Advanced topics on Deep Learning

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

Today

- Advanced topics on Deep Learning
  - Bias-Variance Tradeoff (Recap & In Deep Learning)
  - Dealing with overfitting: Regularization techniques
    - L2 Regularization
    - Dropout
    - Data Augmentation
    - Early Stopping
    - Batch Normalization
  - Vanishing Gradient Problem
    - Weight Initialization
    - Skip Connection
  - Optimization Schemes
  - Transfer Learning

Bias-Variance Tradeoff Recap

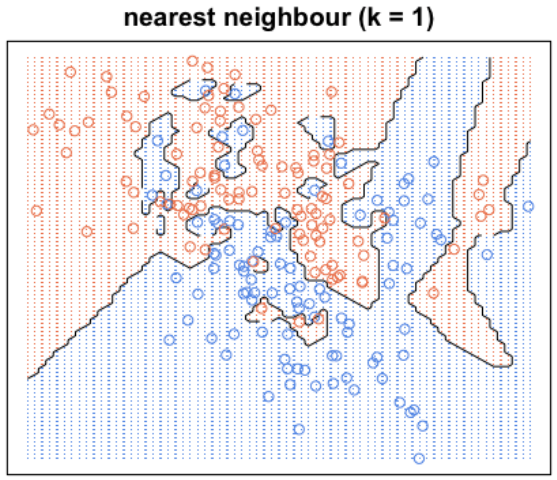
- Assuming true relationship is $Y = f(x) + \varepsilon$
  - where noise $\varepsilon \sim N(0, \sigma^2)$
- We want to make a model $\hat{f}(x; D)$ of f using D as the training samples
- Expected squared test error is $Err(x) = E_D[\left(Y - \hat{f}(x)\right)^2]$
- which can be decomposed as

$$Err(x) = \left(E_D[\hat{f}(x)] - f(x)\right)^2 + E_D\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma^2$$

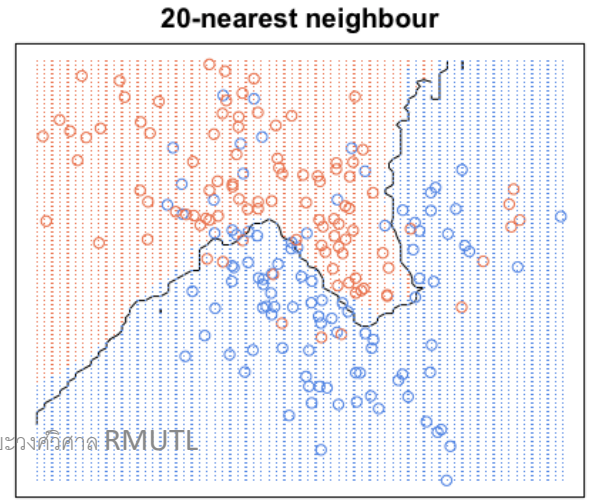
Bias Variance Irreducible Error

Bias-Variance Tradeoff Recap

- ด้วยทฤษฎีทางสถิติ พบว่าเราสามารถแตก **generalization error (expected test error)** ของโมเดลออกเป็นจาก **3** แหล่ง ได้แก่
  - **bias** = ทำนายผิดเพราะ **model** มี **assumption** เกี่ยวกับข้อมูลที่ผิด เรียบง่ายไป (**complexity** ต่ำไป) ทำให้เกิดการ **underfit** (ภาพขวา)
  - **variance** = ทำนายผิดเพราะ **model** อ่อนไหวต่อ **variation** ในข้อมูลเกินไป จึงเกิด **overfit** (ภาพซ้าย)
  - **irreducible error** = ทำนายผิดเพราะ **noise** ในข้อมูลตามธรรมชาติ จะปรับโมเดลอย่างไรก็ลดไม่ได้



low bias ☺
high variance ☹

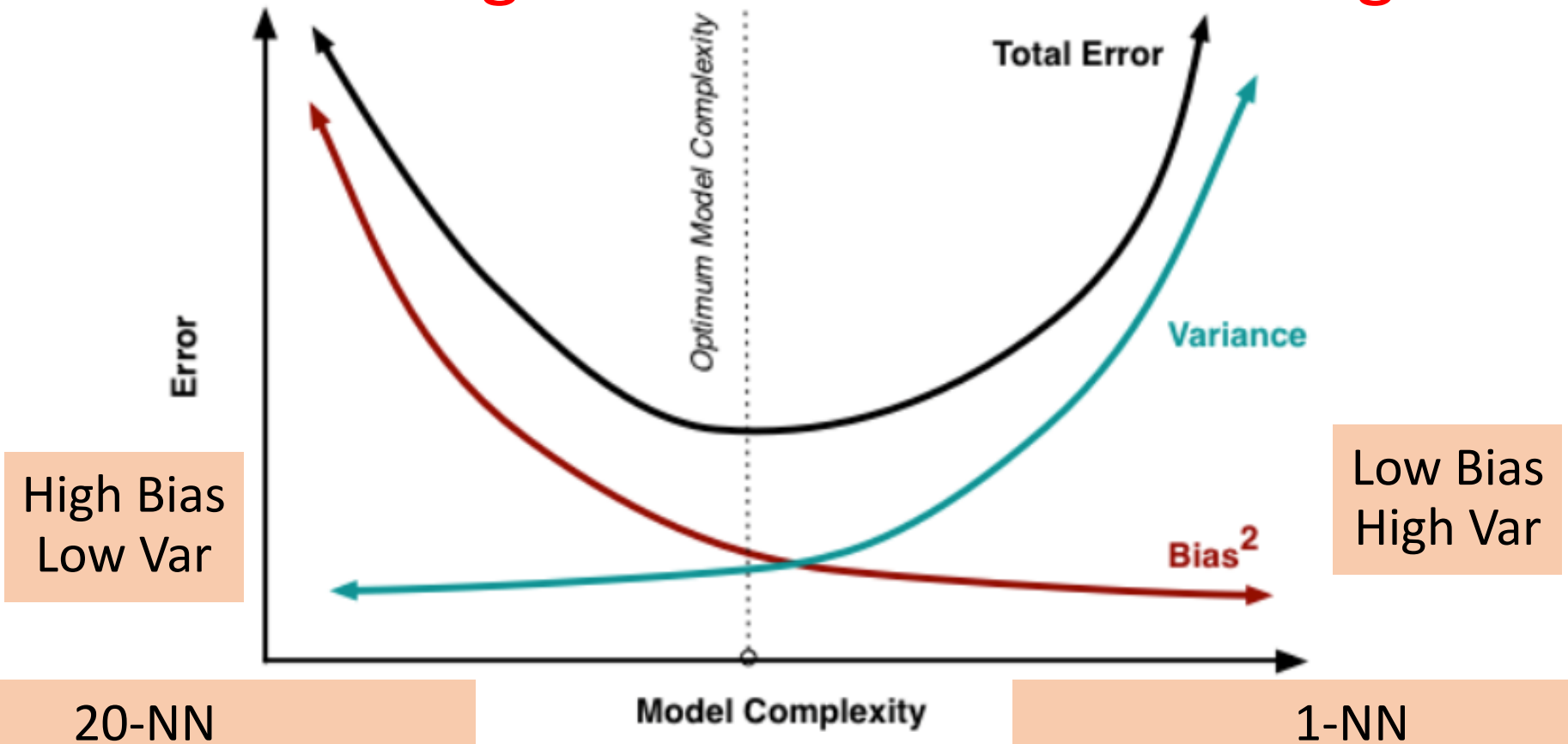


high bias ☹
low variance ☺

Bias-Variance Tradeoff

underfitting

overfitting



High Bias
Low Var

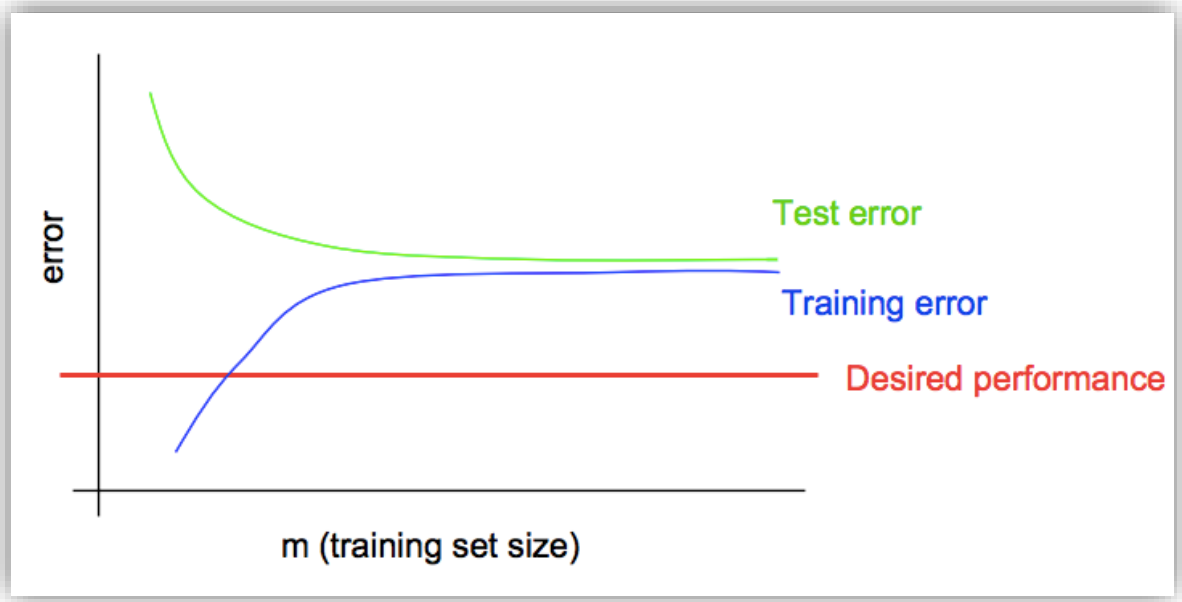
Low Bias
High Var

20-NN
Low-degree Poly Regression
Small Neural Net

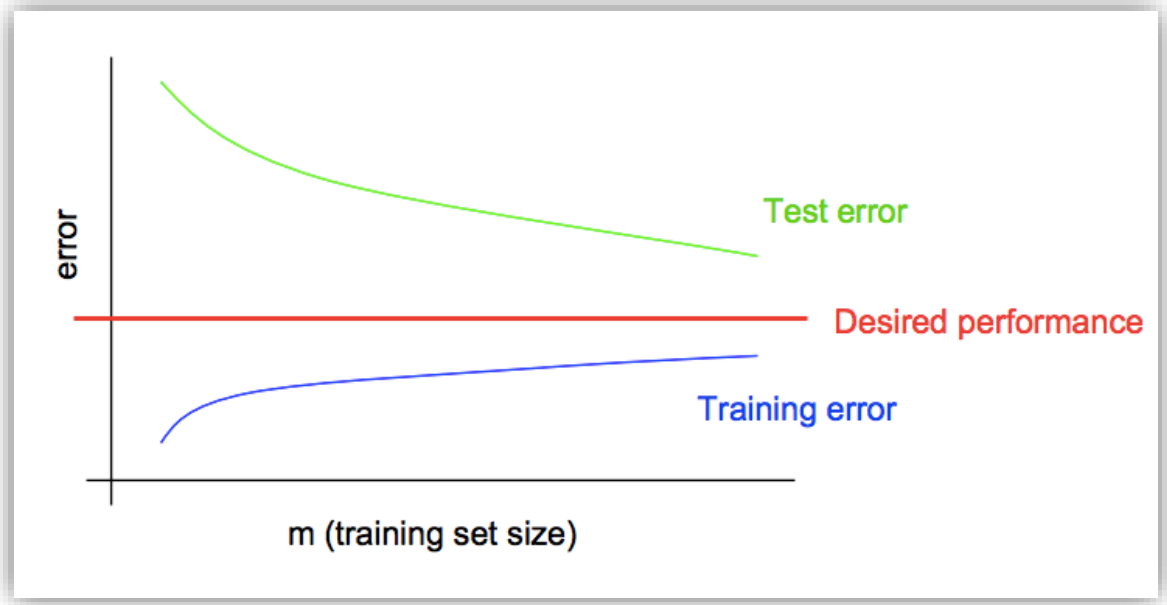
1-NN
High-Degree Poly Regression
Large Neural Nets

Learning curve

- Accuracy vs Training size plot



High Bias



High Variance

Source: http://cs229.stanford.edu/materials/ML-advice.pdf

Bias & Variance in practice

- How to check if our model has high variance/bias?
  - **Ans:** check errors on Train/Val sets

- Examples (assuming ~0% human error)
  - Train error 15%, Val error 16% ➔ High Bias
  - Train error 1%, Val error 11% ➔ High Variance
  - Train error 15%, Val error 30% ➔ High Bias, High Variance
  - Train error 0.5%, Val error 1% ➔ Low Bias, Low Variance

Bias & Variance in Deep Learning

- Basic recipe for fixing high bias/variance
  1. Fix high bias first
     - Try bigger networks (e.g. more layers, #neurons)
     - Train longer
  2. Fix high variance
     - More data
     - Regularization (L1/L2, dropout)

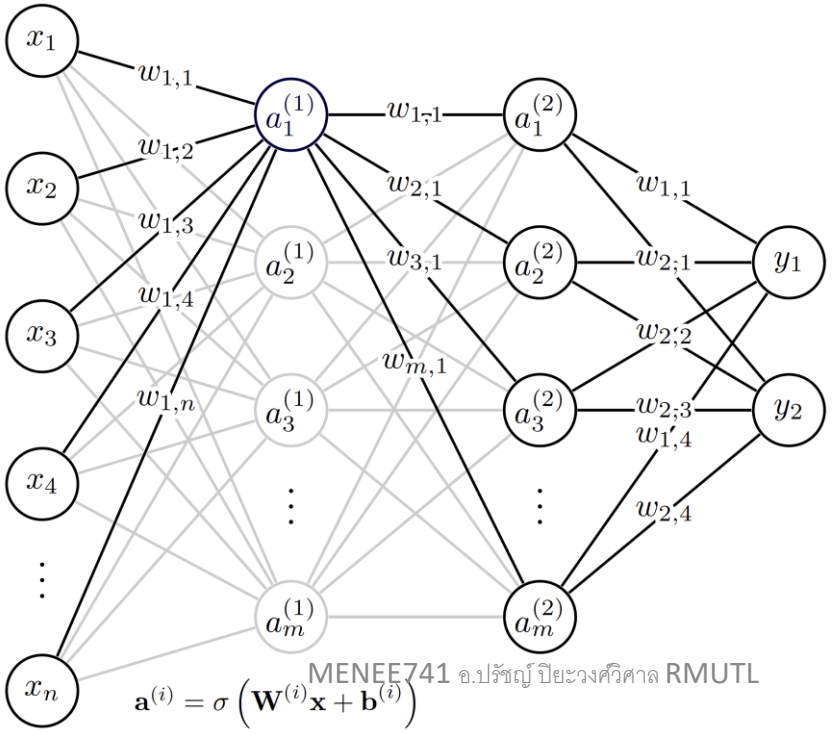
- In DL (vs traditional ML), Bias/Variance become less of a tradeoff
  - bigger network => reduce **bias** without hurting variance much (with proper regularization)
  - getting more data => reduce **variance** without hurting bias much

Regularization Techniques for NN

- Recap: Regularization helps reduce variance (overfitting)
- Methods
  - L1/L2 regularization
  - Dropout
  - Getting more data / Data Augmentation
  - Early stopping
  - Batch Normalization
  - Skipped connection

L2 Regularization

- (Recap) Cost function of NN:
  - $J\left(w^{[1]}, b^{[1]}, \ldots, w^{[L]}, b^{[L]}\right) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$
  - โดยที่ $w^{[l]}$ เป็น matrix ขนาด $\left(n^{[l-1]}, n^{[l]}\right)$

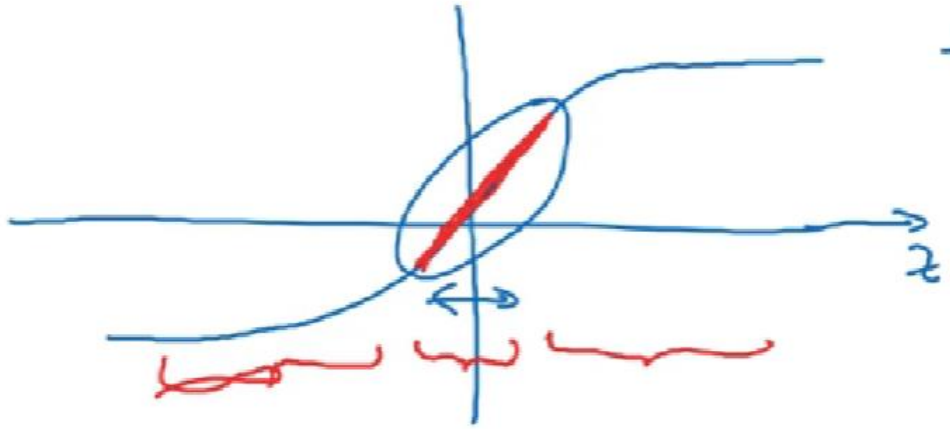


L2 Regularization

- Cost function with L2 regularization
  - $J\left(w^{[1]}, b^{[1]}, \ldots, w^{[L]}, b^{[L]}\right) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) + \frac{\lambda}{2m}\sum_{l=1}^{L}\left\|w^{[l]}\right\|_F^2$
  - โดยที่ λ คือ regularization parameter
  - และ $\left\|w^{[l]}\right\|_F^2 = \sum_{i=1}^{n^{[l-1]}}\sum_{j=1}^{n^{[l]}}\left(w_{ij}^{[l]}\right)^2$ คือ Frobenius norm ของ $w^{[l]}$

- ดังนั้นใน Gradient Descent เราจะสามารถคำนวณ gradient ได้ดังนี้
  - $\frac{\partial J}{\partial w^{[l]}}$ = ค่าที่ได้จาก backprop + $\frac{\lambda}{m}w^{[l]}$
- แล้วหากนำค่านี้ไป update $w^{[l]}$ จะทำให้ค่าของ $w^{[l]}$ ค่อยๆ ลดลง
  - บางทีจึงเรียก L2 regularization ว่า “weight decay”

L2 Regularization & Overfitting



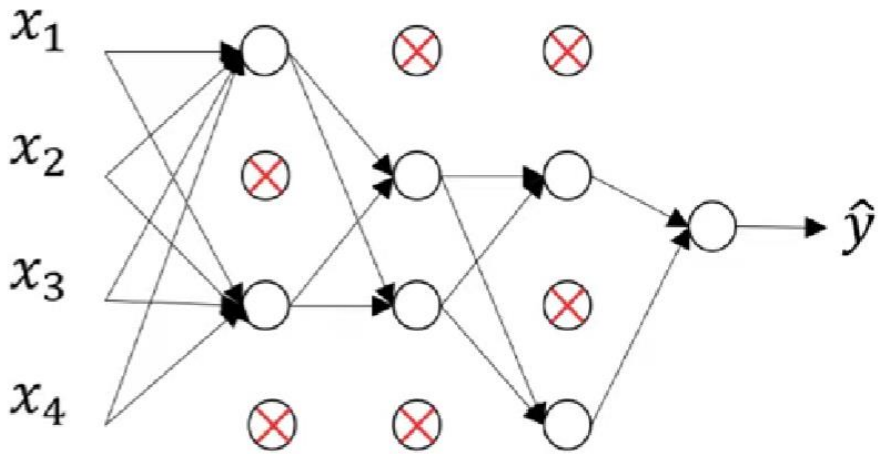
$g(z) = \tanh(z)$

$\lambda \uparrow \quad W^{[l]} \downarrow \quad z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$

Every layer $\approx$ linear.

Dropout

- สุ่มทิ้งบาง neuron ใน**ขณะ train** ด้วยความน่าจะเป็น p
  - neuron ที่ถูกทิ้งจะถูกตัด incoming/outgoing links ออก
- ทำการสุ่มทิ้งใหม่ สำหรับ**ทุก ๆ** training sample
- ใน inference phase จะไม่ทำ dropout
  - มิฉะนั้นคำตอบจะไม่ deterministic



Why dropout?

- มอง dropout เป็นการบังคับให้ neuron ไม่สามารถพึ่งแต่ input feature ใด feature หนึ่ง
  - จึงเกิดการ spread out ของค่า weight
  - และทำให้ค่า weight โดยรวมลดลง = มีผลคล้ายๆ L2 regularization

- **ประโยชน์ของ Dropout**
  - ช่วยลด overfitting
  - ช่วยให้ train เร็วขึ้น
  - ใช้หน่วยความจำในการ train น้อยลง
  - มีผลคล้ายๆ การทำ ensemble

Dropout Implementation

- Inverted dropout (ใช้ keep_prob แทน drop_prob)

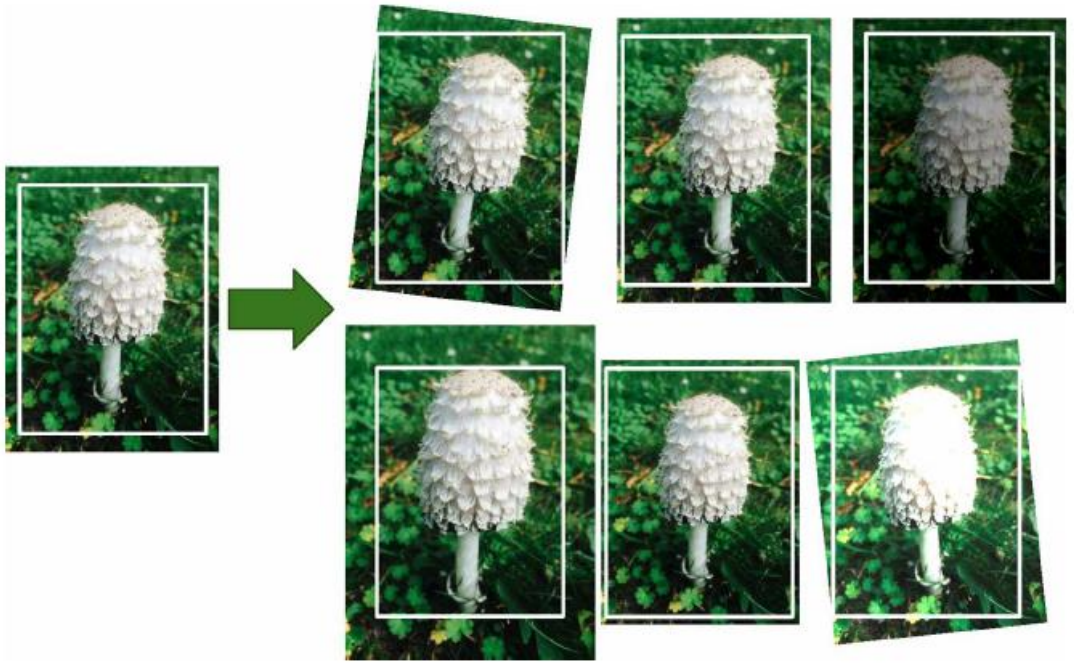
- Example dropout code for layer 3

```
d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep_prob
a3 = np.multiply(a3, d3)
a3 /= keep_prob
```

* โค้ดบรรทัดสุดท้ายทำเพื่อไม่ให้ expectation ของ activation ใน layer ถัดไปเคลื่อน

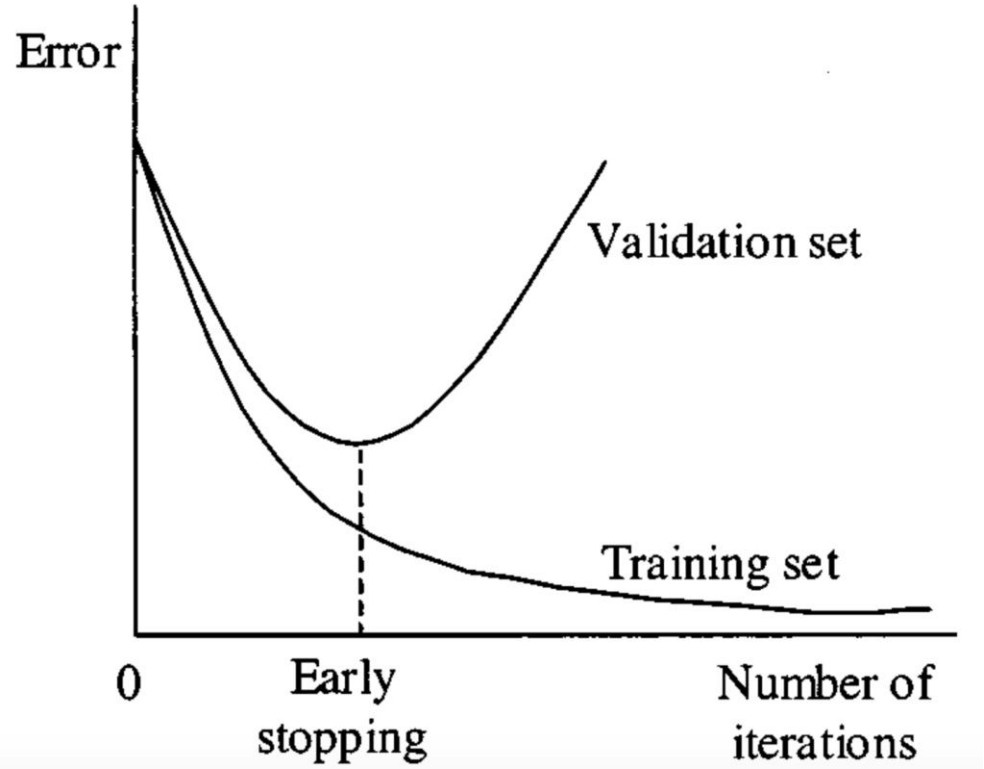
Data Augmentation

- ปัญหา: ข้อมูล **training set** ที่เป็นรูปภาพ **(CNN)** ไม่มากพอ
- **Solution:** สังเคราะห์รูปภาพใหม่ๆ ขึ้นมาจากภาพที่มีอยู่ โดยการ **transform** ภาพ
  - Shift
  - Rotate
  - Scale
  - ปรับสภาพแสง
    - Brightness
    - Contrast
    - Saturation



Early Stopping

- หยุด **train** เมื่อ **validation loss** เริ่มสูงขึ้นเกิน **threshold**



https://paperswithcode.com/method/early-stopping

Batch Normalization

- แทรก BN layer เข้าไปก่อนทุกๆ activation layer
- ทำการ zero-center ค่า output โดยใช้ mean, variance ของค่าใน batch นั้น
- ช่วยทำให้ train เร็วและเสถียรขึ้นด้วย
  - อธิบายยากว่าทำไม
  - intuition: ถ้า feature มี scale เดียวกันจะช่วยให้ cost function รูปร่างดีขึ้น optimize ง่ายขึ้น
- ควรทำ batchnorm ตอน inference ด้วย
- Paper: https://arxiv.org/abs/1502.03167

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

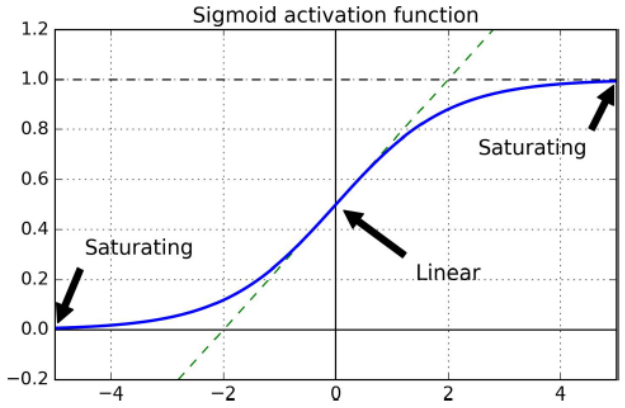
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Vanishing Gradient Problem

- ทบทวน: ใน **Backpropagation** ค่า **gradient** ของ **loss** จะถูกส่งย้อนจาก **layer** ปลายกลับมายัง **layer** ต้น เพื่อ **update** ค่า **weight** ทั้งหมด
- ปัญหา <u>สำหรับ **NN ที่ deep** มากๆ</u> :
  - ในขณะที่ส่งย้อน ค่า **gradient** ลดลงอย่างมาก เข้าใกล้ 0 *
  - ถ้า **gradient** เป็น 0 จะทำให้ **weight** หยุดการ **update**
  - ทำให้การเรียนรู้หยุดชะงัก ☹



- Solution
  - ใช้ **activation function** ที่ไม่ saturate (ใช้ ReLU, Leaky ReLU, ELU แทน Sigmoid)
  - เปลี่ยนวิธี **weight initialization** (ใช้วิธี Xavier หรือ He แทน random, หรือใช้ pre-trained network)
  - ใส่ **carry track** หรือ **skip connection** ในโมเดล (LSTM, ResNet)

* อาจเกิดปัญหาตรงข้าม (Exploding Gradient) ได้เช่นกัน หาก **gradient** โตขึ้นอย่างรวดเร็ว ทำให้การ **optimize** ไม่ converge

Better Optimization Schemes

- ปัญหา: SGD optimizer ใช้เวลา train นานและอาจได้คำตอบที่เป็น local minima
- Solution: เปลี่ยนไปใช้ optimizer ที่ดีกว่า เช่น
  - Momentum ใช้ momentum ทำให้หลุดจากหล่ม local minima
  - AdaGrad ค่อยๆ ลด (decay) learning rate ในทิศที่ชันที่สุด
  - RMSProp ปรับ AdaGrad ให้ไม่ decay แรงเกินไป
  - ADAM รวมข้อดีของ RMSProp กับ Momentum
- ในทางปฏิบัติให้ลองใช้ ADAM ไปเลย ไม่ก็ RMSProp

Transfer Learning

- ปัญหา: ต้องการนำโมเดลที่ **train** กับข้อมูลทั่วไป ไปใช้กับงานที่เฉพาะเจาะจง
- Solution: Transfer Learning
  - นำ **pre-trained weights** ที่ **train** กับข้อมูลทั่วไปมาใช้เป็นจุดเริ่มต้น
  - ทำการ **train** กับข้อมูลชุดใหม่ที่เป็นงานเฉพาะ
  - ระหว่าง **train** ทำการ **freeze layer** แรกๆ แล้ว **train** เฉพาะ **layer** ท้ายๆ
  - ช่วยให้ **train** เร็วขึ้นมาก และทำนายได้แม่นยำสูงขึ้นมาก