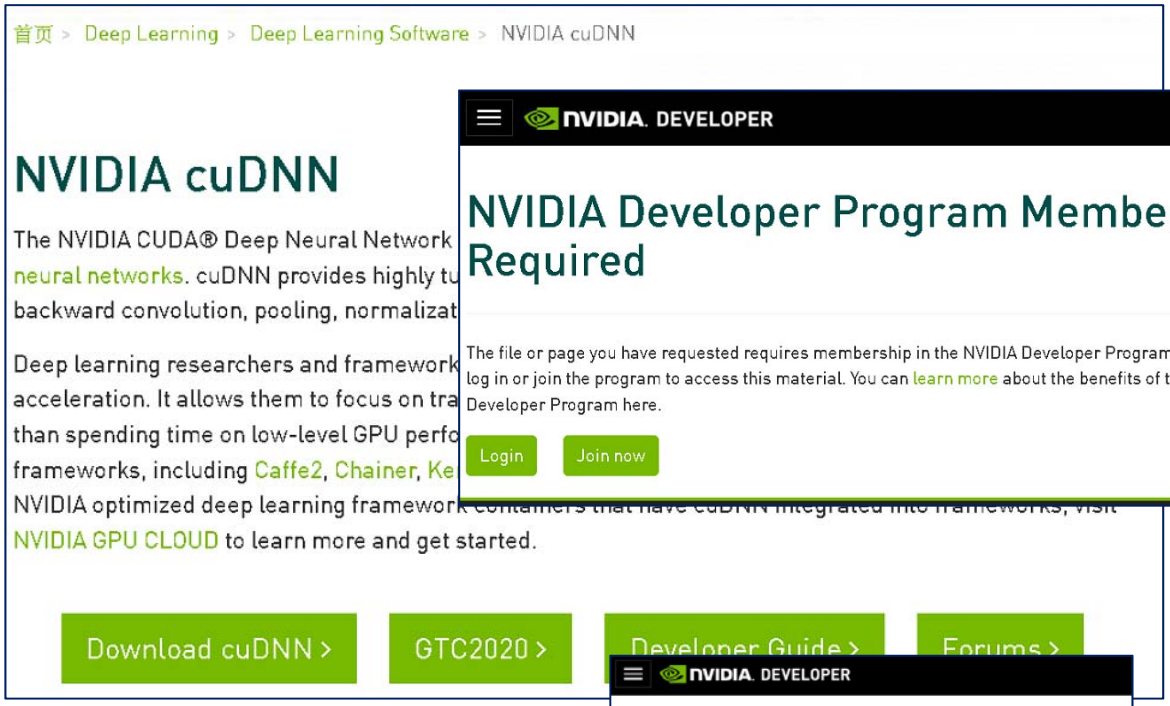


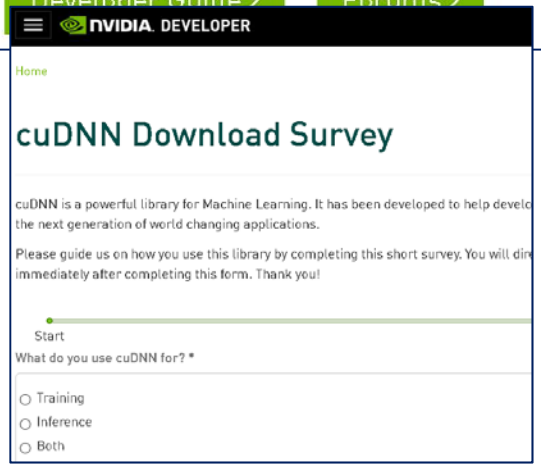




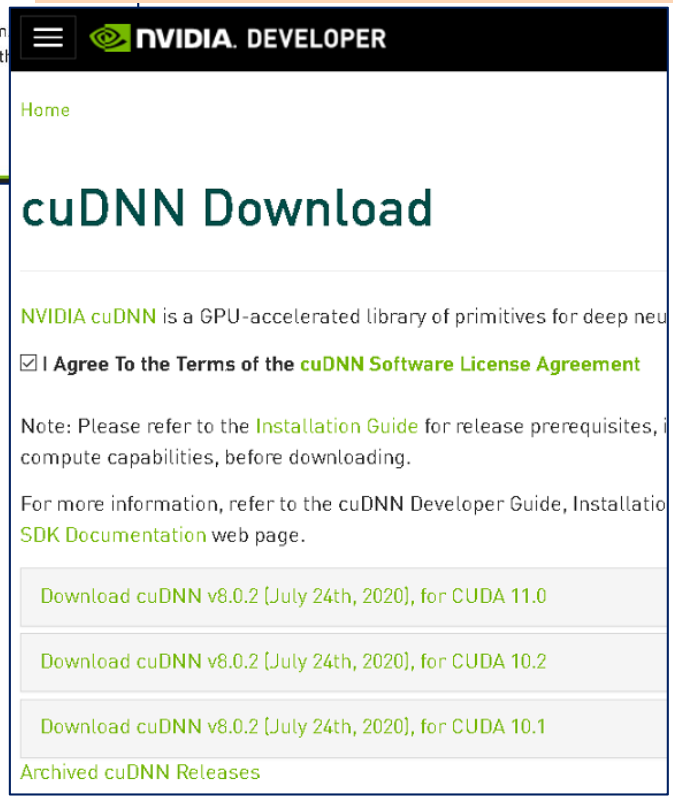
下載 cuDNN

https://developer.nvidia.com/cudnn





注意: 需註冊會員後，始可下載

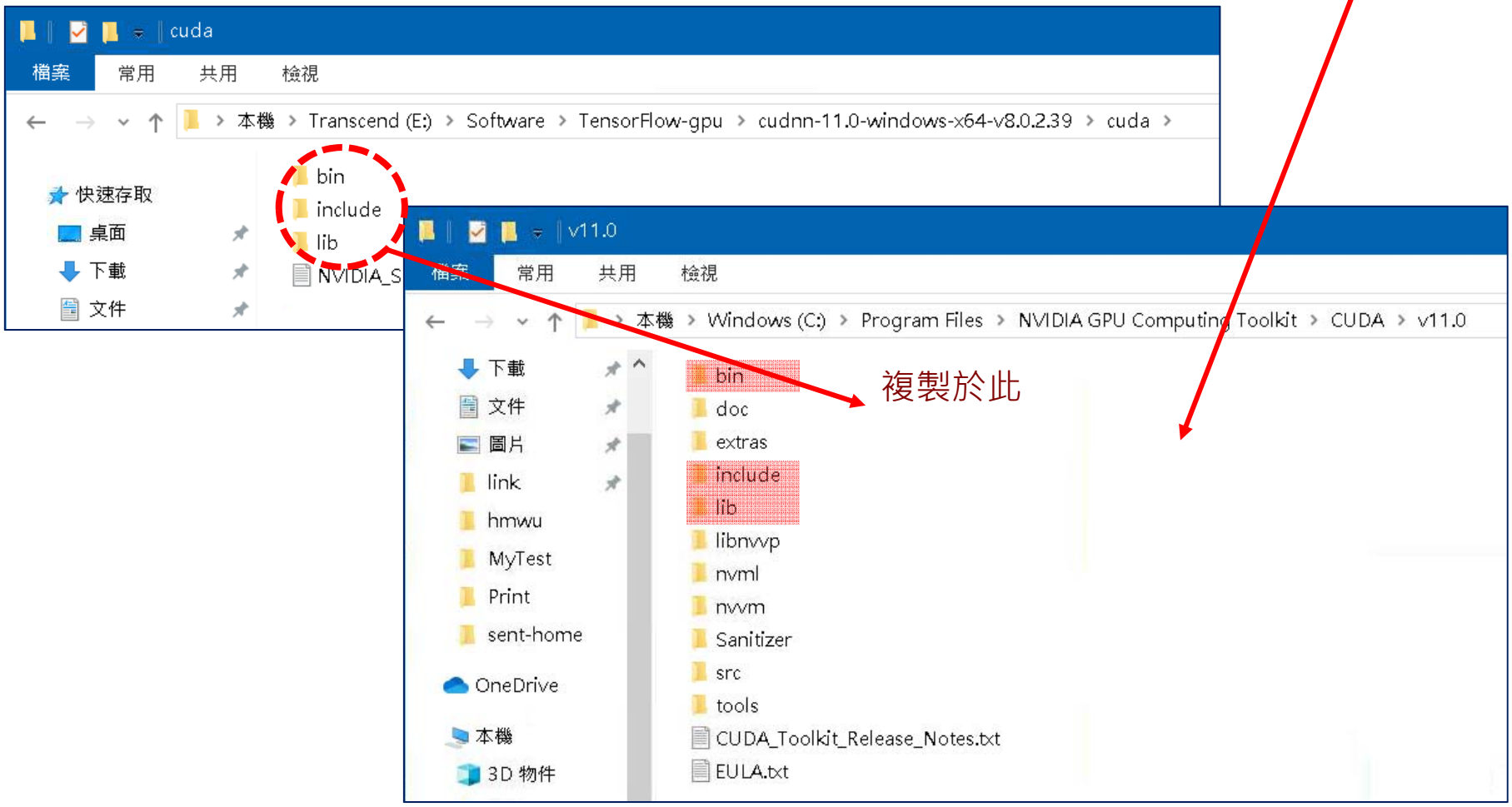


將cuDNN之子目錄解壓縮至CUDA\v11.0\

若是不同版本
作法一樣

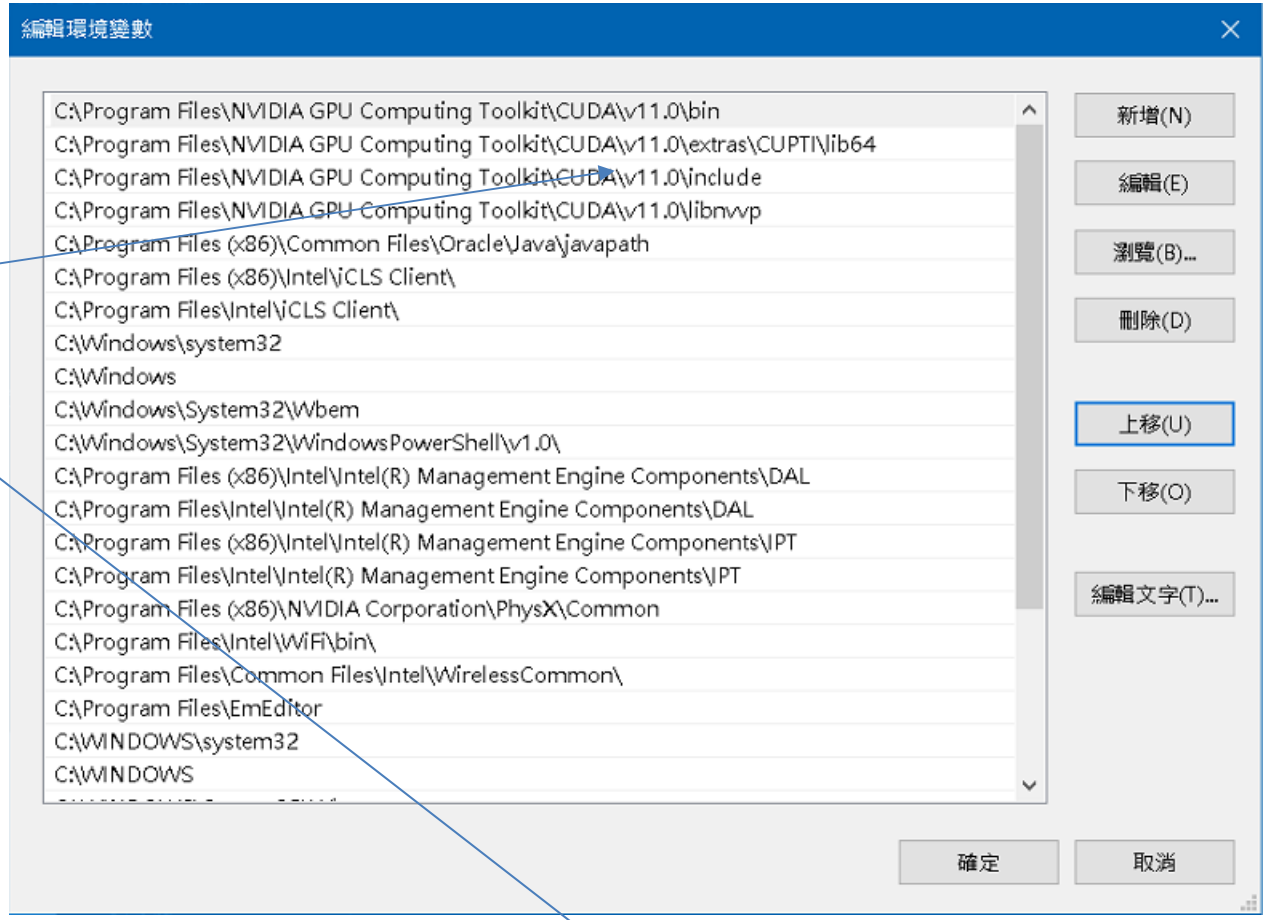
cudnn-11.0-windows-x64-v8.0.2.39.zip

或解壓縮至



新增CUDA、cuDNN至 %PATH% 環境變數中

可能不同版本



上述必需之軟體安裝後，將 CUDA、CUPTI 和 cuDNN 安裝目錄新增至 %PATH% 環境變數中。

```
SET PATH=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0\bin;%PATH%
SET PATH=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0\extras\CUPTI\lib64;%PATH%
SET PATH=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0\include;%PATH%
```

https://www.tensorflow.org/install/gpu#hardware_requirements

安裝Keras和GPU版本的TensorFlow並測試

```
> # 安裝GPU版本的TensorFlow
> # (a single-user / desktop environment):
> library(tensorflow)
> install_tensorflow(version = "gpu")
```

```
> # 或同時安裝 Keras 和 GPU版本的TensorFlow:
> library(keras)
> install_keras(tensorflow = "gpu")
```

```
> # 測試:
> library(keras)
> tensorflow::tf$test$is_gpu_available()
> tensorflow::tf$config$list_physical_devices('GPU')
```

```
>
> library(keras)
> tensorflow::tf$test$is_gpu_available()
2020-08-21 23:07:41.917678: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: GeForce GTX 1060 3GB computeCapability: 6.1
coreClock: 1.7085GHz coreCount: 9 deviceMemorySize: 3.00GiB deviceMemoryBandwidth: 178.99GiB/s
2020-08-21 23:07:41.918541: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudart64_101.dll
2020-08-21 23:07:41.919004: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cublas64_10.dll
2020-08-21 23:07:41.919364: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cufft64_10.dll
2020-08-21 23:07:41.919610: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library curand64_10.dll
2020-08-21 23:07:41.919911: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cusolver64_10.dll
2020-08-21 23:07:41.920219: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cusparse64_10.dll
2020-08-21 23:07:41.920514: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully opened dynamic library cudnn64_7.dll
2020-08-21 23:07:41.920822: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1703] Adding visible gpu devices: 0
2020-08-21 23:07:41.921055: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1102] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-08-21 23:07:41.921303: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1108]      0
2020-08-21 23:07:41.921454: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1121] 0:   N
2020-08-21 23:07:41.921739: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1247] Created TensorFlow device (/device:GPU:0 with 2100 MB memory) -
rce GTX 1060 3GB, pci bus id: 0000:01:00.0, compute capability: 6.1)
[1] TRUE
> tensorflow::tf$config$list_physical_devices('GPU')
[[1]]
PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')
```

成功!!

可能出現的錯誤 (1)

```
...
Successfully opened dynamic library cudart64_101.dll
...
Could not load dynamic library 'cudnn64_7.dll'; dlerror: cudnn64_7.dll not found
```

Solved:

Download **cuDNN v7.6.5 (November 5th, 2019)**, for CUDA 10.1
(**cudnn-10.1-windows10-x64-v7.6.5.32.zip**)
extract "**cudnn64_7.dll**"

https://developer.nvidia.com/cudnn

0

Just fould the solution:

I checked the \tensorflow\python\platform\build_info.py and found:

```
msvcp_dll_name = 'msvcp140.dll'
cudart_dll_name = 'cudart64_90.dll'
cuda_version_number = '9.0'
nvcuda_dll_name = 'nvcuda.dll'
cudnn_dll_name = 'cudnn64_7.dll'
cudnn_version_number = '7'
```

It assumes cudnn version is 7. So just need to correct it as:

```
cudnn_dll_name = 'cudnn64_6.dll'
cudnn_version_number = '6'
```

可能出現的錯誤 (2)

警告訊息: Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2

https://stackoverflow.com/questions/47068709/your-cpu-supports-instructions-that-this-tensorflow-binary-was-not-compiled-to-u

What is this warning about?

848

Modern CPUs provide a lot of low-level instructions, besides the usual arithmetic and logic, known as extensions, e.g. SSE2, SSE4, AVX, etc. From the Wikipedia:

Advanced Vector Extensions (**AVX**) are extensions to the x86 instruction set architecture for microprocessors from Intel and AMD proposed by Intel in March 2008 and first supported by Intel with the Sandy Bridge processor shipping in Q1 2011 and later on by AMD with the Bulldozer processor shipping in Q3 2011. AVX provides new featu
scheme.

If you don't have a GPU and want to utilize CPU as much as possible, **you should build tensorflow from the source optimized for *your* CPU** with AVX, AVX2, and FMA enabled if your CPU supports them. It's been discussed in this question and also this GitHub issue. Tensorflow uses an ad-hoc build system called bazel and building it is not that trivial, but is certainly doable. After this, not only will the warning disappear, tensorflow performance should also improve.

```
> model <- keras_model_sequential()
> model %>%
+   layer_flatten(input_shape = c(28, 28)) %>%
+   layer_dense(units = 128, activation = "relu") %>%
+   layer_dropout(0.2) %>%
+   layer_dense(10, activation = "softmax")
2020-08-22 22:40:38.476411: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU
 supports instructions that this TensorFlow binary was not compiled to use: AVX2
```

```
> model %>%
+     layer_flatten(input_shape = c(28, 28)) %>%
+     layer_dense(units = 128, activation = "relu") %>%
+     layer_dropout(0.2) %>%
+     layer_dense(10, activation = "softmax")
2020-08-22 22:48:27.511136: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1561] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: GeForce GTX 1060 3GB computeCapability: 6.1
coreClock: 1.7085GHz coreCount: 9 deviceMemorySize:
2020-08-22 22:48:27.512148: I tensorflow/stream_exec
2020-08-22 22:48:27.512667: I tensorflow/stream_exec
2020-08-22 22:48:27.512975: I tensorflow/stream_exec
2020-08-22 22:48:27.513260: I tensorflow/stream_exec
2020-08-22 22:48:27.513537: I tensorflow/stream_exec
2020-08-22 22:48:27.513822: I tensorflow/stream_exec
2020-08-22 22:48:27.514368: I tensorflow/stream_exec
2020-08-22 22:48:27.514692: I tensorflow/core/common
2020-08-22 22:48:27.522711: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1102] Device interconnect StreamExecutor with strength 1 edge matrix:
2020-08-22 22:48:27.523069: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1108]      0
2020-08-22 22:48:27.523277: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1121] 0:   N
2020-08-22 22:48:27.526857: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1247] Created TensorFlow device (/job:localhost/replica:0/task:0/devic
mory) -> physical GPU (device: 0, name: GeForce GTX 1060 3GB, pci bus id: 0000:01:00.0, compute capability: 6.1)
```

What should you do?

If you have a GPU, you shouldn't care about AVX support, because most expensive ops will be dispatched on a GPU device (unless explicitly set not to). In this case, you can simply ignore this warning by

以上安裝測試完，印出相關系統資訊

```
> sessionInfo()
R version 4.0.2 (2020-06-22)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 18362)

Matrix products: default

Random number generation:
 RNG:     Mersenne-Twister
 Normal:  Inversion
 Sample:  Rounding

locale:
[1] LC_COLLATE=Chinese (Traditional)_Taiwan.
[2] LC_CTYPE=Chinese (Traditional)_Taiwan.95
[3] LC_MONETARY=Chinese (Traditional)_Taiwar
[4] LC_NUMERIC=C
[5] LC_TIME=Chinese (Traditional)_Taiwan.950

attached base packages:
```

```
> tensorflow::tf_config()
TensorFlow v2.2.0 ()
Python v3.6 (C:/Users/User/AppData/Local/r-minicon
> keras:::keras_version()
[1] '2.3.0'
```

```
> Sys.getenv()
ALLUSERSPROFILE                 C:\ProgramData
APPDATA                         C:\Users\User\AppData\Roaming
classpath                       C:\Program Files (x86)\GAP\GAPSoftware.jar;C:\Program Files
                                (x86)\GAP\lib\epsgraphics.jar;C:\Program Files
                                (x86)\GAP\lib\jcommon-0.9.5.jar;C:\Program Files
                                (x86)\GAP\lib\jfreechart-0.9.20.jar;C:\Program Files
                                (x86)\GAP\lib\jmathplot.jar;
CLICOLOR_FORCE                  1
CommonProgramFiles              C:\Program Files\Common Files
CommonProgramFiles(x86)         C:\Program Files (x86)\Common Files
CommonProgramW6432              C:\Program Files\Common Files
COMPUTERNAME                    HMWU-OFFICE
ComSpec                         C:\WINDOWS\system32\cmd.exe
CUDA_PATH                       C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1
CUDA_PATH_V10_1                 C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1
CUDA_PATH_V11_0                 C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0
DISPLAY                         :0
DriverData                      C:\Windows\System32\Drivers\DriverData
GFORTRAN_STDERR_UNIT            -1
GFORTRAN_STDOUT_UNIT            -1
HOME                            C:\Users\User\Documents
HOMEDRIVE                       C:
HOMEPATH                        \Users\User
LOCALAPPDATA                    C:\Users\User\AppData\Local
LOGONSERVER                     \\HMWU-OFFICE
MEmu_Path                       C:\Program Files (x86)\Microvirt
MPLENGINE                       tkAgg
MSYS2_ENV_CONV_EXCL             R_ARCH
NUMBER_OF_PROCESSORS            8
NVCUDASAMPLES_ROOT              C:\ProgramData\NVIDIA Corporation\CUDA Samples\v10.1
NVCUDASAMPLES10_1_ROOT          C:\ProgramData\NVIDIA Corporation\CUDA Samples\v10.1
NVCUDASAMPLES11_0_ROOT          C:\ProgramData\NVIDIA Corporation\CUDA Samples\v11.0
NVTOOLSEXT_PATH                 C:\Program Files\NVIDIA Corporation\NvToolsExt\
OneDrive                        C:\Users\User\OneDrive
OS                              Windows_NT
PATH                            C:\Users\User\AppData\Local\r-miniconda\envs\r-reticulate;C:\
\AppData\Local\r-miniconda\envs\r-reticulate\Library\bin;C:\rtools40\usr\bin;C:\Program
                                Files\R\R-4.0.2\bin\x64;C:\Program Files\NVIDIA GPU Computing
                                                        Files\NVIDIA GPU Computing
                                                        gram Files\NVIDIA GPU Comput
```

```
> reticulate::py_config()
python:         C:/Users/User/AppData/Local/r-miniconda/envs/r-reticulate/python.exe
libpython:      C:/Users/User/AppData/Local/r-miniconda/envs/r-reticulate/python36.dll
pythonhome:     C:/Users/User/AppData/Local/r-miniconda/envs/r-reticulate
version:        3.6.10 |Anaconda, Inc.| (default, May  7 2020, 19:46:08) [MSC v.1916 64 bit (AMD64)]
Architecture:   64bit
numpy:          C:/Users/User/AppData/Local/r-miniconda/envs/r-reticulate/Lib/site-packages/numpy
numpy_version:  1.19.1
```

RStudio.com TensorFlow範例教程

https://tensorflow.rstudio.com/tutorials/

Tutorials

In the tutorials section you will find documentation for solving common Machine Learning problems using TensorFlow.

Overview

In this section you will find tutorials that can be used to get started with TensorFlow for R or, for more advanced users, to discover best practices for loading data, building complex models and solving common problems.

The best place to get started with TensorFlow is using Keras - a Deep Learning API created by François Chollet and ported to R by JJ Allaire. Keras makes it easy to get started, and it allows you to progressively build more complex workflows as you need to use advanced models and techniques.

For beginners

We recommend the following tutorials for your first contact with TensorFlow. Feel free to navigate through the 'beginners' section in the sidebar.

- Quickstart: the minimal getting started guide to Keras.
- Basic ML with Keras: use Keras to solve basic Machine Learning tasks.
- Load data: learn to efficiently load data to TensorFlow using `tfdatasets`.

For experts

- Advanced Quickstart: learn the subclassing API and how to create custom loops.
- Customization: build custom layers and training loops in TensorFlow.
- Distributed Training: distribute your model training across multiple GPU's or machines.

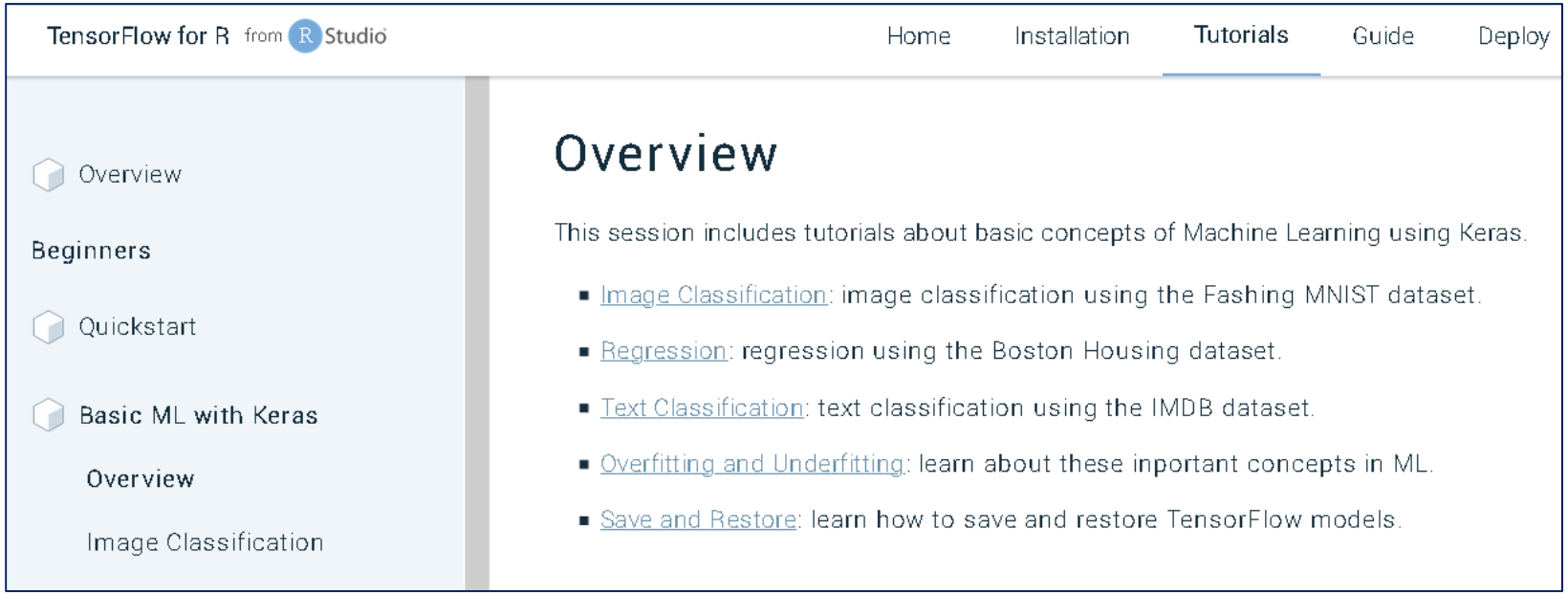
We also provide tutorials focused on different types of data:

- Images: Build more advanced models for classification and segmentation of images.
- Structured Data: Build models for structured data.

範例: 以Keras建立一個神經網路模型進行影像分類

This short introduction uses **`Keras`** to:

1. Build a neural network that classifies images.
2. Train this neural network.
3. And, finally, evaluate the accuracy of the model.
4. Save and restore the created model.



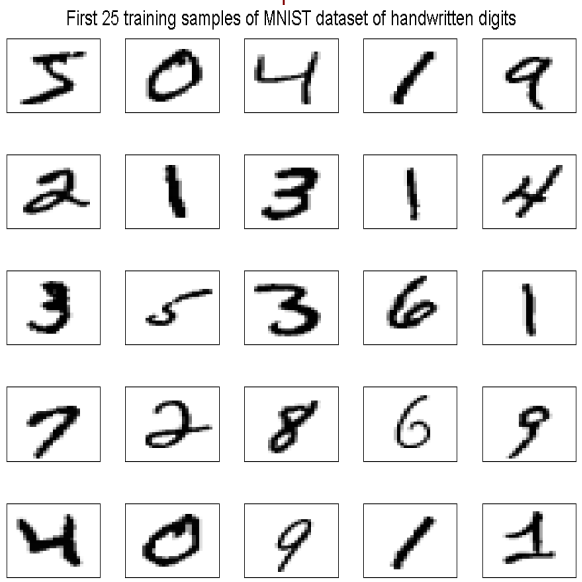
https://tensorflow.rstudio.com/tutorials/beginners/basic-ml/

範例: THE MNIST DATABASE of handwritten digits

Quickstart: the minimal getting started guide to Keras

- MNIST ("Modified National Institute of Standards and Technology")
- the classic MNIST dataset—often used as the “Hello, World” of machine learning programs for computer vision.

```
> library(keras)
> # loading the MNIST dataset
> mnist <- dataset_mnist()
> str(mnist)
List of 2
 $ train:List of 2
  ..$ x: int [1:60000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ y: int [1:60000(1d)] 5 0 4 1 9 2 1 3 1 4 ...
 $ test :List of 2
  ..$ x: int [1:10000, 1:28, 1:28] 0 0 0 0 0 0 0 0 0 0 ...
  ..$ y: int [1:10000(1d)] 7 2 1 0 4 1 4 9 5 9 ...
>
> # Display the first 25 images from the training set
> par(mfrow=c(5, 5), mai=c(0.1, 0.2, 0.4, 0.2))
> empty <- lapply(1:25, function(i){
+   x <- mnist$train$x[i,,]
+   image(t(x)[, nrow(x):1], axes=FALSE, col=grey(255:0/255))
+   box()
+ })
> mtext("First 25 training samples of MNIST dataset of handwritten digits",
+       side=3, line=-2, outer=TRUE, cex=1.5)
>
> # convert pixels values from (0~255) to (0~1)
> mnist$train$x <- mnist$train$x/255
> mnist$test$x <- mnist$test$x/255
```



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

Define the a Keras model using the sequential API

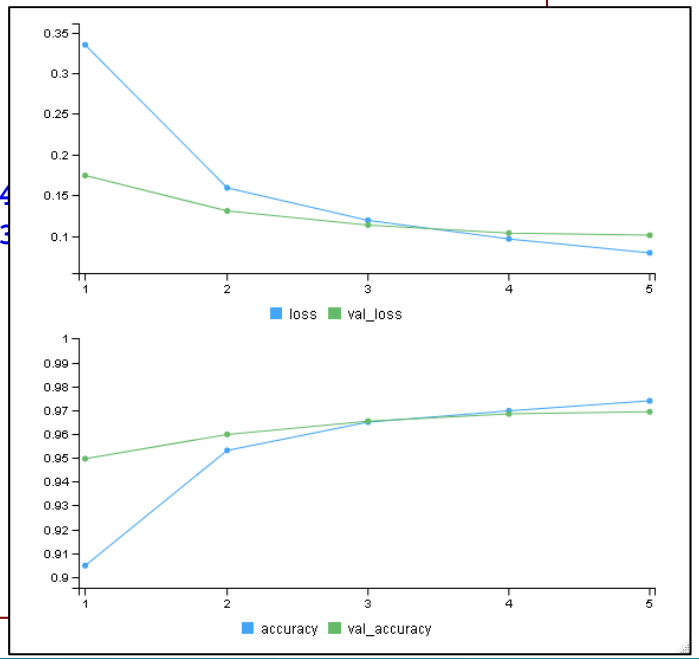

```
> model <- keras_model_sequential()
> # Buildinng the model
> # Note that when using the Sequential API
> # the first layer must specify the input_shape argument
> # which represents the dimensions of the input (images 28x28)
> model %>%
+   layer_flatten(input_shape=c(28, 28)) %>%
+   layer_dense(units=128, activation="relu") %>%
+   layer_dropout(0.2) %>%
+   layer_dense(10, activation="softmax")
>
> # print out layers, number of parameters, etc
> summary(model)
Model: "sequential_1"
_____________________________________________________________________________
Layer (type)                     Output Shape                  Param #
=============================================================================
flatten_1 (Flatten)              (None, 784)                   0
_____________________________________________________________________________
dense_2 (Dense)                  (None, 128)                   100480
_____________________________________________________________________________
dropout_1 (Dropout)              (None, 128)                   0
_____________________________________________________________________________
dense_3 (Dense)                  (None, 10)                    1290
=============================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_____________________________________________________________________________
```

Define the a Keras model using the sequential API


```
> # Compiling the model
> # Define what loss will be optimized and what optimizer will be used.
> # You can also specify metrics, callbacks and etc that are meant
> # to be run during the model fitting.
> model %>%
+   compile(loss="sparse_categorical_crossentropy",
+           optimizer="adam",
+           metrics="accuracy")
>
> # Fit the model
> model %>%
+   fit(x=mnist$train$x,
+       y=mnist$train$y,
+       epochs=5,
+       validation_split=0.3,
+       verbose=2)
Train on 42000 samples, validate on 18000 samples
Epoch 1/5
42000/42000 - 4s - loss: 0.3348 - accuracy: 0.9047 - val_loss: 0.1744 - val_accuracy: 0.9496
Epoch 2/5
42000/42000 - 4s - loss: 0.1593 - accuracy: 0.9531 - val_loss: 0.1309 - val_accuracy: 0.9598
Epoch 3/5
42000/42000 - 4s - loss: 0.1194 - accuracy: 0.9650 - val_loss: 0.1136 - val_accuracy: 0.9654
Epoch 4/5
42000/42000 - 4s - loss: 0.0966 - accuracy: 0.9697 - val_loss: 0.1038 - val_accuracy: 0.9684
Epoch 5/5
42000/42000 - 4s - loss: 0.0795 - accuracy: 0.9739 - val_loss: 0.1010 - val_accuracy: 0.9693
```

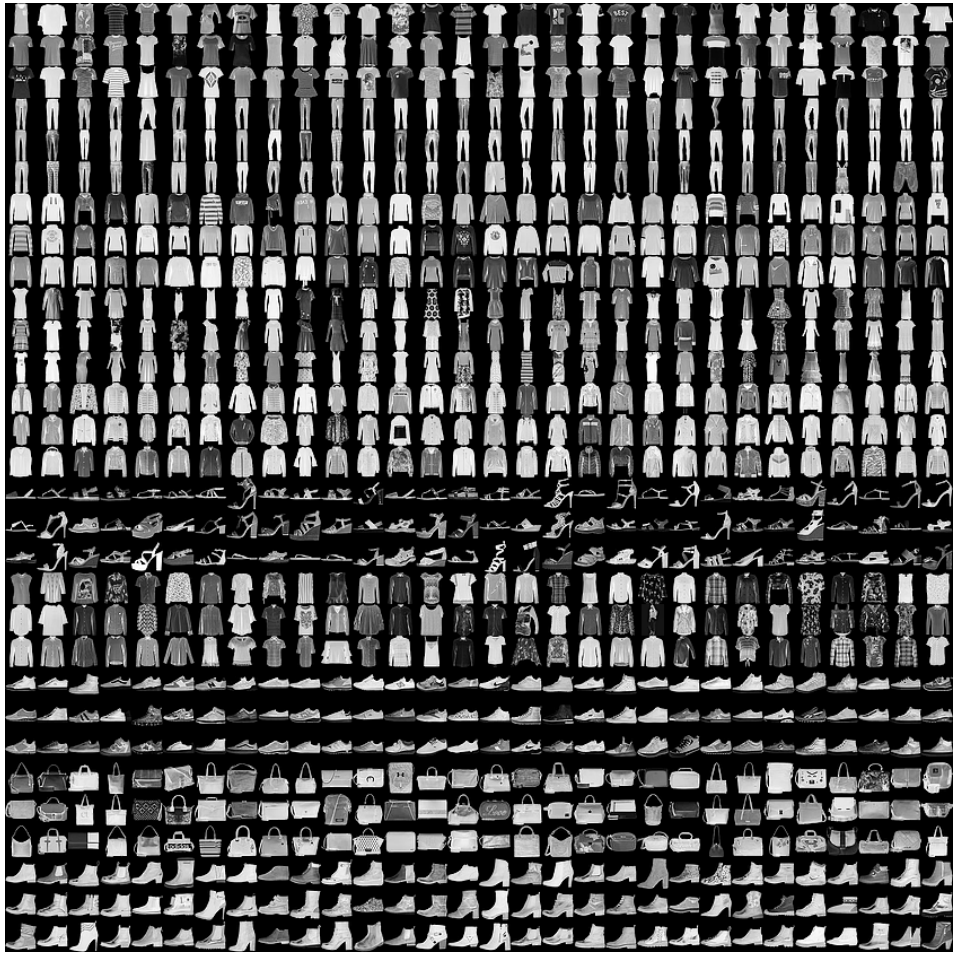
Define the a Keras model using the sequential API


```
> # make predictions
> predictions <- predict(model, mnist$test$x)
> head(predictions, 2)
             [,1]         [,2]         [,3]         [,4]         [,5]         [,6]
[1,] 4.436160e-07 3.306080e-08 0.0000719672 1.470676e-04 4.095156e-10 4.301288e-08
[2,] 3.864101e-08 3.210807e-05 0.9999113083 5.517886e-05 3.312657e-12 6.469448e-08
...
>
> # By default predict will return the output of the last Keras layer.
> # In our case this is the probability for each class.
> # use predict_classes and predict_proba to generate class and probability
> predictions.classes <- predict_classes(model, mnist$test$x)
> head(predictions.classes, 2)
[1] 7 2
>
> predictions.proba <- predict_proba(model, mnist$test$x)
> head(predictions.proba, 2)
             [,1]         [,2]         [,3]         [,4]
[1,] 4.436160e-07 3.306080e-08 0.0000719672 1.470676e-04 4
[2,] 3.864101e-08 3.210807e-05 0.9999113083 5.517886e-05 3
...
> # access the model performance
> model %>%
+   evaluate(mnist$test$x, mnist$test$y, verbose = 0)
$loss
[1] 0.08969676

$accuracy
[1] 0.9719
```



範例: Basic Image Classification: Fashion MNIST dataset

This guide uses the Fashion MNIST dataset which contains 70,000 grayscale images in 10 categories. The images show individual articles of clothing at low resolution (28 by 28 pixels), as seen here:



Fashion MNIST is intended as a drop-in replacement for the **classic MNIST** dataset—often used as the "Hello, World" of machine learning programs for computer vision. The MNIST dataset contains images of handwritten digits (0, 1, 2, etc) in an identical format to the articles of clothing we'll use here.

練習範例經驗: 同個例子有時順利成功，有時錯誤發生~

https://tensorflow.rstudio.com/tutorials/beginners/basic-ml/tutorial_basic_classification/

範例: Training a simple CNN to classify CIFAR images

TensorFlow for R from RStudio

Home | Installation | Tutorials | Guide | Deploy | Tools | API

- Overview

Beginners

- Quickstart
- Basic ML with Keras
- Load and preprocess data

Advanced

- Quickstart
- Customization
- Images
  - Convolutional Neural Network
  - Transfer Learning with tfhub
- Structured data
- Distributed training

Convolutional Neural Network (CNN)

This tutorial demonstrates training a simple Convolutional Neural Network (CNN) to classify CIFAR images. Because this tutorial uses the Keras Sequential API, creating and training our model will take just a few lines of code.

Setup

```
library(tensorflow)
library(keras)
```

Download and prepare the CIFAR10 dataset

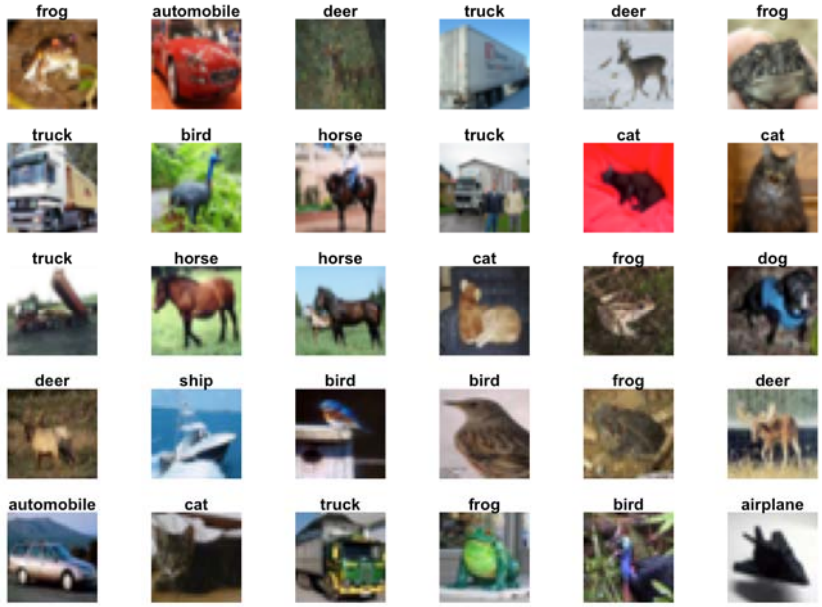
The CIFAR10 dataset contains 60,000 color images in 10 classes, with 6,000 images in each class. The dataset is divided into 50,000 training images and 10,000 testing images. The classes are mutually exclusive and there is no overlap between them.

```
cifar <- dataset_cifar10()
```

Verify the data

To verify that the dataset looks correct, let's plot the first 25 ir
name below each image.

```
class_names <- c('airplane', 'automobile', 'bird
               'dog', 'frog', 'horse', 'ship',

index <- 1:30
```



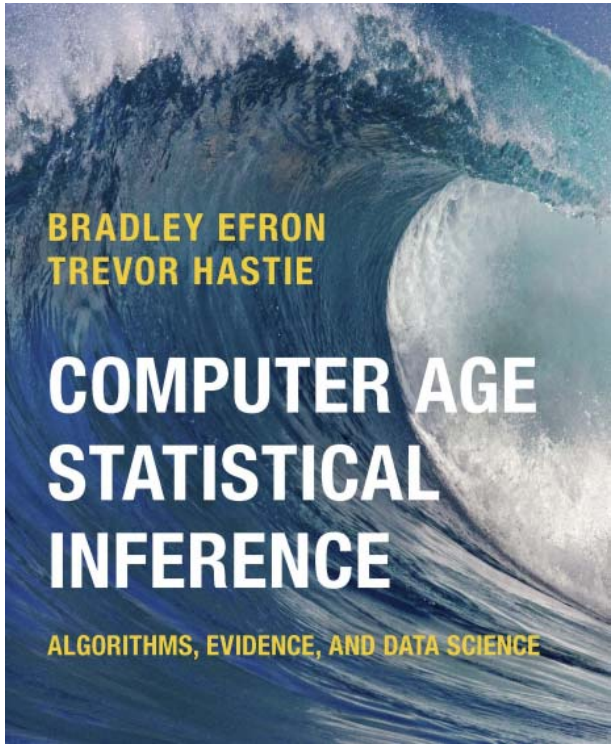
歸納/展望/QA

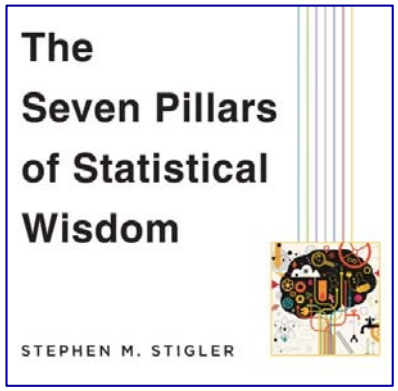
- 機器學習是資料科學的核心，是現代人工智慧的本質，其通用的架構為
  (1) 資料、(2) 模型、(3) 最佳化和(4)求解。

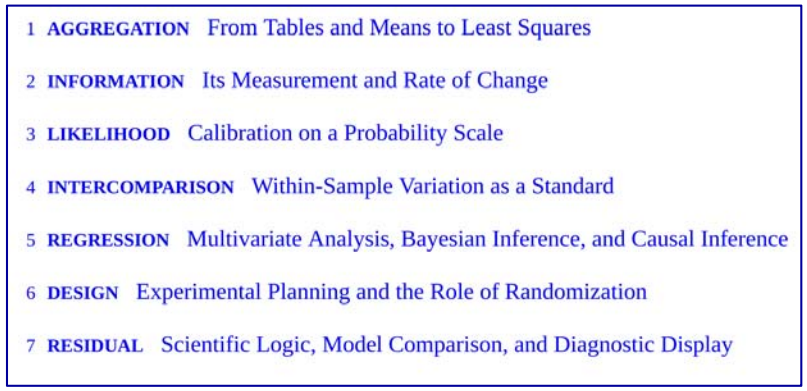
- 大數據、雲端運算、深度學習使得人工智慧在諸多領域得到突破性的進展。

- 給統計背景的學生: 技術方法學不完，該學(培養)些什麼?

learning from data，making sense out of data







https://web.stanford.edu/~hastie/CASI/