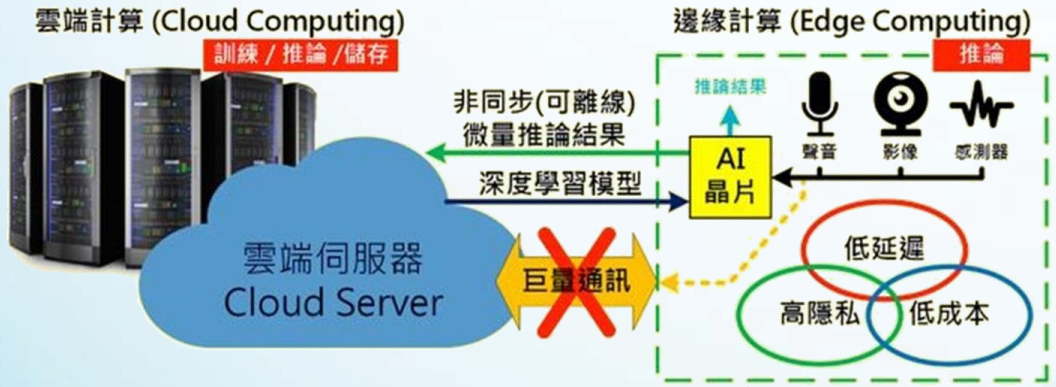
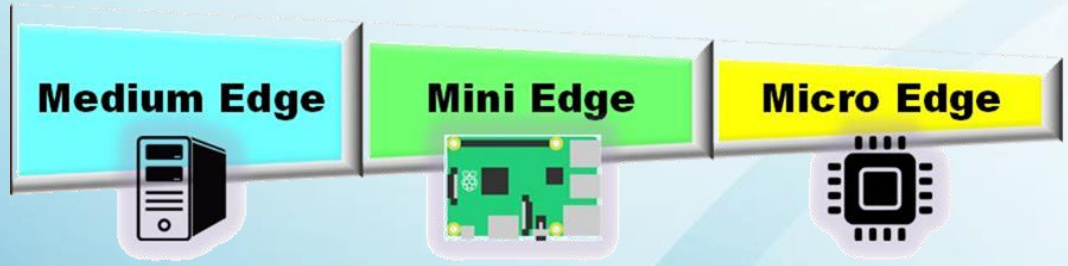
OmniXRI's Edge AI & TinyML 小學堂





【第6講】
模型優化與佈署

歐尼克斯實境互動工作室 (OmniXRI Studio)
許哲豪 (Jack Hsu)

簡報大綱

- 6.1. 模型訓練優化
- 6.2. 加速訓練方式
- 6.3. 模型推論優化

本課程完全免費，請勿移作商業用途！
歡迎留言、訂閱、點讚、轉發，讓更多需要的朋友也能一起學習。

完整課程大綱： https://omnixri.blogspot.com/2024/02/omnixris-edge-ai-tinyml-0.html
課程直播清單： https://www.youtube.com/@omnixri1784/streams

6.1.
模型訓練優化

- 數值擬合
- 輸出型式
- 損失函數
- 反向傳播
- 梯度下降
- 慣性動量
- 隨機丟棄

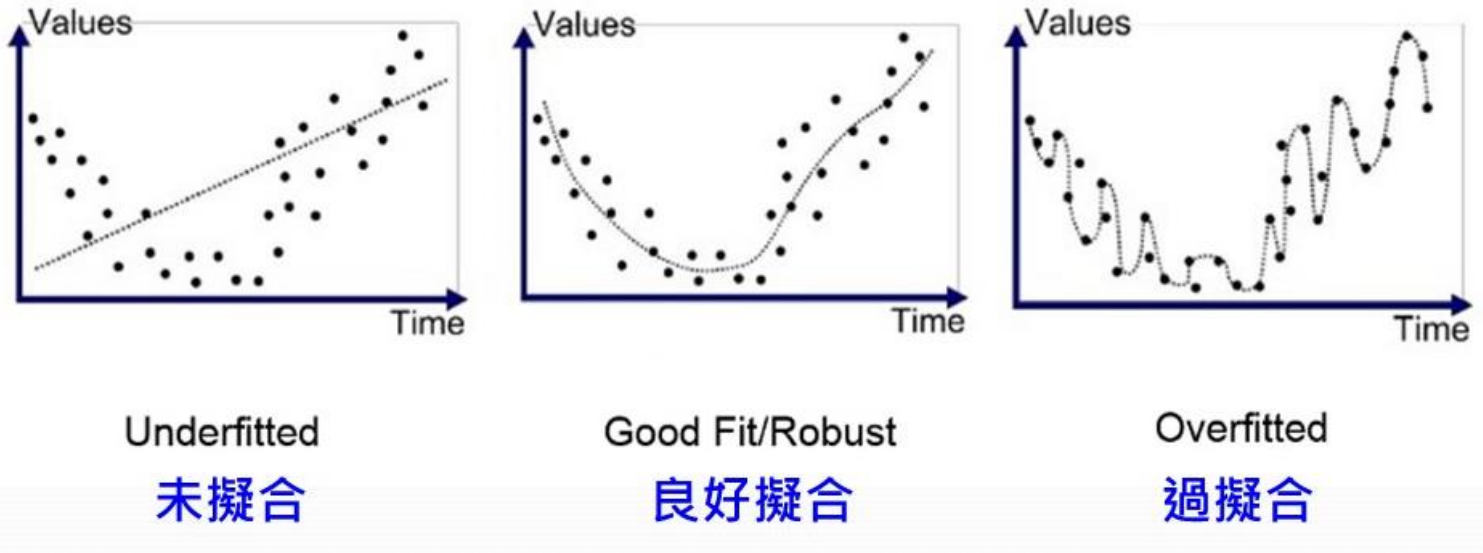
數值擬合

$y = f(x)$

$f(x) = ax + b$ (直線)

$f(x) = ax^2 + bx + c$ (二次曲線)

$f(x) = ax^n + bx^{n-1} + \ldots + c$ (n次曲線)



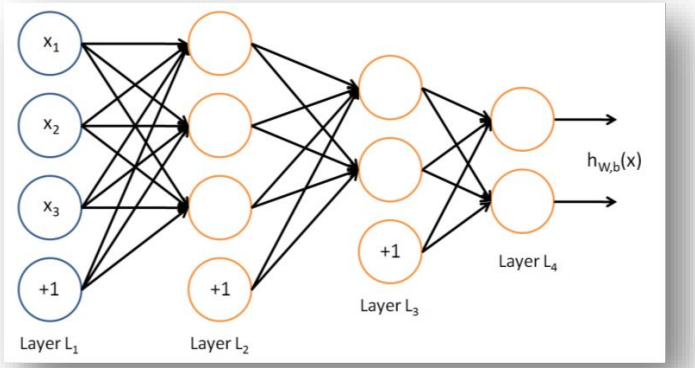
輸出型式 – Ordinal / One-Hot

序數編碼
（0~N類）

Ordinal Encoding

	boro	vegan
0	Manhattan	No
1	Queens	No
2	Manhattan	No
3	Brooklyn	Yes
4	Brooklyn	Yes
5	Bronx	No

	boro	boro_ordinal	vegan
0	Manhattan	2	No
1	Queens	3	No
2	Manhattan	2	No
3	Brooklyn	1	Yes
4	Brooklyn	1	Yes
5	Bronx	0	No



One-Hot Encoding

	vegan	boro_Bronx	boro_Brooklyn	boro_Manhattan	boro_Queens
0	No	0	0	1	0
1	No	0	0	0	1
2	No	0	0	1	0
3	Yes	0	1	0	0
4	Yes	0	1	0	0
5	No	1	0	0	0

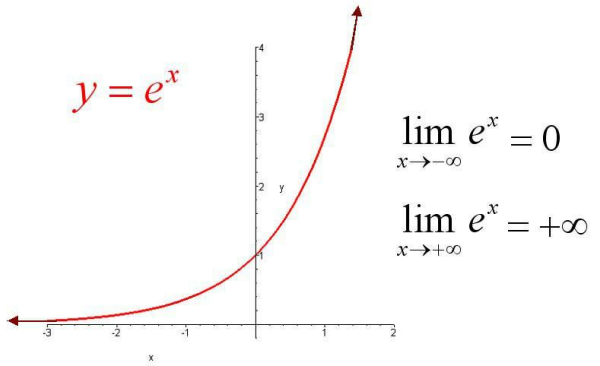
獨熱編碼
（只有一個1）

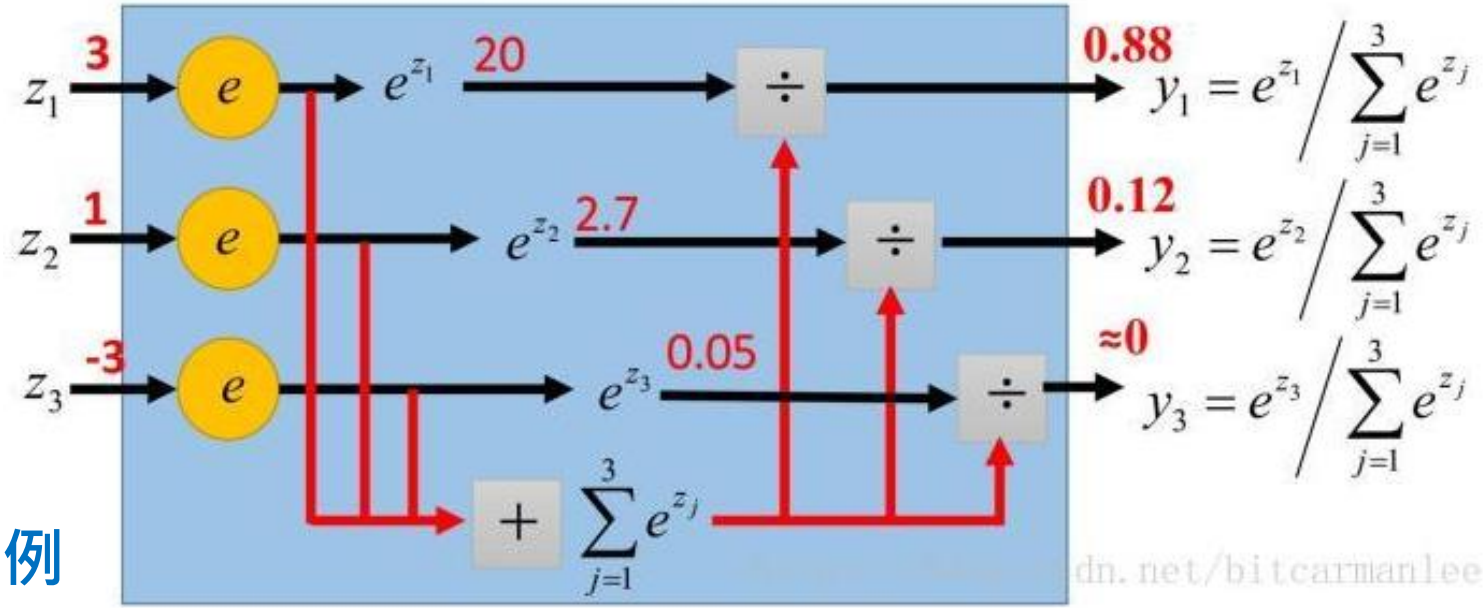
資料來源：https://axk51013.medium.com/%E4%B8%8D%E8%A6%81%E5%86%8D%E5%81%9Aone-hot-encoding-b5126d3f8a63

輸出型式 – Softmax

- 歸一化指數函數
- 輸出值在0.0~1.0間
- 所有輸出加總為1.0

$$S_i = \frac{e^i}{\sum_j e^j}$$





以三分類為例

損失函數

損失函數 = 實際值和預測值的殘差，亦可視為

實際值 = 預測值 + 誤差值，期望 $loss$ 越小越好

$$loss = y - \hat{y} \Rightarrow y = \hat{y} + \epsilon$$

常見誤差度量方式

- 平均絕對誤差 **MAE**
- 均方誤差 **MSE**
- 均方根誤差 **RMSE**
- 交叉熵 **Cross-Entropy**

損失函數 - MAE

MAE (Mean-Absolute Error, L1 Loss)

$$MAE = \frac{1}{n}\sum \left| y - \hat{y} \right|$$

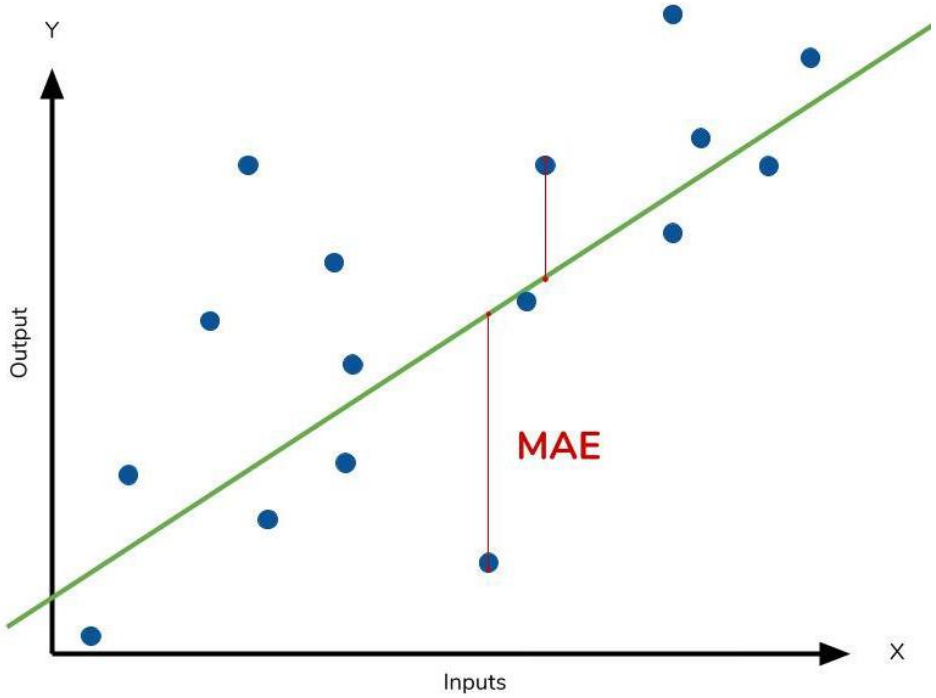
Divide by the total number of data points

Predicted output value

Actual output value

Sum of

The absolute value of the residual



資料來源：https://www.dataquest.io/blog/understanding-regression-error-metrics/

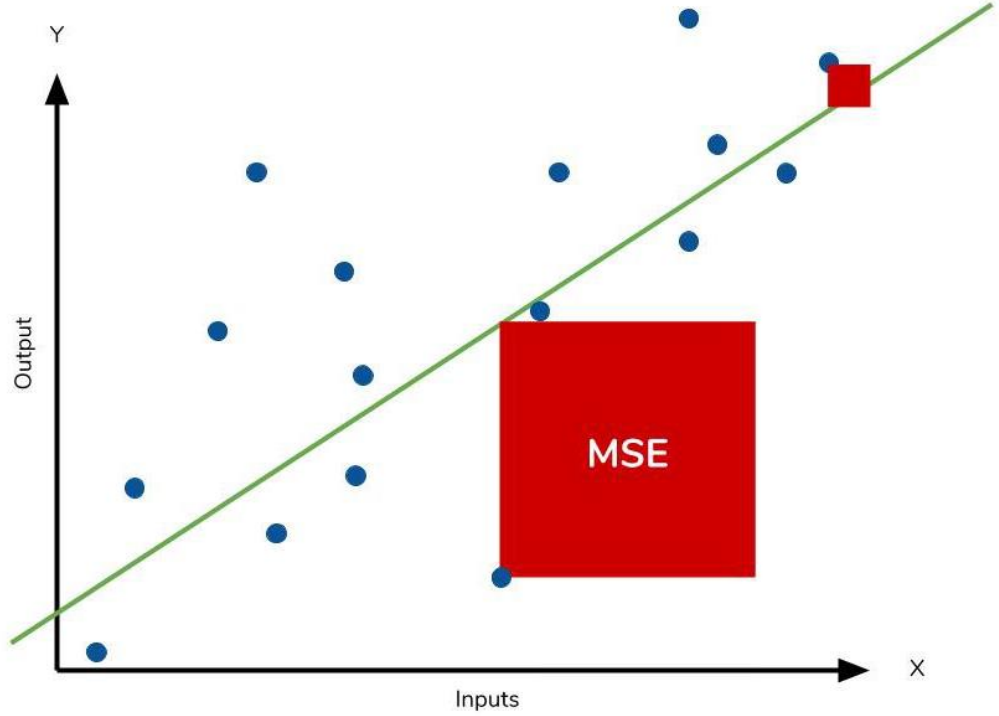
損失函數 – MSE / RMSE

MSE (Mean-Square Error, L2 Loss)

RMSE (Root-Mean-Square Error)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}$$



資料來源：https://www.dataquest.io/blog/understanding-regression-error-metrics/

損失函數 - Cross-Entropy

交叉熵(cross-entropy)

- 熵，即亂度，資訊越混亂時熵值越高。
- 分類問題，交叉熵值越小越有序，即越接近目標。

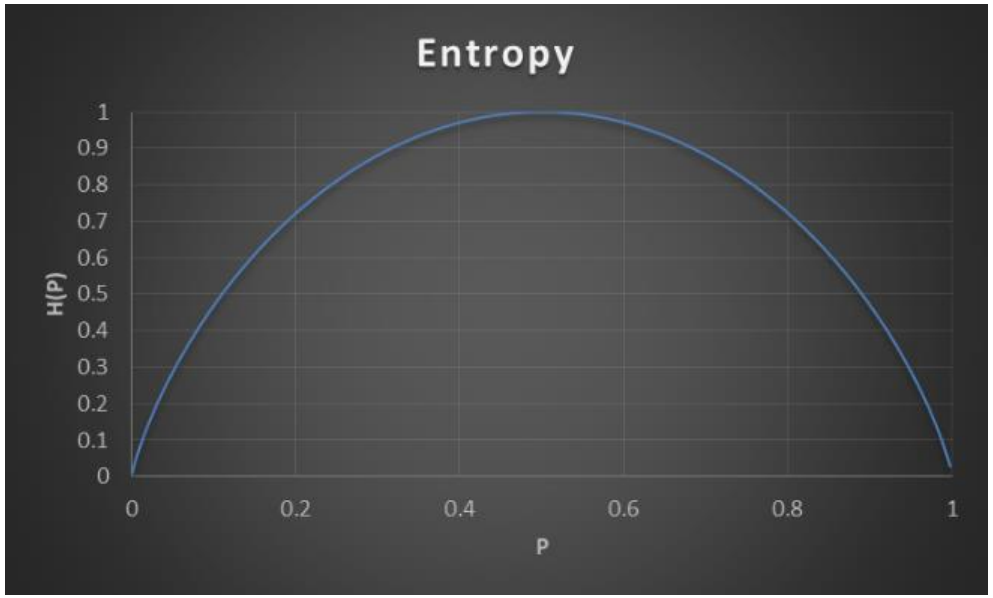
熵

$$H(X) = \sum_{i} -p_i log_2(p_i)$$

交叉熵

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

類別 資料數

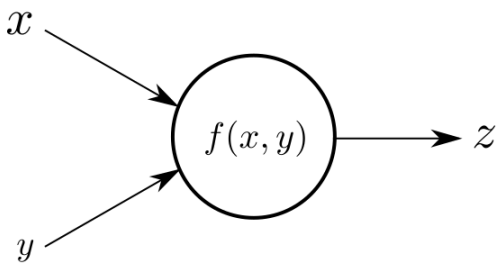


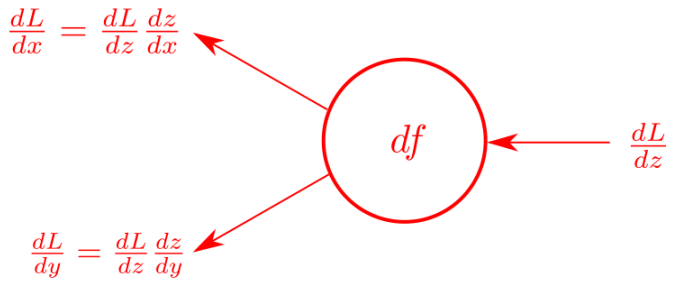
反向傳播

Forwardpass

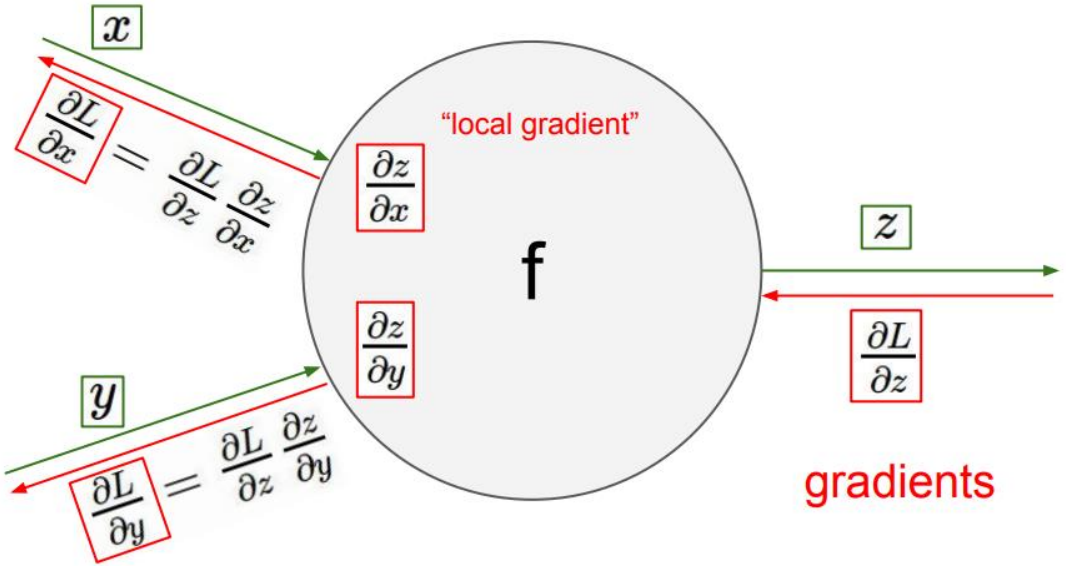
Backwardpass

正向傳播
（推論）

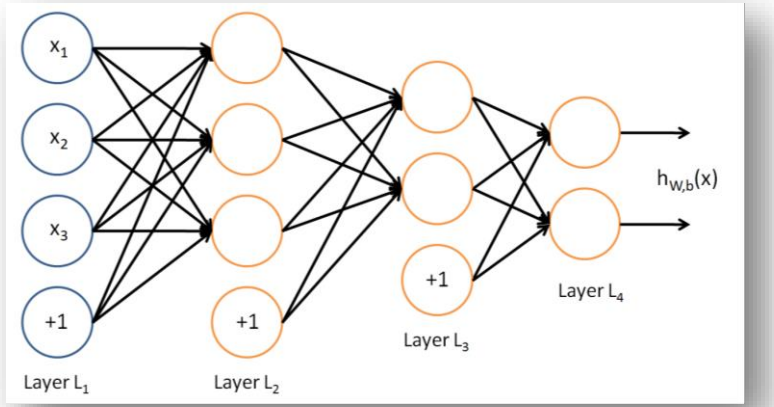




反向傳播
（訓練）



根據損失函數值反向調整
每個權重值，直到收斂。



資料來源：https://qingfenghcy.github.io/2018/04/26/%E5%8D%B7%E7%A7%AF%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C/

梯度下降 — 極值、梯度

- 微分找極值，一階微分為零有**極值**。
- 二階微分大於零有**極小值**，反之為**極大值**。

$$f(x) \qquad f'(x) = \frac{\partial f(x)}{\partial x} \qquad f''(x) = \frac{\partial f'(x)}{\partial x}$$

- 多維向量找極值即對所有元素進行偏微分，求得「**梯度**」

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}, \nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_d} \end{bmatrix}$$

資料來源：https://chih-sheng-huang821.medium.com/

梯度下降 – 學習率

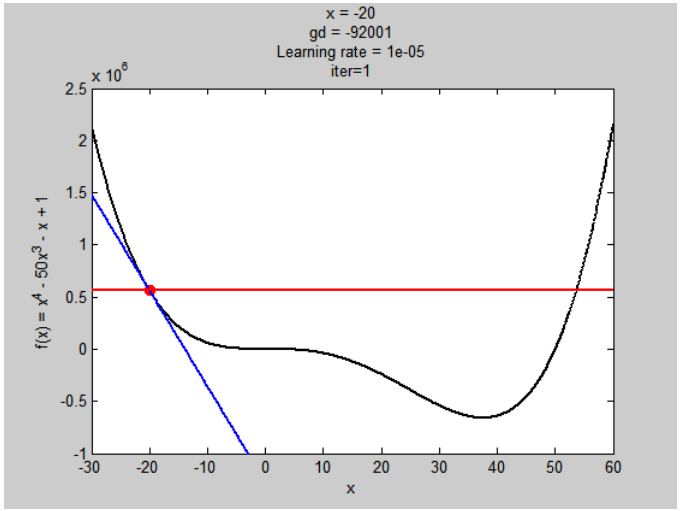
梯度下降(Gradient Descent)

是一種不斷向梯度方向更新參數找解的方法

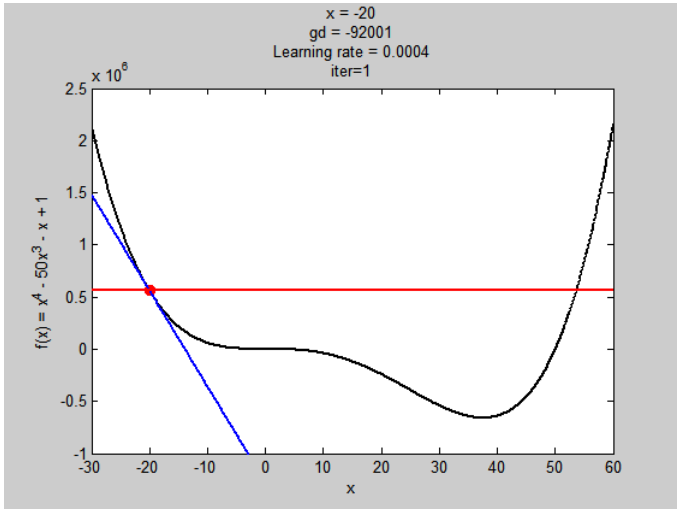
其中 t 為第幾次更新， r 為學習率。 $x^{(t+1)} = x^{(t)} - \gamma \nabla f\left(x^{(t)}\right)$

固定學習率（超參數）

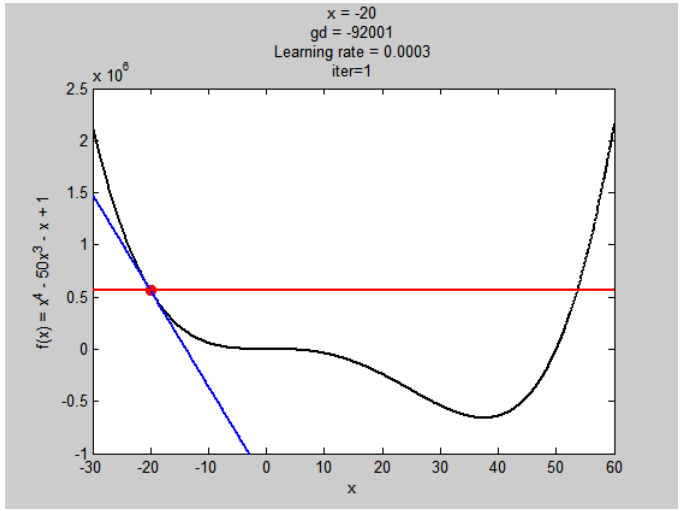
學習率過小
易陷入區域極小值



學習率過大
易產生振盪



學習率適中
快速收斂到最小值



資料來源：https://chih-sheng-huang821.medium.com/

梯度下降 – 常見優化器調整方式

$$w_{\text{new}} = w - \alpha \frac{\partial L}{\partial w}$$

學習率(α)　梯度(∇)

Optimiser	Year	Learning Rate	Gradient
Momentum	1964		✓
AdaGrad	2011	✓	
RMSprop	2012	✓	
Adadelta	2012	✓	
Nesterov	2013		✓
Adam	2014	✓	✓
AdaMax	2015	✓	✓
Nadam	2015	✓	✓
AMSGrad	2018	✓	✓

資料來源：https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9

梯度下降 – BGD / SGD / MBGD

批量梯度下降法(Batch Gradient Descent)

- 全部樣本計算平均損失函數後再求梯度進行下降更新。

隨機梯度下降法(Stochastic Gradient Descent)

- 每個樣本求得損失函數就進行更新，由於樣本是隨機的，所以就稱為隨機梯度下降法。

小批量梯度下降法(Mini-Batch Gradient Descent)

- 搭配指定數量樣本(Batch Size)計算平均損失函數後再求梯度進行下降更新。

缺點：很難找到一個適合的固定學習率或批量大小來滿足各種情況。

資料來源：https://chih-sheng-huang821.medium.com/

梯度下降 – Adagrad

自適應梯度（自動學習率調整）法：Adagrad

- 隨時間令學習率逐漸自動變小，以減少振盪，快速收斂。

$$\eta^t = \frac{\eta}{\sqrt{t+1}} \qquad g^t = \frac{\partial L(\theta^t)}{\partial w}$$

Adagrad

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t$$

σ^t: ***root mean square*** of the previous derivatives of parameter w

Parameter dependent

資料來源：https://medium.com/chung-yi

梯度下降 – RMSProp

均方根傳播法(Root-Mean-Square Propagation)

- 如同Ardgrad除了考量時間因素令學習率逐漸衰減外，另外RMSProp再加入衰減因子α。

$$w^1 \leftarrow w^0 - \frac{\eta}{\sigma^0} g^0 \qquad \sigma^0 = g^0$$

$$w^2 \leftarrow w^1 - \frac{\eta}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\alpha(\sigma^0)^2 + (1-\alpha)(g^1)^2}$$

$$w^3 \leftarrow w^2 - \frac{\eta}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\alpha(\sigma^1)^2 + (1-\alpha)(g^2)^2}$$

$$\vdots$$

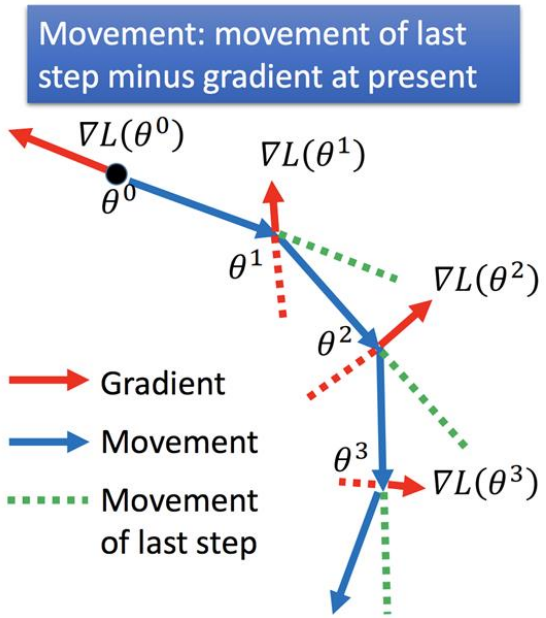
$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1-\alpha)(g^t)^2}$$

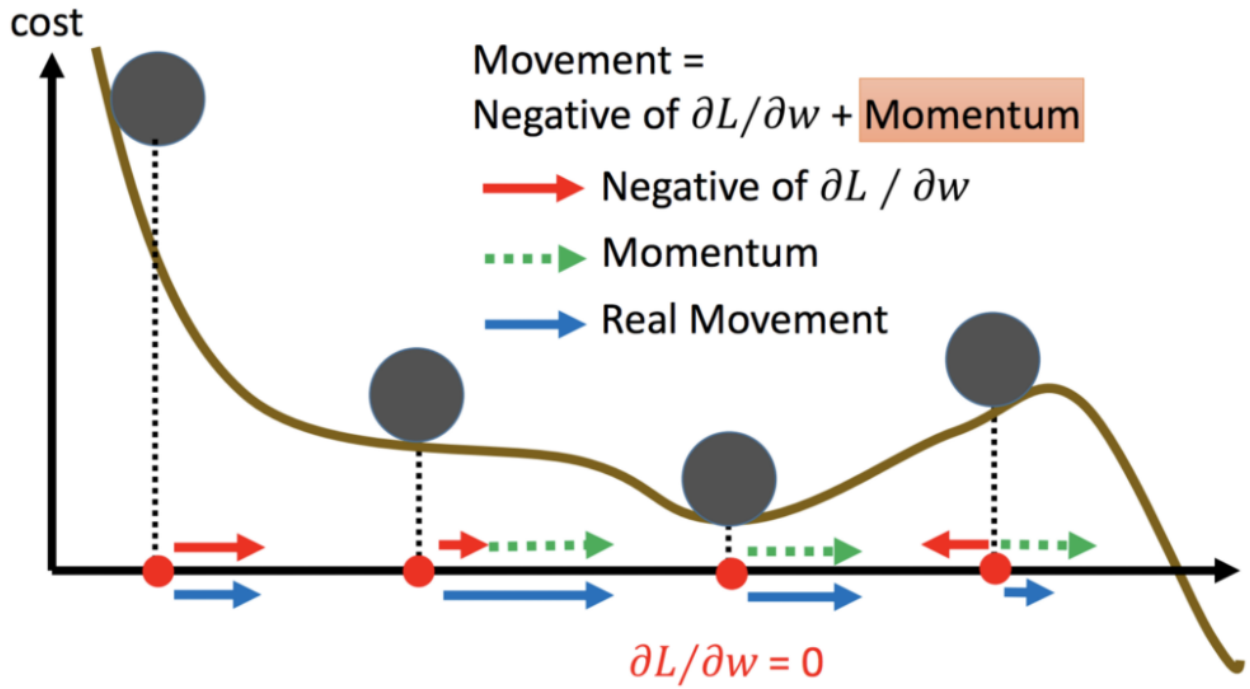
資料來源：https://medium.com/chung-yi

慣性動量 - Momentum

動量(Momentum)

- 單純梯度下降容易陷入區域最小值。
- 利用物理慣性概念來修正運動方向，即**梯度**和**動量**組成的新向量作為最後移動方向及大小。





資料來源：https://medium.com/chung-yi

慣性動量 - Adam

自適應動量估測(Adaptive Moment Estimation)

- 相當是加了動量的RMSProp，並在梯度更新時考慮偏差校正(Bias-Correction)
- 結合Adagrad和RMSProp優點，可處理高維及大資料集梯度差異較大的數據。

一階動量 $m_t = \beta_1 m_{t-1} + (1-\beta_1)\frac{\partial L_t}{\partial W_t}$

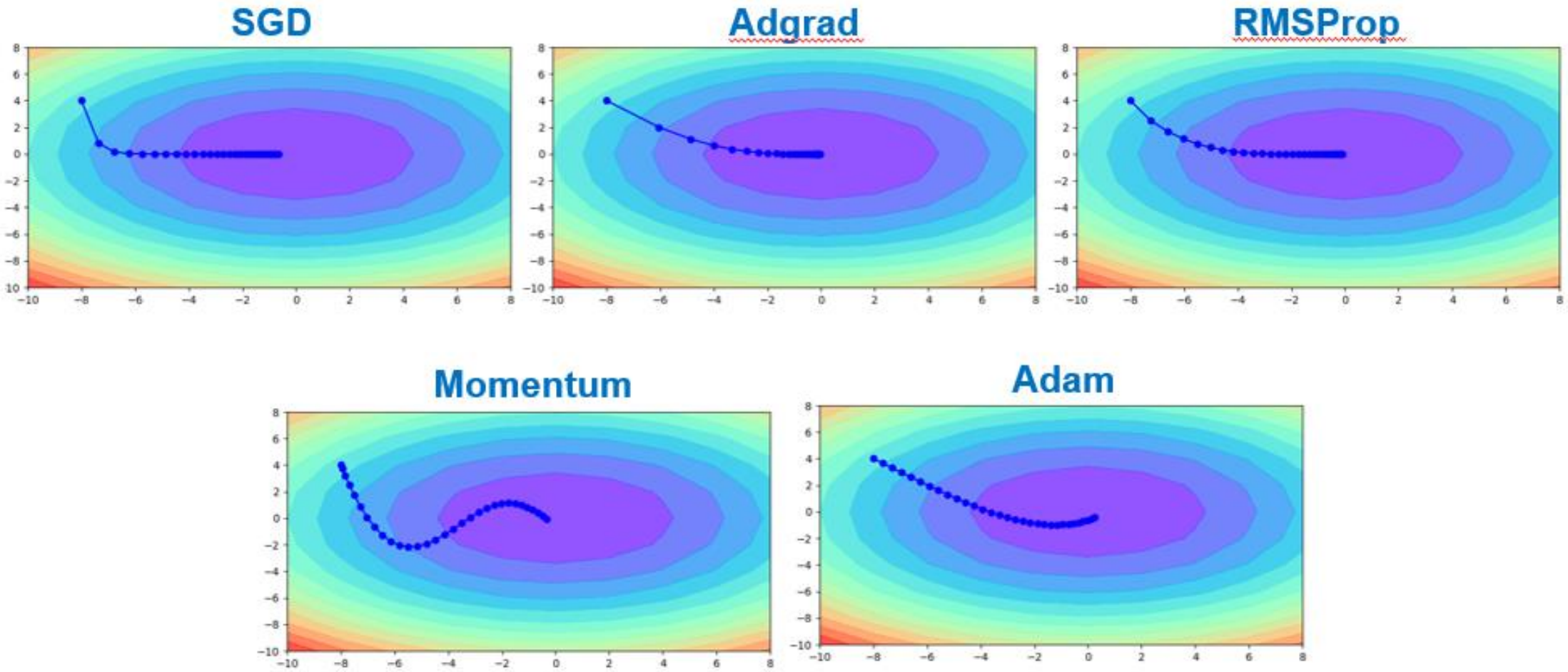
一階校正 $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$

二階動量 $v_t = \beta_1 v_{t-1} + (1-\beta_2)(\frac{\partial L_t}{\partial W_t})^2$

二階校正 $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$

資料來源：https://medium.com/chung-yi

慣性動量 - 實驗比較

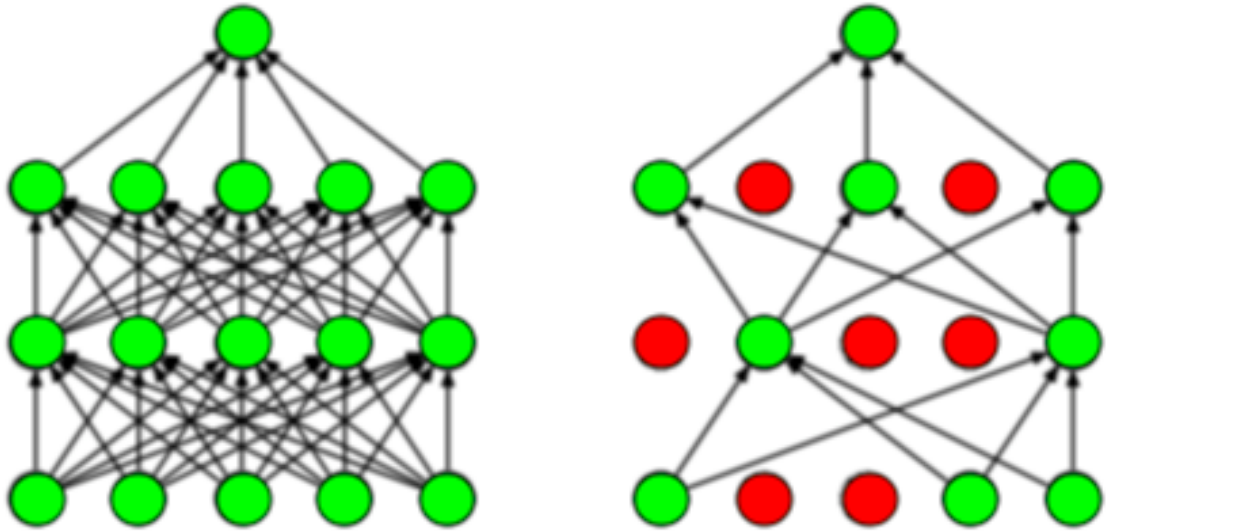


資料來源：https://medium.com/chung-yi

隨機丟棄

Dropout

- Google提出的一種正規化技術，用來防止小資料集過擬合。
- 在訓練階段隨機丟棄部份神經元，只更新其它權重。
- 可減少計算數量。



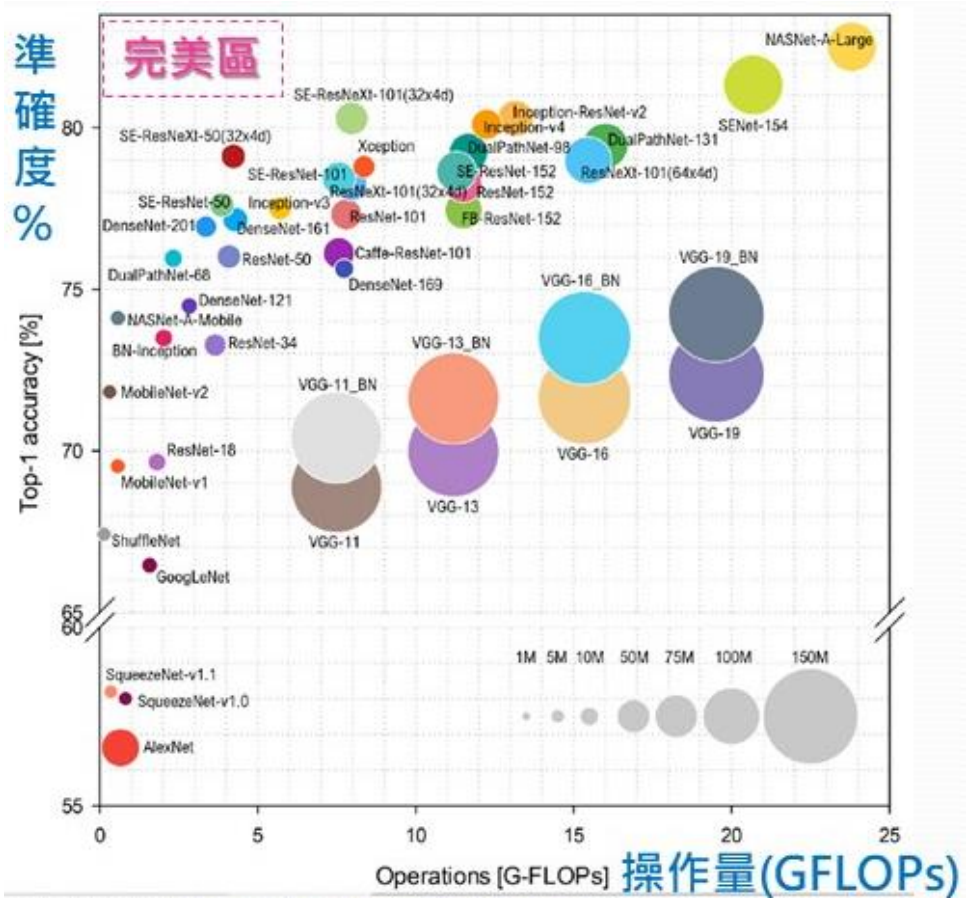
原始網路　　紅點為被隨機丟棄的神經元

6.2.
加速訓練方式

- 低複雜度模型
- 遷移式學習
- 自動學習
- 分散式學習

低複雜度模型

影像分類模型



重要評估指標

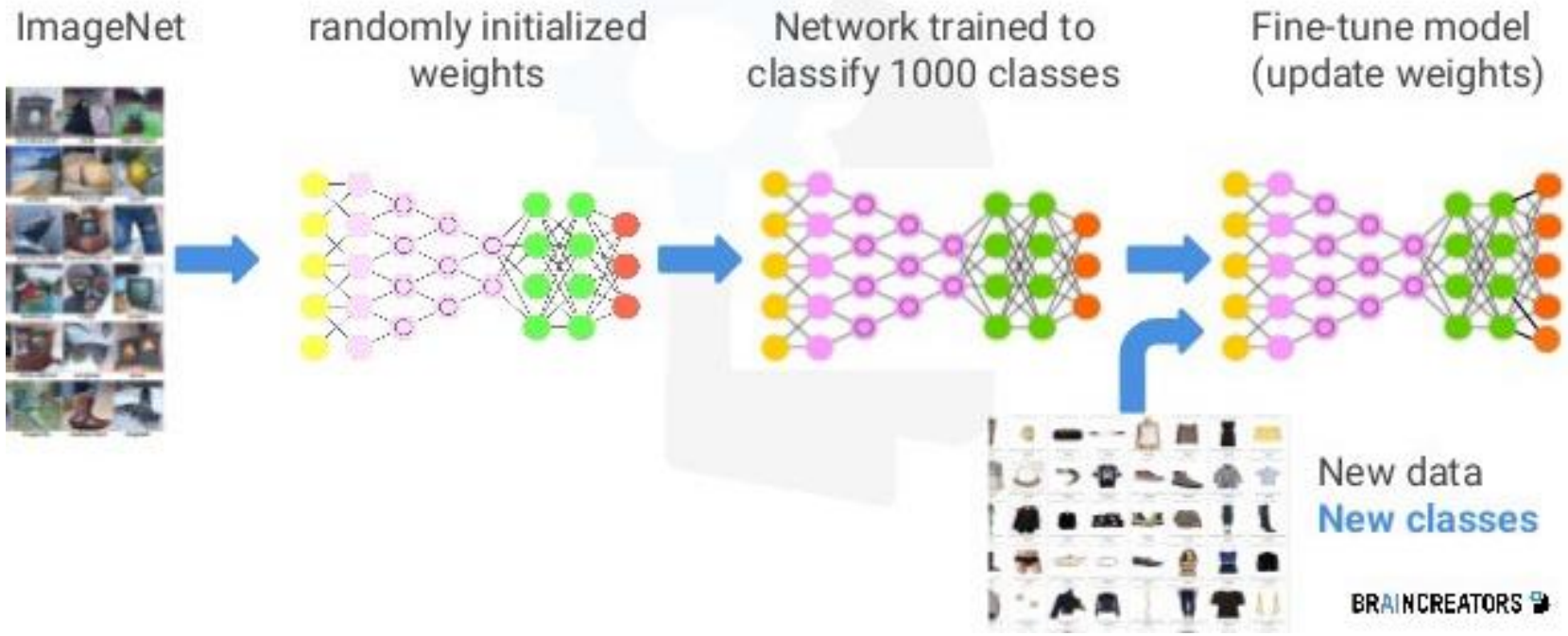
- 準確率(Accuracy)
- 記憶體使用量(Memory Footprint)
- 參數量(Parameter)
- 操作數(Operations Count)
- 推理時間(Inference Time)
- 功耗(Power Consumption)

模型小參數少不代表計算量少，計算少通常推論時間、功耗較小

遷移式學習

遷移式學習(Transfer Learning)

- 利用已訓練好的模型，拆解最後幾層，換上新的再重訓。
- 適用資料集不足或想加快訓練時間場合。

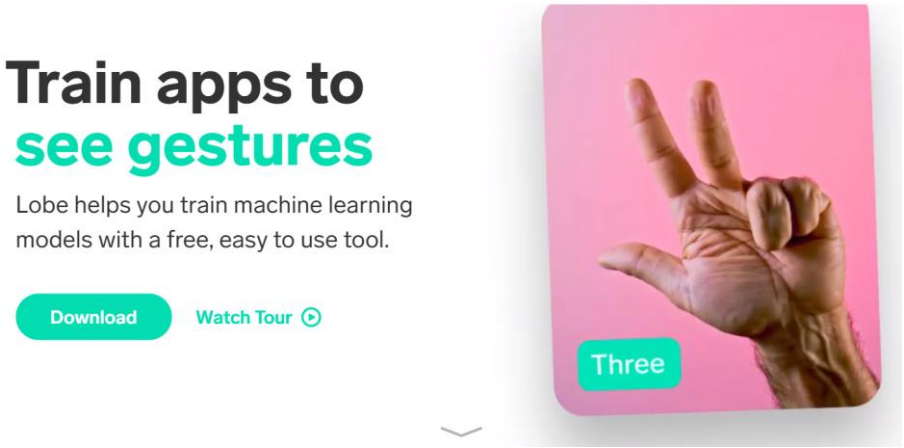


資料來源：https://madhuramiah.medium.com/deep-learning-using-resnets-for-transfer-learning-d7f4799fa863

遷移式學習 – 應用程式

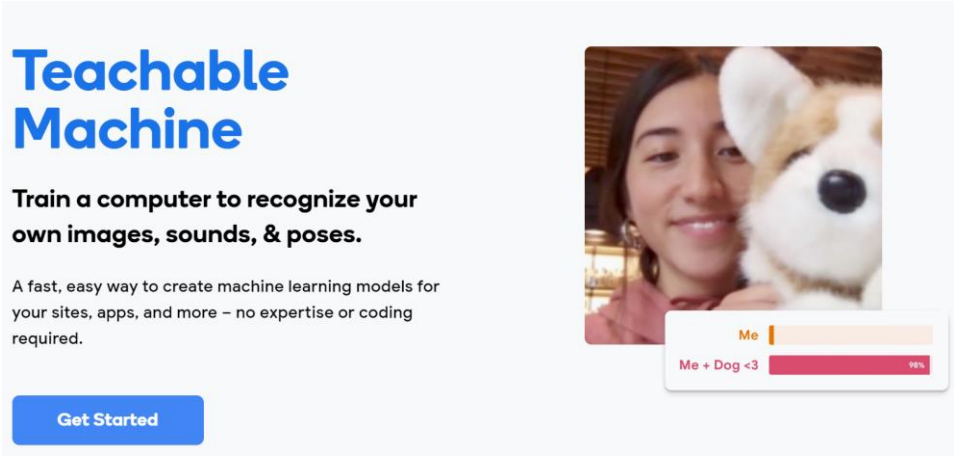
Microsoft Lobe.ai

- 資料收集標記
- 訓練模型
- 匯出模型
- 可離線進行影像分類、物件偵測及資料分類



Google Teachable Machine

- 擷取影像並分類
- 開始訓練
- 輸出及佈署
- 僅能線上進行影像、聲音、姿態分類



自動學習

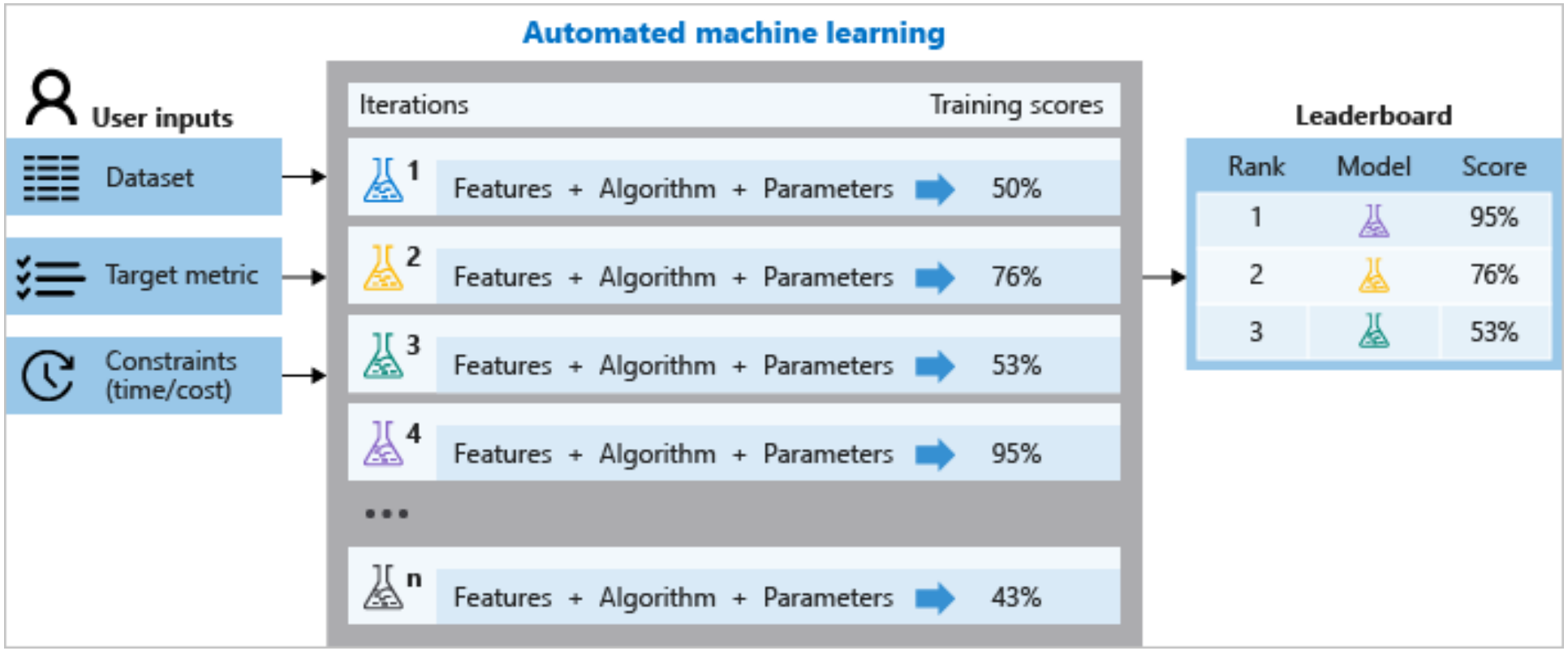
自動學習(AutoML)

- 使用者僅需準備已標註資料集
- 不需資料科學家介入、免程式開發、自動找出最合適算法
- 自動反覆調整參數使模型效率達到最高
- 適合迴歸分析、分類及時間預測等算法。

常見框架及平台

- Google GCP, Amazon AWS, Microsoft Azure付費式AutoML
- AutoKeras、 AutoSKlearn、Automl-gs
- Auto-Weka、FeatureTools、Ludwig
- MLBox、 TPOT、 H2O AutoML
- Neural Network Intelligence、TransmografAI

自動學習 – 基本框架



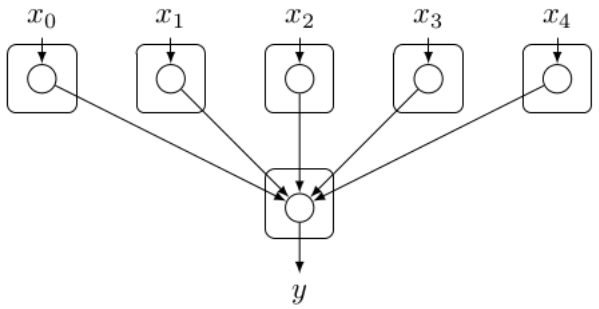
主要工作：資料預處理、特徵工程、特徵提取、特徵選擇、模型訓練、算法選擇、超參數優化

資料來源：https://docs.microsoft.com/zh-tw/azure/machine-learning/concept-automated-ml

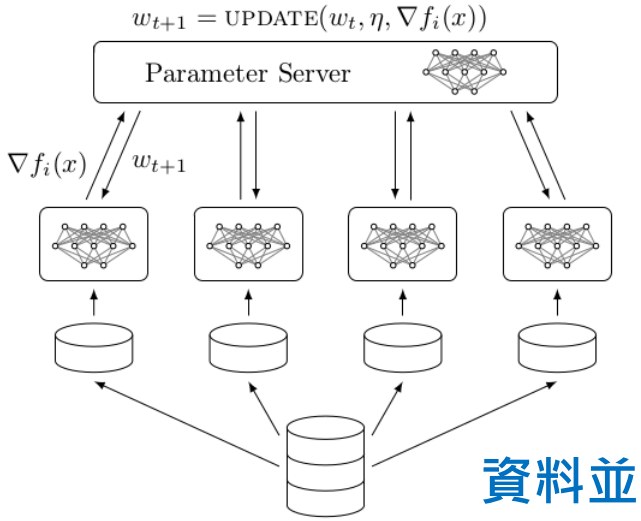
分散式學習 – 並行學習

分散式訓練/學習(Distributed Training / Learning)

- 模型並行(Model Parallelism)
- 資料並行(Data Parallelism)
- 引數平均(Model Averaging)
- 非同步隨機梯度下降(Asynchronous SGD)



模型並行



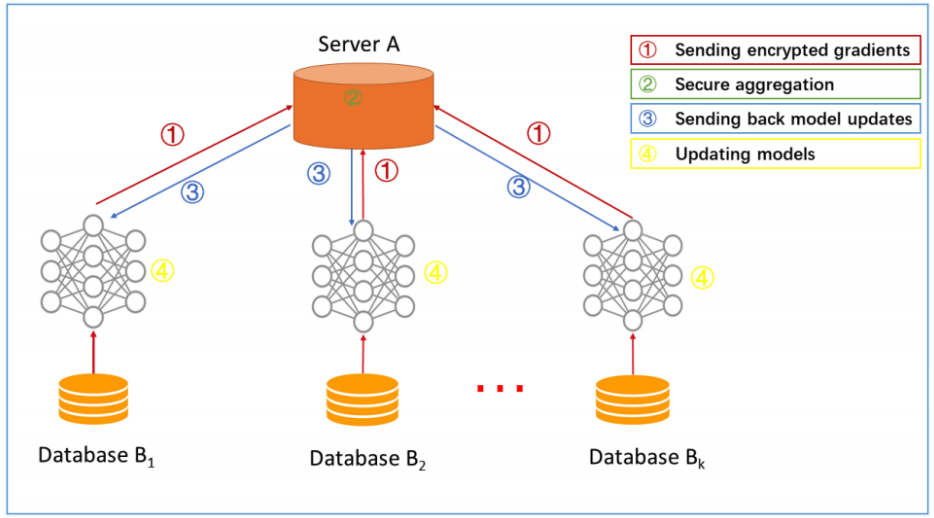
資料並行

資料來源：https://joerihermans.com/ramblings/distributed-deep-learning-part-1-an-introduction/

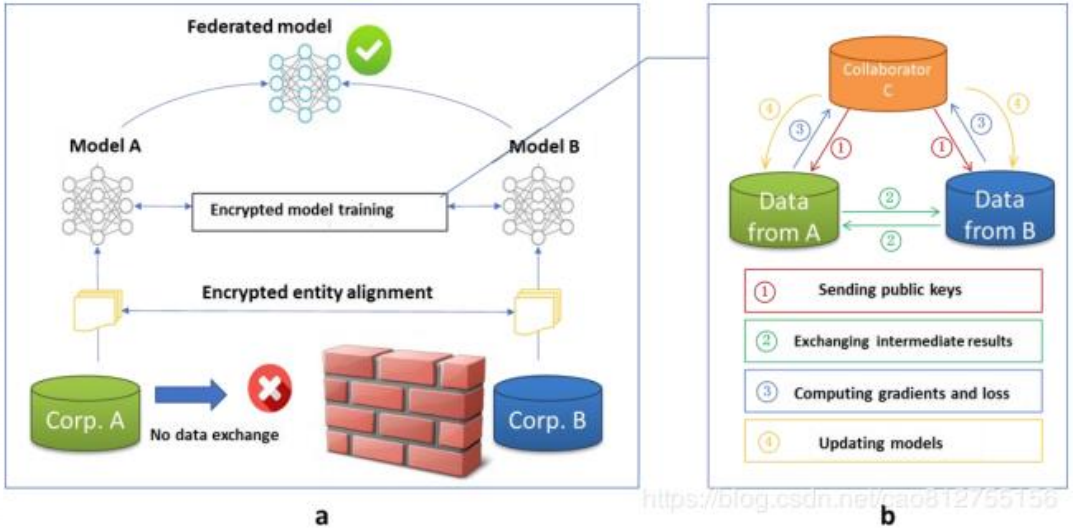
分散式學習 – 聯邦學習

聯合/聯邦/聯盟學習(Federated Learning)

- 資料共享訓練但不提供原始資料，具高隱私性。
- 模型各自訓練但共享訓練成果，再重新佈署調整參數。



水平式



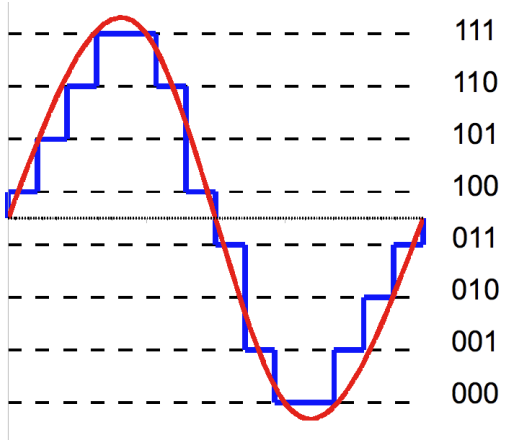
垂直式

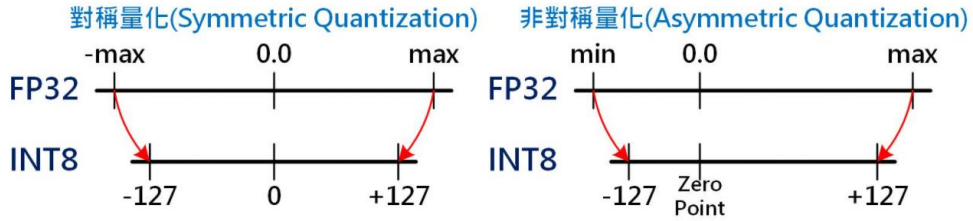
資料來源：https://medium.com/disassembly/architecture-of-federated-learning-a36905c1d225

6.3.
模型推論優化

- 數值量化
- 模型剪枝
- 模型壓縮
- 知識蒸餾

數值量化





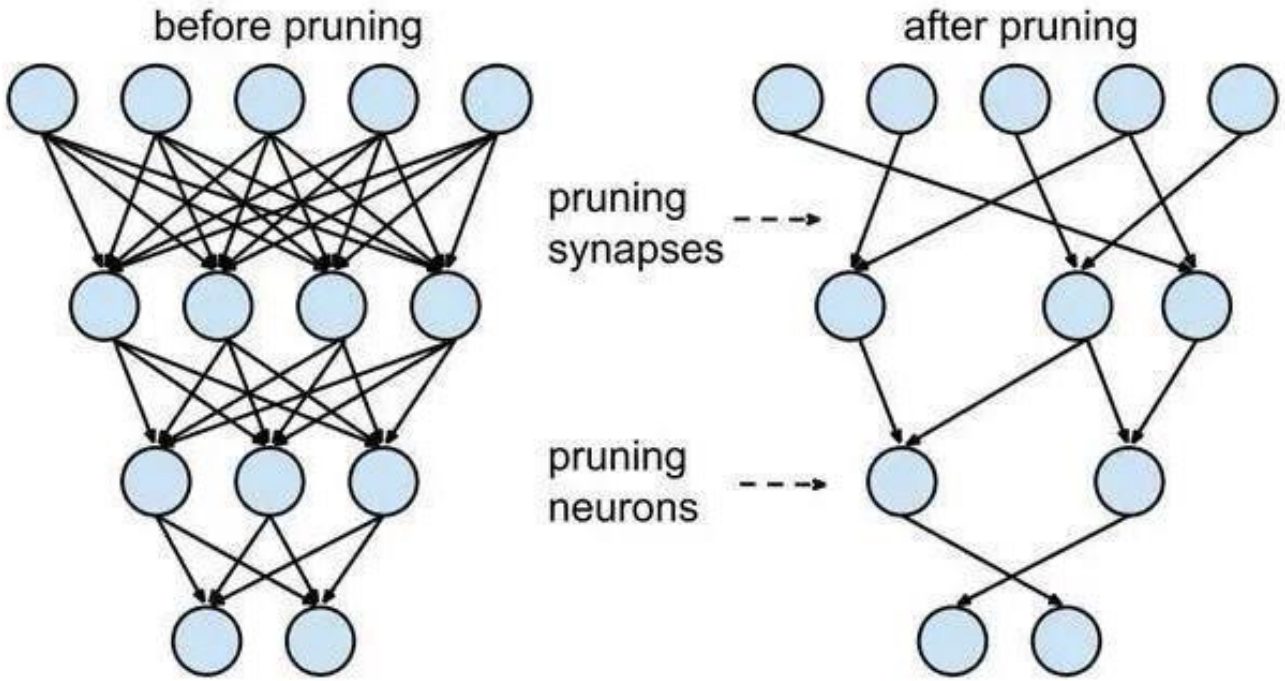
連續(類比)資料數位化

- 二值 (0/1)
- 整數 (INT16 / INT8 / INT4)
- 定點數 (Qf / Qm.f)
- 浮點數 (FP32 / FP16 / FP8, TF32 / BF16)

位數壓縮

- 對稱量化
- 非對稱量化
- **混合精度**

模型剪枝

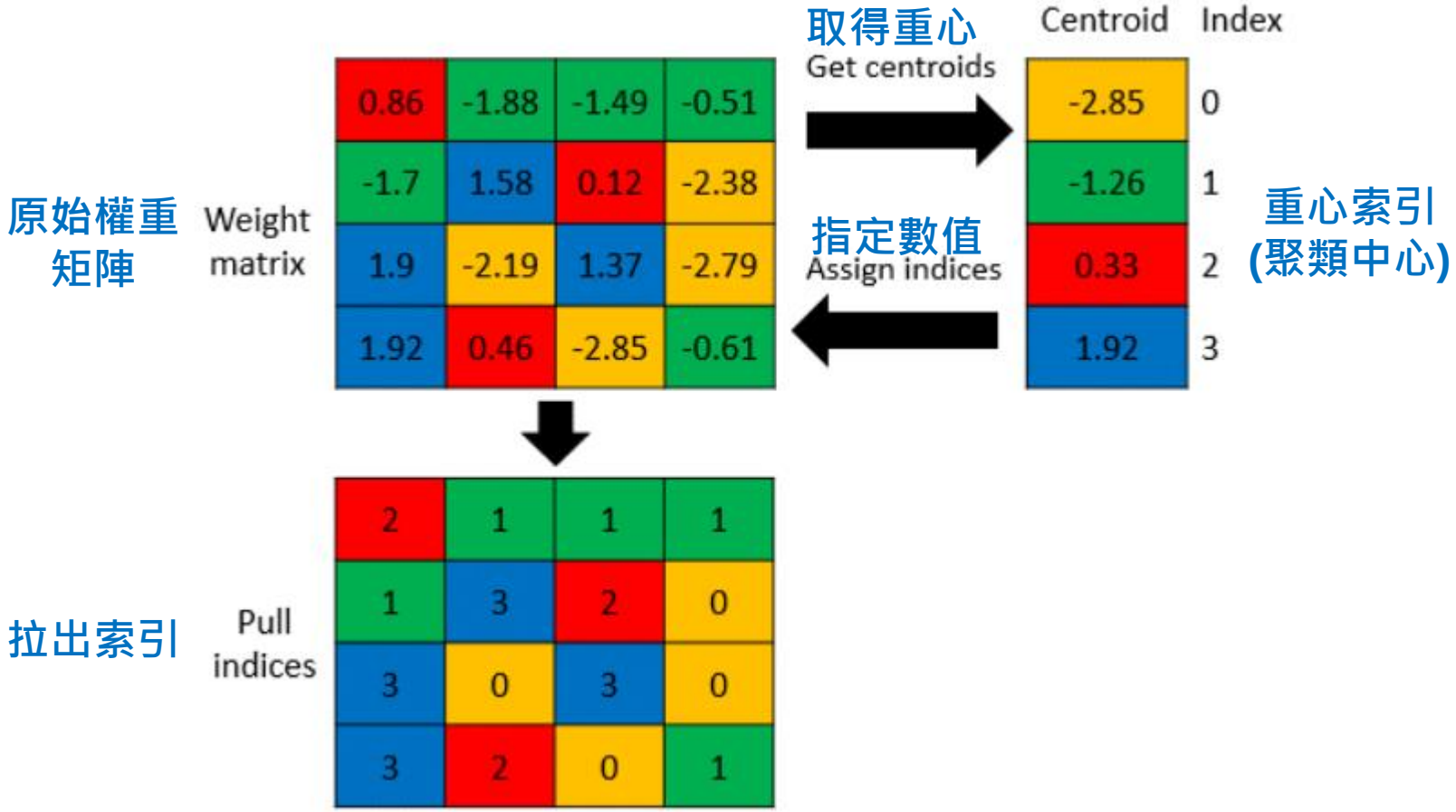


模型剪枝(Pruning)

- 刪除不重要權重神經元
- 直接指定刪除比例
- 不同層刪減不同比例
  - 接近輸入層很敏感
  - 接近輸出層不敏感
- 考慮硬體運算架構
- 迭代式 (Iterative)剪枝
  - 修剪
  - 訓練
  - 重複

模型壓縮 – 權重共享

權重共享(Weight Share / Clustering)

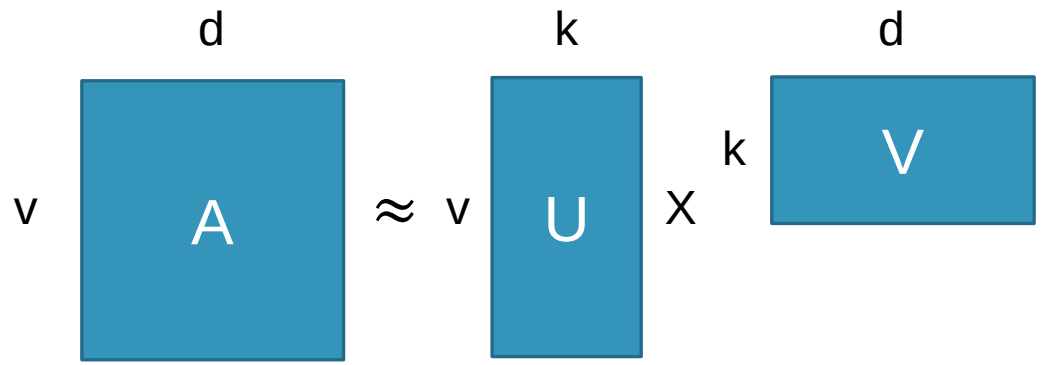


資料來源：https://blog.tensorflow.org/2020/08/tensorflow-model-optimization-toolkit-weight-clustering-api.html

模型壓縮 – 低秩逼近

低秩逼近(Low Rank Approximation)

- 去除冗餘資訊，並分解成較小矩陣相乘代替大矩陣相乘。
- 對CNN網路而言
  - 大部份參數來自於全連接層
  - 大部份計算來自於卷積層
  - 卷積時間大部份花費在前幾層
  - 若能有效處理卷積及全連接層可大幅壓縮模型



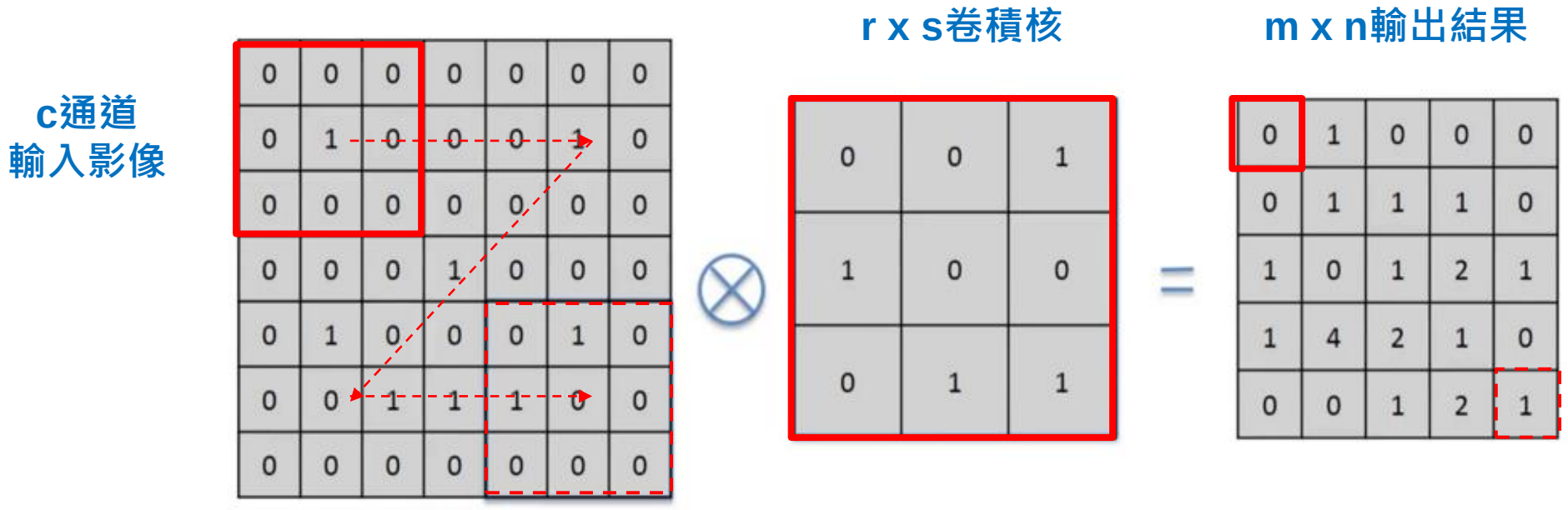
LeNet-5網路結構

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1 × 1	5 × 5	1	tanh
S4	Avg Pooling	16	5 × 5	2 × 2	2	tanh
C3	Convolution	16	10 × 10	5 × 5	1	tanh
S2	Avg Pooling	6	14 × 14	2 × 2	2	tanh
C1	Convolution	6	28 × 28	5 × 5	1	tanh
In	Input	1	32 × 32	–	–	–

模型壓縮 – 標準卷積

標準2D卷積法(Convolution)

- c = 1, r = 3, s = 3, m = 5, n = 5時需要
- r x s x m x n = 225 個**乘法**
- [(r x s)-1] x (m x n) = 200 個加法

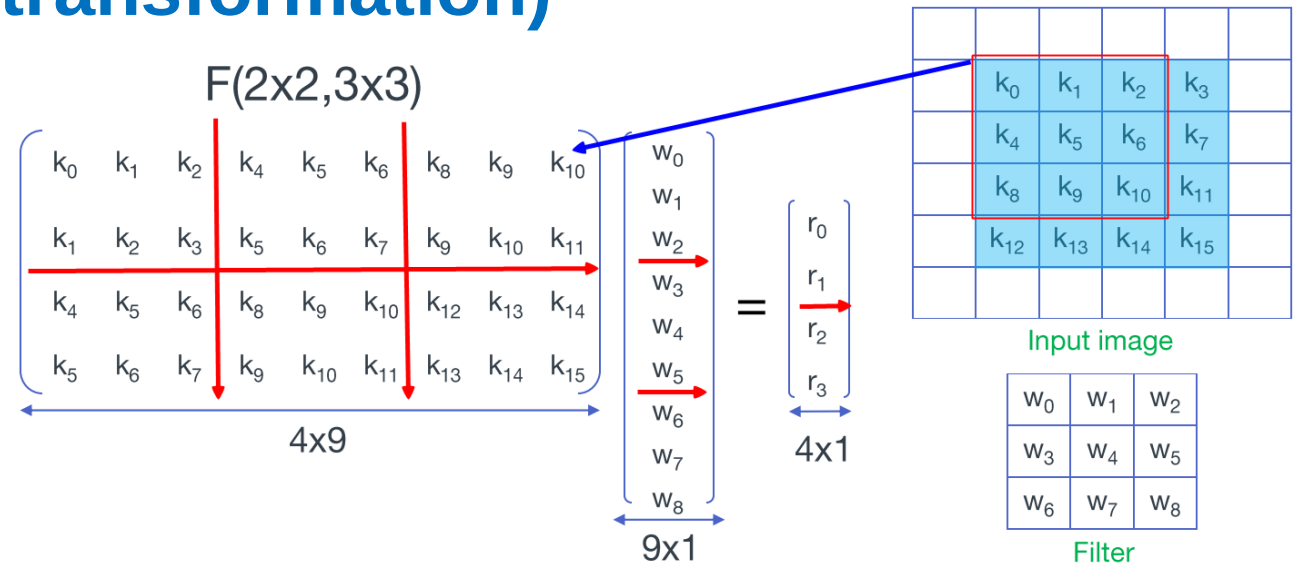


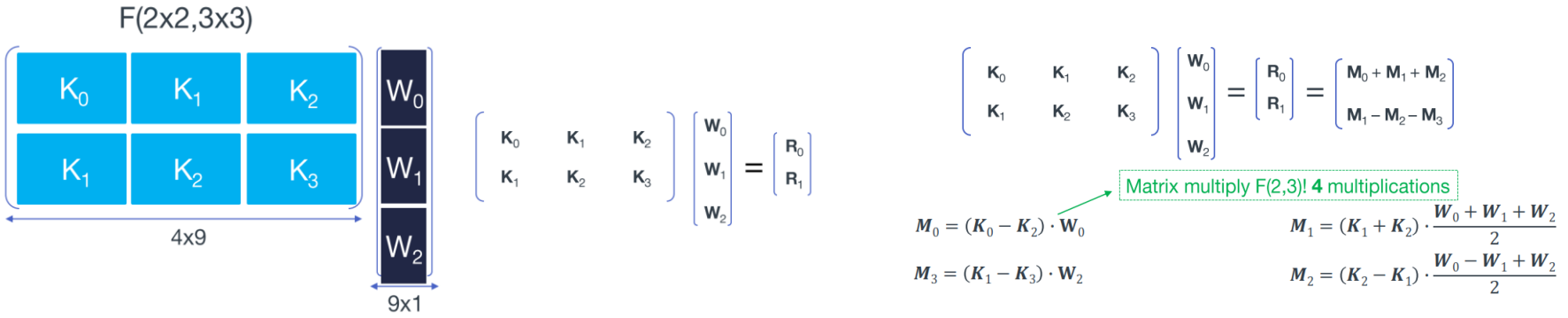
模型壓縮 – 快速卷積

快速卷積法(Winograd transformation)

- Winograd算法

  直接卷積36次乘法
  快速卷積16次乘法
  減少2.25倍次乘法
  可擴展至3維矩陣
  適合較小卷積核



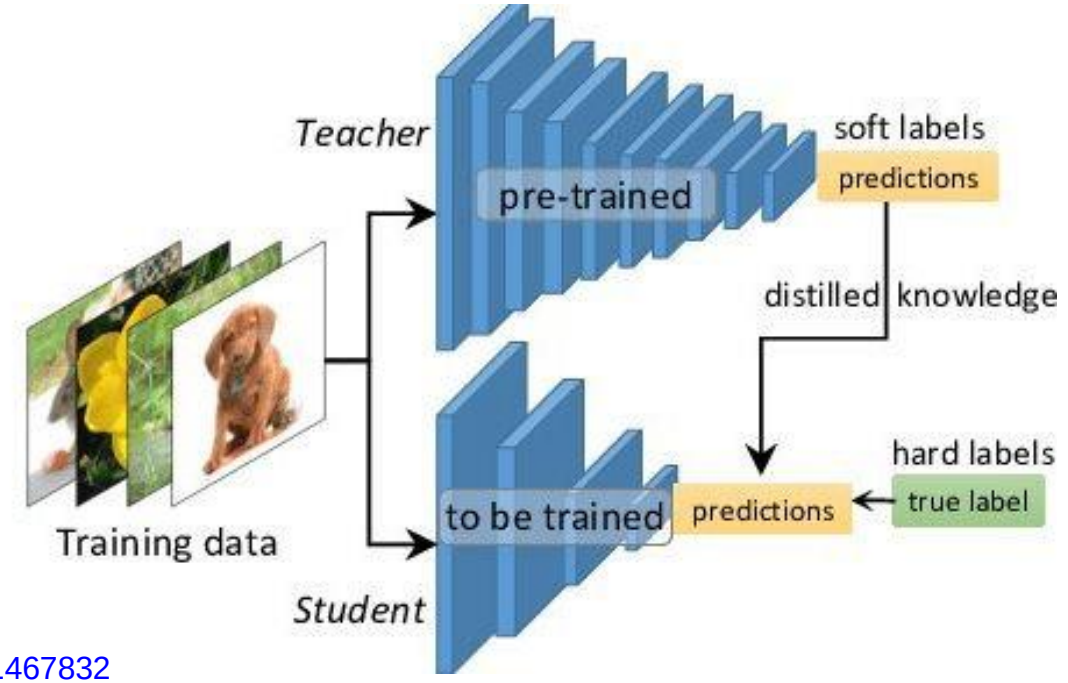


Fast Algorithms for Convolutional Neural Networks https://arxiv.org/abs/1509.09308

知識蒸餾

知識蒸餾(Knowledge Distillation)

- 具有老師和學生網路。老師網路具有複雜結構及高準確度推論能力。學生網路企圖以一簡單網路來學習老師網路輸出結果，如此便將重要資訊保留下來，同時縮小模型，加速推論速度。



資料來源：https://zhuanlan.zhihu.com/p/81467832

小結

模型訓練優化

了解損失函數定義，並學習梯度下降，慣性動量及隨機丟棄如何提升訓練成果及快速收斂。

加速訓練方式

選擇低複雜度模型，或透過遷移式學習快速訓練，並可利用自動學習及分散式學習來增加大規模訓練效率。

模型推論優化

透過數值量化、模型剪枝、模型壓縮和知識蒸餾可快速從已訓練好模型中得到更精簡模型並保持一定推論準確率。

參考文獻

- 許哲豪，臺灣科技大學資訊工程系「人工智慧與邊緣運算實務」（2021~2023）

https://omnixri.blogspot.com/p/ntust-edge-ai.html

延伸閱讀

- 陽明交通大學電子工程學系張添烜老師，「深度學習的模型壓縮與加速」(YouTube)

https://www.youtube.com/watch?v=LlJQ_agQJyM&list=PLj6E8qlqmkFv3cCjjX2SA1D4FJ9fadDij

- MIT Han Lab (韓松），「TinyML and Efficient Deep Learning Computing | MIT 6.S965 Fall 2022」(Youtube)

https://www.youtube.com/watch?v=5HpLyZd1h0Q&list=PL80kAHvQbh-ocildRaxjjBy6MR1ZsNCU7

歐尼克斯實境互動工作室
(OmniXRI Studio)
許哲豪 (Jack Hsu)

Facebook : Jack Omnixri
FB社團： Edge AI Taiwan邊緣智能交流區
電子信箱：omnixri@gmail.com
部落格：https://omnixri.blogspot.tw
開　源：https://github.com/OmniXRI

YOUTUBE 直播： https://www.youtube.com/@omnixri1784/streams