AI training course
HW12

STuser19 賴昱凱

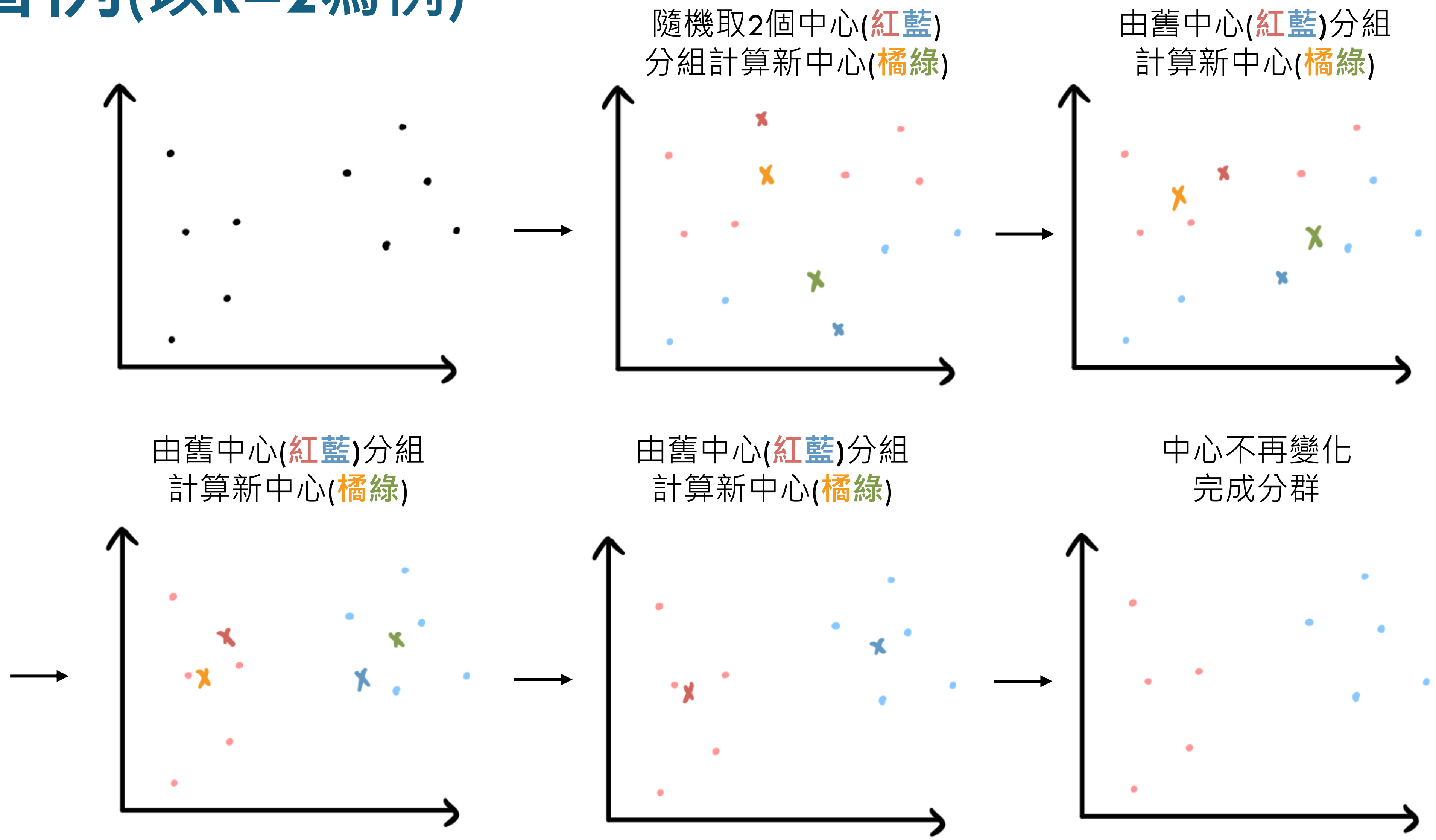
k-means分群演算法

將資料依照相似度分成k個群組

演算法

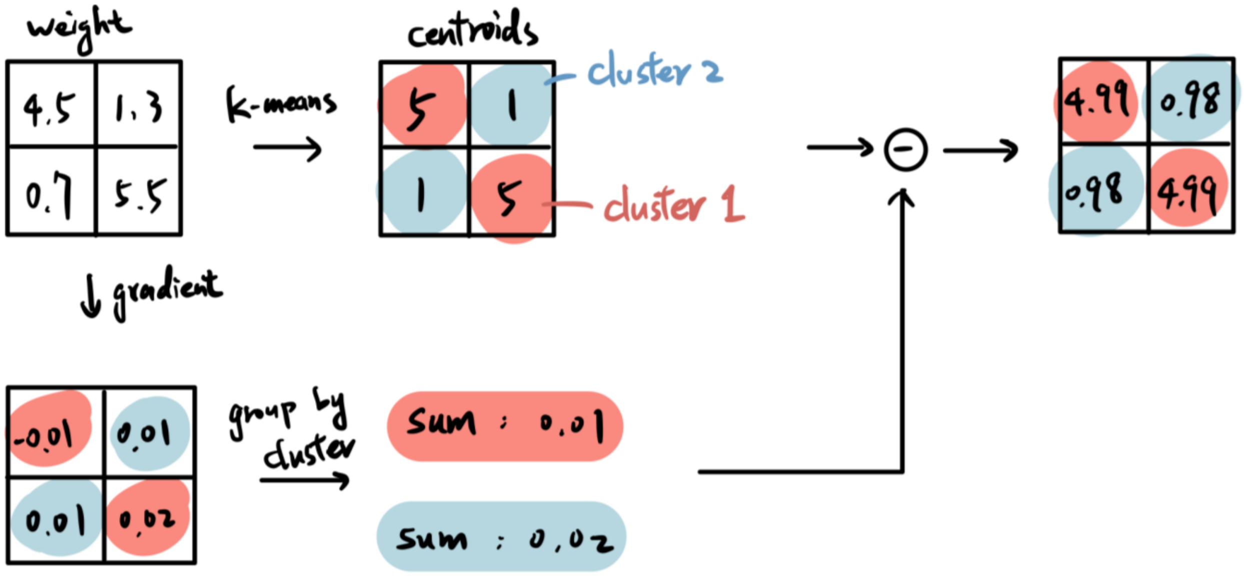
1. 隨機取k點作為中心
2. 計算各資料點與各中心的直線距離
3. 將資料點納入直線距離最近的中心群組
4. 計算群組的新中心
5. 重複執行3、4直到中心不再變化

圖例(以k=2為例)


隨機取2個中心(紅藍)
分組計算新中心(橘綠)
由舊中心(紅藍)分組
計算新中心(橘綠)
由舊中心(紅藍)分組
計算新中心(橘綠)
由舊中心(紅藍)分組
計算新中心(橘綠)
中心不再變化
完成分群

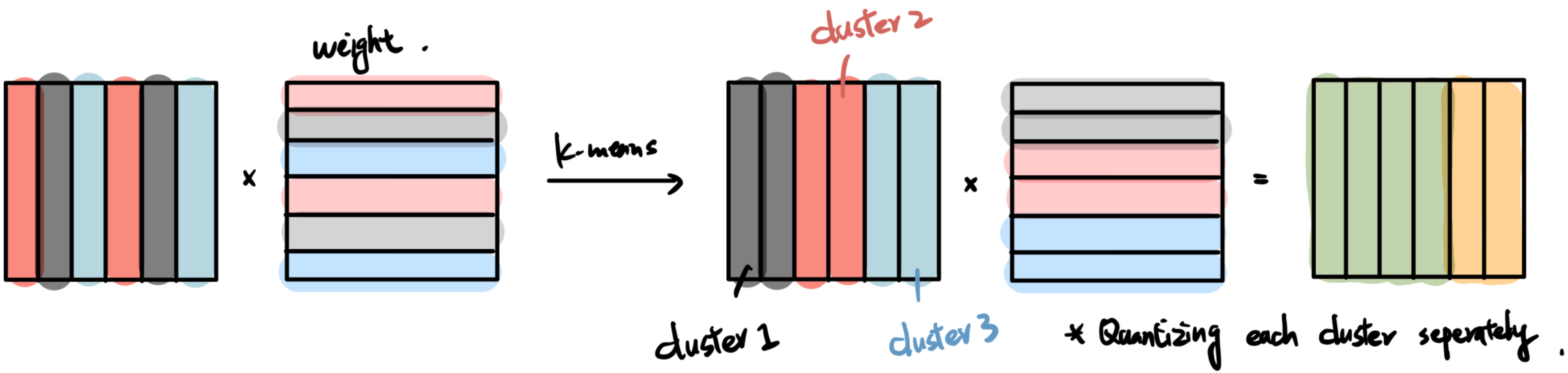
怎麼用來壓縮NN模型(一)

1. 利用k-means將權重分群，形成k個clusters
2. 每個Cluster中的資料皆由中心取代(可減少記憶體需求)
3. Fine-tuning with summation of corresponding gradients



怎麼用來壓縮NN模型(二)

1. 將data利用k-means分成k個clusters
2. Reorder by clusters
3. 各clusters分別做Quantization (可降低模型壓縮後的表現缺失)



Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang ,Andrew Howard, Hartwig Adam, Dmitry Kalenichenko

Google Inc.

Introduction

現狀問題

- CNN模型過大無法於mobile device使用

現有解法

- 新架構: MobileNet、SqueezeNet、ShuffleNet、DenseNet等
- Quantization: 用低位元降低儲存成本

現有Quantization缺陷

- 過度參數化的模型易壓縮，較無參考價值: AlexNet等
- 未於硬體驗證效果，且主要著重於減少儲存而非增加速度

Specific Contributions

Quantization scheme

- weight、activation: 8-bit int, Bias: 32-bit int

Quantized inference framework

- 於int-only hardware實現高效運算

Quantized training framework

- low-precision fixed-point arithmetic

benchmark results

- 基於MobileNet在ARM CPU執行，ImageNet及COCO表現優異平衡

Quantized Scheme

Inference : integer-only, training : floating-point

$$\boldsymbol{Quantization\ scheme:\ r = S(q - Z)} \qquad (1)$$

r : real number

q : integer (quantized values)

S : positive real number (const)

Z : integer corresponding to r = 0 (zero-point, const)

Integer-arithmetic-only
matrix multiplication

multiplication of two N × N matrices of real numbers r_1, r_2 and $r_3 = r_1 r_2$

$$r_\alpha^{i,j} = S_\alpha \left(q_\alpha^{i,j} - Z_\alpha \right), \qquad \begin{cases} 1 \le i, j \le N \\ \alpha = 1, 2, 3 \end{cases} \qquad (2)$$

From the definition of matrix multiplication

$$S_3 \left(q_3^{i,k} - Z_3 \right) = \sum_{j=1}^{N} \left(q_1^{i,j} - Z_1 \right) \left(q_2^{j,k} - Z_2 \right) \qquad (3)$$

$$q_3^{i,k} = Z_3 + M \sum_{j=1}^{N} (q_1^{i,j} - Z_1)(q_2^{j,k} - Z_2) \qquad (4), \qquad M := \frac{S_1 S_2}{S_3} \qquad (5)$$

We empirically find M to always be in the interval (0, 1)

$$\boldsymbol{normalized\ form : M = 2^{-n} M_0 \ ,\qquad \begin{cases} M_0 \in [0.5, 1) \\ n \in N \end{cases} \qquad (6)}$$

$$\boldsymbol{M_0 : fixed - point\ multiplier}$$

- 以int32為例，表示M_0的整數為最接近$2^{31}M_0$的int32值，且因$M_0 \geq 0.5$故該值至少為2^{30}。
- 乘以2^{-n}可以用bit-shifting實現

Efficient handling of zero-points

Efficiently implement the evaluation of Equation (4), $O(2N^3)$

$$\boldsymbol{q_3^{i,k} = Z_3 + M\left(NZ_1Z_2 - Z_1a_2^k - Z_2\bar{a}_1^i + \sum_{j=1}^{N} q_1^{i,j}q_2^{j,k}\right)} \qquad (7)$$

$$where\ a_2^k = \sum_{j=1}^{N} q_2^{j,k}, \bar{a}_1^i = \sum_{j=1}^{N} q_1^{i,j} \qquad (8), each\ N\ additions$$

a_2^k and $\bar{a}_1^i$: $2N^2$ additions, $\sum_{j=1}^{N} q_1^{i,j} q_2^{j,k}$ (9): $2N^3$ arithmetic operations

Implementation of a typical fused layer

- 將bias-addition、activation 融合至矩陣乘法
- Weights and activations use uint8 or int8
- Accumulator uses int32 to store (uint8 * uint8)
- Bias vector is quantized to int32 with $S_{bias} = S_1 S_2$, Z=0
- Output: int32 → int8 [0, 225] (fixed-point multiplication)

Training with simulated quantization

Inference with int、training with FP 的缺點

- 小模型表現差
- 數值範圍小的weight誤差大
- 異常值使精度降低

解法

在forward中simulates quantization，並維持bias、weight用FP

- Weights are quantized before ConV.
- Activations are quantized at points they would be during inference

Learning quantization ranges

Weight Quantization

- 範圍由[min w, max w] 量化為 [−127, 127]

Activation Quantization

- 由Training及EMA找範圍[a, b]。訓練初期不使用防止遺失重要數值

Boundary Adjustment

- 微調[a, b] 確保 0.0 在量化後精確表示為integer

Quantization in TensorFlow

- training使用simulated quantization，inference使用low-bit quantization

Batch normalization folding

- 使用Batch Norm的模型
  - Training : 獨立操作
  - Inference :“folding” into layers. e.g. ConV. 包含$\text{Weight}_{\text{fold}}$及bias

$$w_{fold} := \frac{\gamma w}{\sqrt{\sigma_B^2 + \varepsilon}}, \begin{cases} \gamma:\ scale\ parameter \\ \sigma_B^2: estimated\ variance\ of\ ConV.\ results \\ \varepsilon: small\ constant \end{cases}$$

EXP-Quantized training of Large Networks

Accuracy :

- for ResNet and InceptionV3, FP vs. int accuracy is within 2%

Different Quantization

- ResNet

$$Integer - only \approx 5 - bit\ INQ > 2 - bit\ FGQ$$

- Inception v3

$$7 - bit \approx 8 - bit$$

$ReLU6$表現較佳

EXP-Quantization of MobileNets

MobileNet Quantization (within the same runtime)

- accuracy : Integer-only quantized > floating-point models for ~10%
- Snapdragon 835 LITTLE cores : 33ms latency for real-time (30 fps) operation

COCO Object Detection

- 2x speedup with only ~1.8% accuracy loss

Face Detection and Attribute Classification

- reduce latency by~2x with only ~2% accuracy loss
- Single, Multi core 皆有大幅加速