CHƯƠNG 3
TỐI ƯU HOÁ MÔ HÌNH MẠNG NEURAL (P1)

Khoa Khoa học và Kỹ thuật thông tin
Bộ môn Khoa học dữ liệu

NỘI DUNG

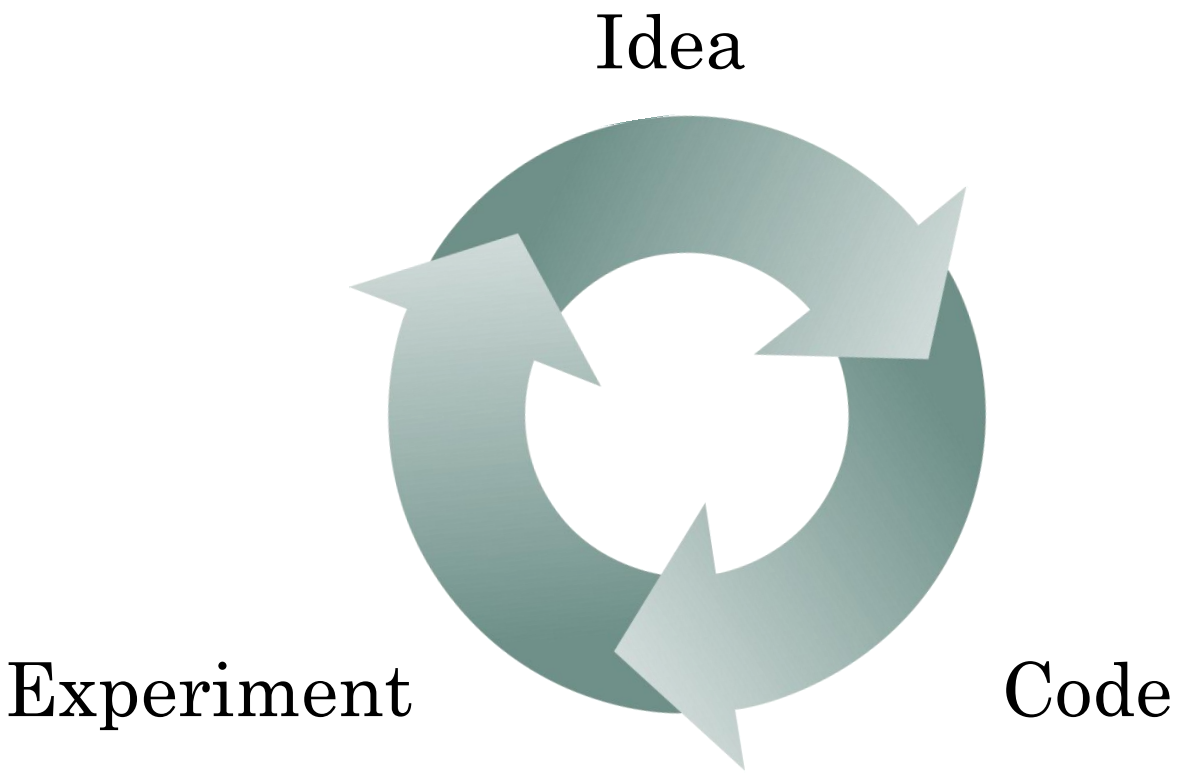
1. Phân chia dữ liệu huấn luyện.
2. Deep learning error.
3. Chuẩn hoá mạng neural (Regulazation)
4. Khởi tạo trọng số.

PHÂN CHIA DỮ LIỆU HUẤN LUYỆN

Các siêu tham số khác trong mô hình

- Momentum.
- Epochs.
- Mini batch size.
- Regularzation.
- .....

Điều chỉnh siêu tham số



- Quá trình áp dụng deep learning là 1 quá trình thực nghiệm.
  + Điều chỉnh các siêu tham số (ví dụ như learning rate, iteration, etc) sao cho hàm chi phí đạt giá trị nhỏ nhất.
- Ngoài ra, việc phân chia dữ liệu thành các tập train/dev/test ảnh hưởng rất lớn đến hiệu quả của mô hình

Phân chia dữ liệu

Cách chia dữ liệu

	TRAIN	DEV	TEST
	%	%	%
Truyền thống (~10K)	60	20	20
Big data (1M)	98	1	1
Hơn 1M dữ liệu	99.5	0.25	0.25

Trường hợp dữ liệu train và dev/test không khớp

Train

- Hình chụp các con mèo từ các trang web.
- Bộ văn bản tiếng Việt chính quy (từ Wikipedia)

Dev/test

- Hình các con mèo tự chụp bằng camera.
- Bộ văn bản tiếng Việt trên mạng xã hội.

Dữ liệu từ tập train và tập dev/test phải từ cùng một nguồn, cùng một tính chất với nhau (có cách gọi khác là same distribution). Nếu không thì việc đánh giá mô hình mạng neural sẽ không thể chính xác được.

Tập test có cần thiết không

- Tập dev thường dùng để đánh giá và so sánh độ chính xác giữa các mô hình với nhau.
- Tập test dùng để kiểm tra lại việc đánh giá trên tập dev xem kết quả có chính xác hay không (đôi khi còn gọi là unbiased estimate).
  + Nếu như kết quả giữa dev và test không chênh lệch nhiều thì có thể chấp nhận độ chính xác của mô hình, ngược lại thì sẽ xảy ra hiện tượng overfiting.
- Trên thực tế, có khi **sẽ không nhất thiết phải dùng tập test set mà chỉ cần dev set là đủ** ➔ kiểm tra độ chính xác của mô hình.
  + Nếu chỉ có train set và dev set mà không có test set thì tập d**ev set sẽ được gọi luôn thành test set**.

DEEP LEARNING ERROR

Underfiting và Overfiting

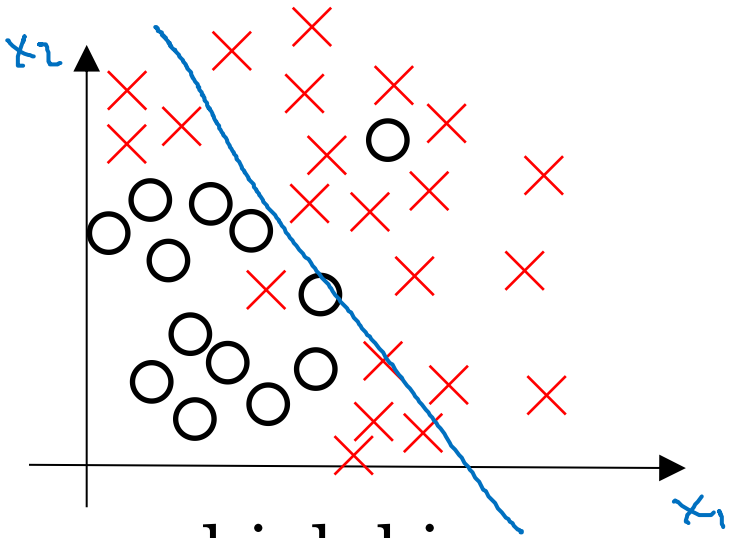
Underfiting

- Mô hình xây dựng được chưa tổng quát được trên tổng thể bộ dữ liệu huấn luyện ➔ độ chính xác khi dự đoán của mô hình thấp.
- **High bias.**
- Độ chính xác trên tập train thấp.

Overfiting

- Mô hình xây dựng thể hiện quá chi tiết trên bộ dữ liệu huấn luyện ➔ các dữ liệu nhiễu cũng được đưa vào ➔ khi gặp các dạng nhiễu khác, mô hình sẽ không còn chính xác ➔ trên bộ dev (hoặc test) độ chính xác khi dự đoán sẽ thấp.
- **High variance.**
- Độ chính xác trên tập train cao, nhưng tập dev/test thấp.

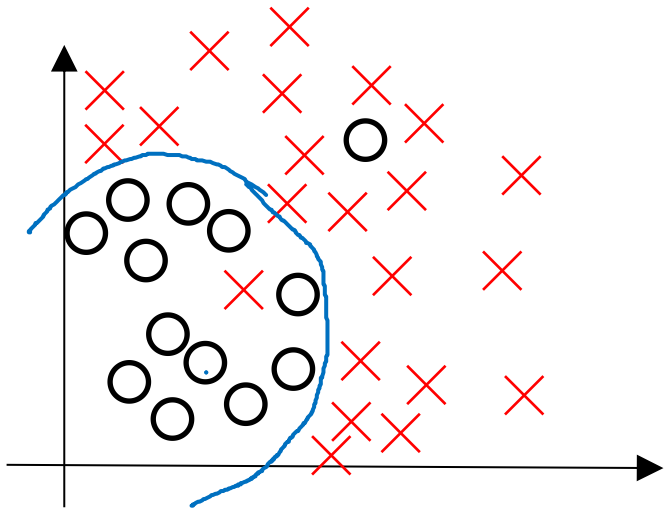
Bias and Variance – Deep learning error



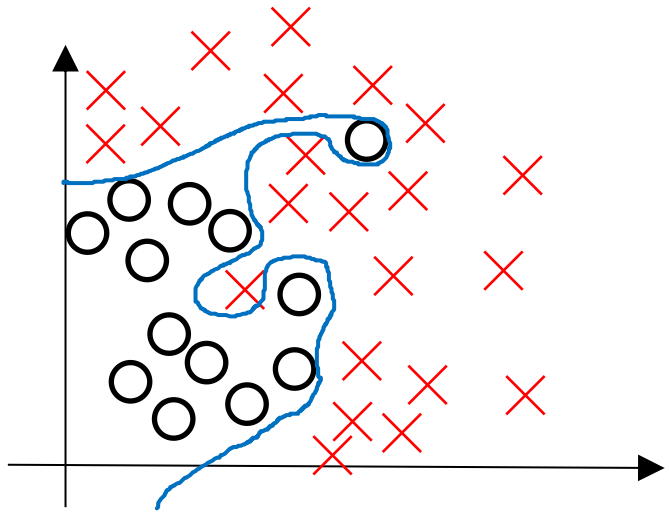
high bias

Underfiting

Underfiting: Đường thẳng dự đoán không khớp với lại dữ liệu dự đoán ➔ không tổng quát được trên dữ liệu.



→ "just right"



high variance

Overfiting

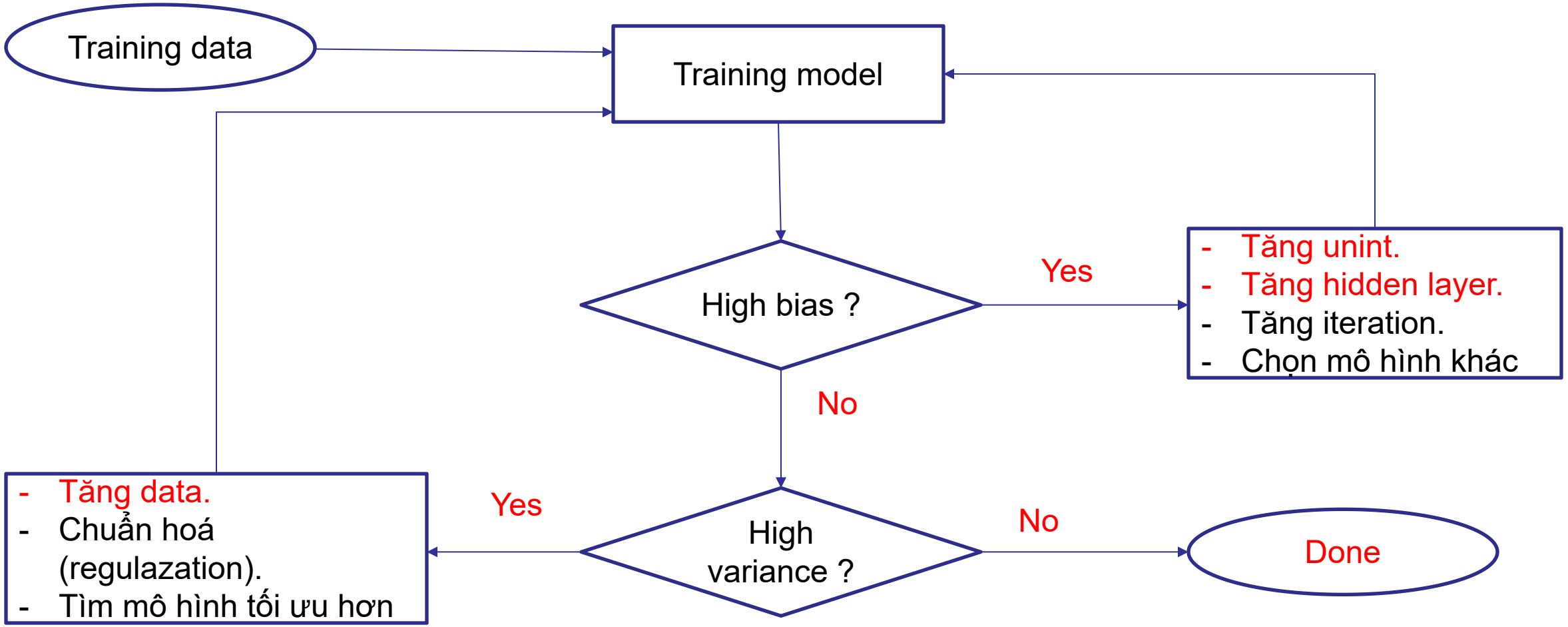
Overfiting: Đường thẳng dự đoán quá khớp với lại dữ liệu dự đoán (kể cả nhiễu).

Ví dụ

Cat classification

Train set error:	1%	15%	15%	1%
Dev set error:	11%	16%	30%	0.5%
	high variance	high bias	high bias + high variance	low bias + low variance

Quy trình tối ưu cho mô hình



Bias and Variance trade off

- Đối với máy học truyền thống, Bias và Variance tỉ lệ nghịch với nhau:
  + Tăng bias sẽ làm giảm variance.
  + Giảm bias sẽ làm tăng variance.
- Tuy nhiên, trong kỹ nguyên Big data và deep learning:
  + Tăng kích thước mạng neural (số lớp ẩn, số unit) sẽ làm *giảm bias mà không ảnh hưởng nhiều tới variance*.
  + Tăng dữ liệu sẽ làm giảm variance mà không ảnh hưởng nhiều tới bias.

CHUẨN HOÁ MẠNG NEURAL – GIẢM DEEP LEARNING ERROR: OVERFITING

Regularzation

- Để hạn chế việc overfiting (high variance) trên mô hình, ta có các kỹ thuật sau:
  + Tăng cường data.
  + Chuẩn hoá mô hình mạng neural (regularzation).
- Regularzation là một kỹ thuật điều chỉnh tham số cho mạng neural nhằm giải quyết quá trình overfiting.

Nhắn lại về Logistic regression

– Hàm chi phí:

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

Trong đó:

$W \in R^{nx}$, $b \in R$.

Chuẩn hoá L2 cho Logistic regression

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m} * \|w\|_2^2 + \frac{\lambda}{2m} * b^2$$

L2 Regularzation $\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w$

Ridge Regression.

b là một số đơn lẻ (single number) ➔ xét về kích thước, b nhỏ hơn rất nhiều so với w – vốn chứa rất nhiều tham số. Do đó, việc chuẩn hoá b không ảnh hưởng đáng kể so với chuẩn hóa w ➔ có thể bỏ đi.

Chuẩn hoá L1 cho Logistic regression

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \boxed{\frac{\lambda}{2m} * \|w\|_1}$$

L1 Regularzation $\|w\|_1 = \sum_{j=1}^{n_x} |w|$

Lasso Regression

Trong trường hợp này, **w là một ma trận thưa (sparse matrix)** – là một ma trận mà hầu hết các phần tử bằng 0. Ngược lại với ma trận thưa là ma trận dày đặc (dense matrix), nơi mà hầu hết các phần tử khác 0.

MỘT SỐ LƯU Ý

- L1 regularzation sẽ khiến cho tham số của mô hình bị thưa, do đó trong thực tế, người ta thường sử dụng chuẩn hoá L2 regularzation.
- λ được gọi là tham số chuẩn hoá (regularzation parameter).
  + Cài đặt thông số cho λ sử dụng tập dev (development test) hoặc là sử dụng cross validation.
  + λ cũng là một loại siêu tham số (hyper parameter) trong mô hình.

Regularzation trong mạng neural

– Hàm chi phí của mô hình mạng neural L lớp:

$$J(w^{[1]}, b^{[1]}, \ldots, w^{[L]}, w^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

– L2 Regularzation:

$$J(w^{[1]}, b^{[1]}, \ldots, w^{[L]}, w^{[L]}) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} ||w^{[l]}||_F^2$$

Frobenius norm

Frobenius norm

$$||w||_F^2 = \sum_{i=1}^{n^{[l-1]}} \sum_{j=1}^{n^{[l]}} \left(w_{ij}^{[l]}\right)^2$$

Chiều của w: ($n^{[l]}$, $n^{[l-1]}$)

REGULARZATION TRONG MẠNG NEURAL

Gradient Descent:

$W^{[l]} = W^{[l]} - \alpha * dW^{[l]}$ Với: $dW^{[l]} = <back\_propagation> + \frac{\lambda}{m} w^{[l]}$

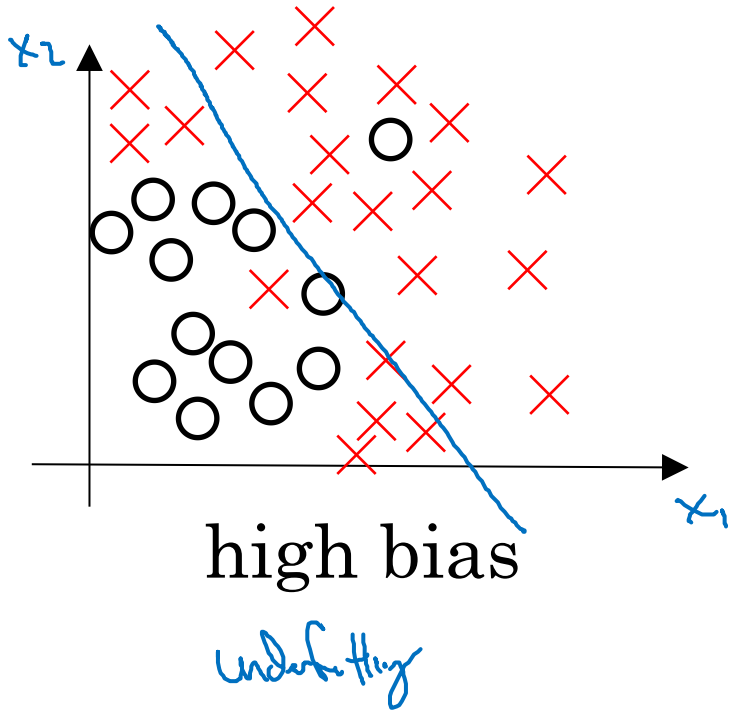
$= W^{[l]} - \boxed{\alpha * \frac{\lambda}{m} w^{[l]}} - \alpha * <back\_propagation>$

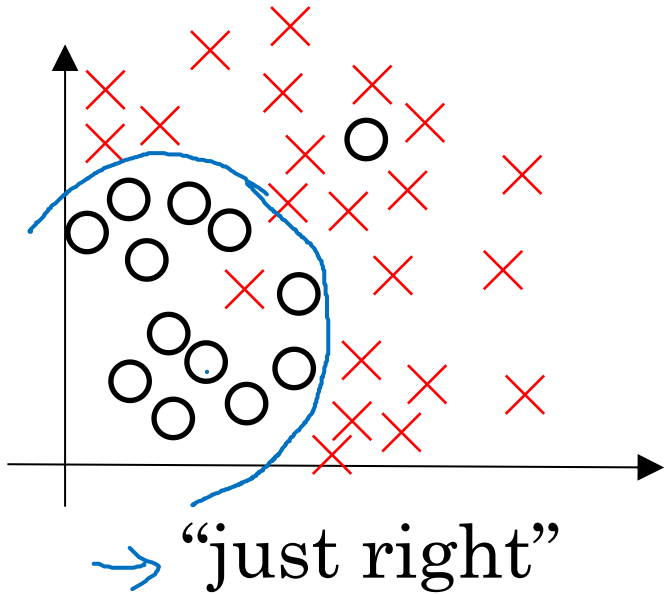
Mỗi lần cập nhật trọng số, thì giá trị trọng số W sẽ nhỏ đi một lượng $\alpha * \frac{\lambda}{m} w^{[l]}$

➔ weigh decay

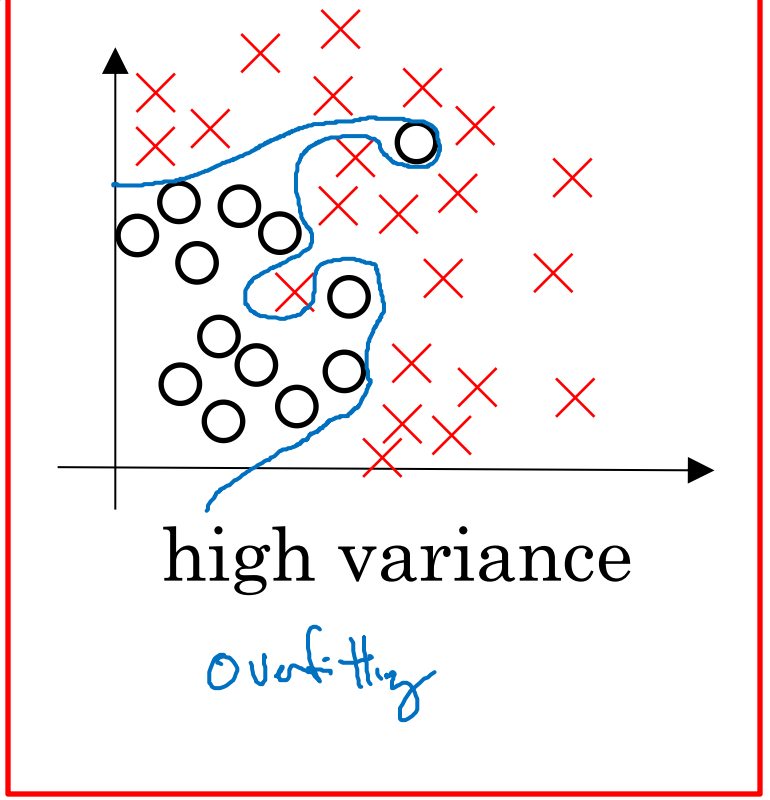
TẠI SAO REGULARZATION GIẢM ĐƯỢC OVERFITING

Nhắc lại về các dạng deep learning error





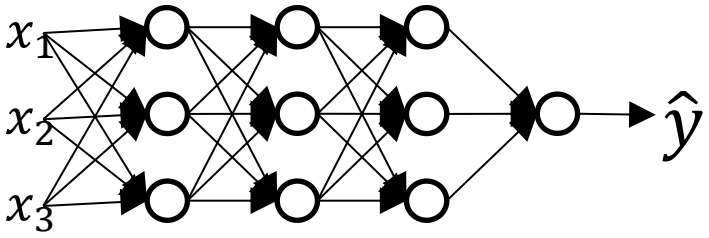
overfiting
(high variance)



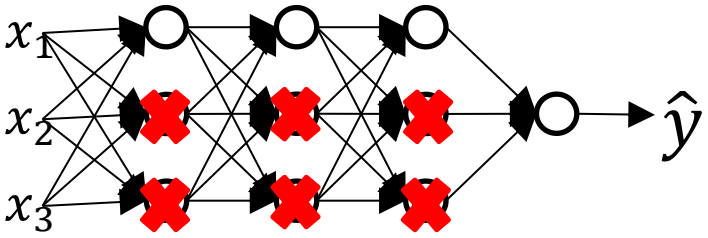
Giải thích 1: Trên hàm chi phí

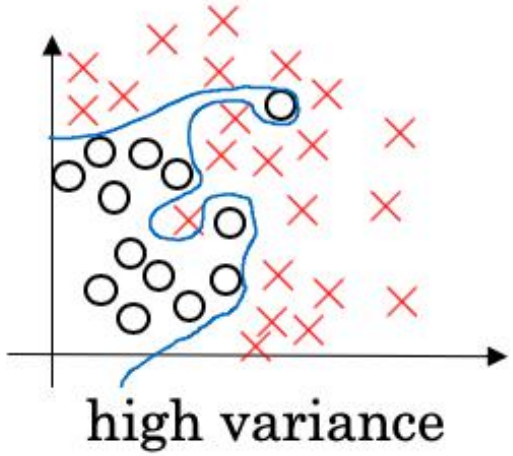
$$J(W^{[l]}, b^{[l]}) = \frac{1}{m} * L(\hat{y}, y)$$

$$J(W^{[l]}, b^{[l]}) = \frac{1}{m} * L(\hat{y}, y) + \frac{\lambda}{2m} \sum_{l=0}^{L} ||w^{[l]}||_F^2$$

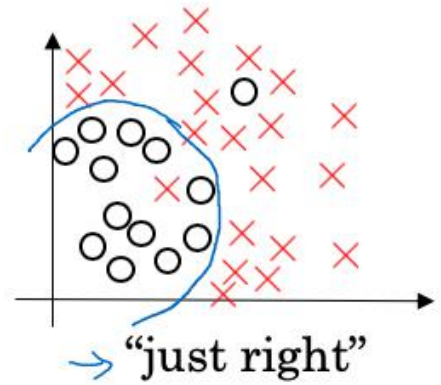


Regularzation

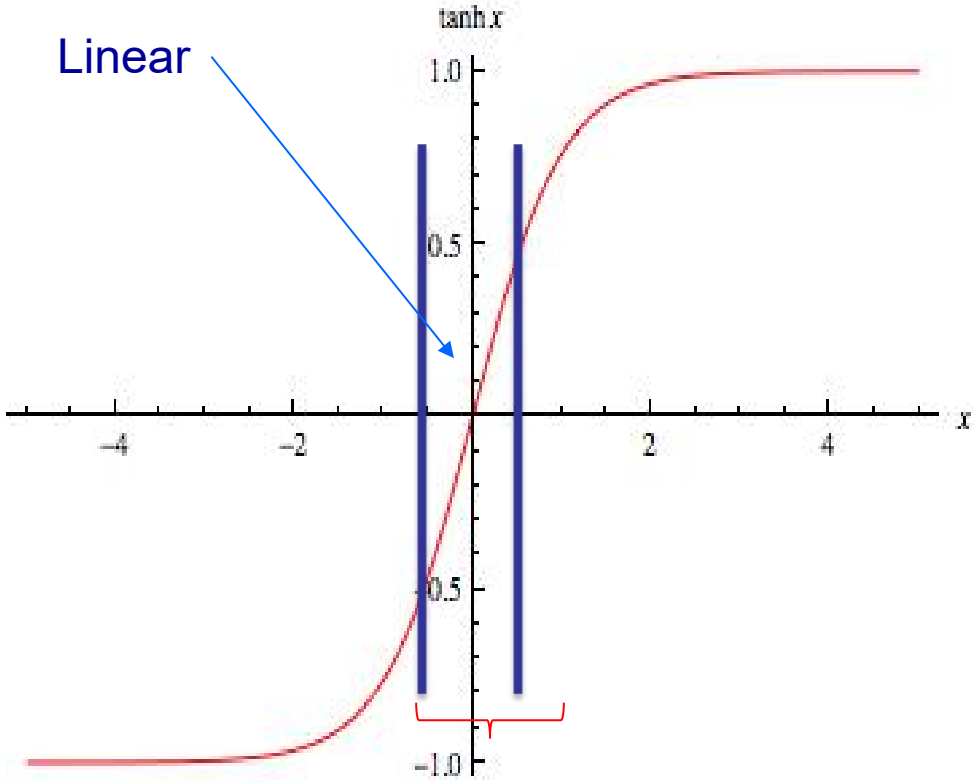




Khi có chuẩn hoá (regularzation), thì trong quá trình tính Gradient descent, một số trọng số **W[l]** (ứng với các unit) sẽ giảm về gần bằng 0 (**W[l]** ≈ **0**)
➔ một số neural (unit) bị vô hiệu hoá. Khi đó, đường dự đoán (mô hình) sẽ "bớt khớp" với dữ liệu hơn



Giải thích 2: Trên hàm truyền



Z càng nhỏ ➔ hàm tanh có xu hướng quay về linear

Mỗi lần cập nhật trọng số, thì giá trị trọng số W sẽ nhỏ đi một lượng $\alpha * \frac{\lambda}{m} w^{[l]}$

➔ **λ càng lớn thì $w^{[l]}$ càng nhỏ.**

$Z^{[l]} = W^{[l]} * a^{[l-1]} + b^{[l]}$

$g(Z^{[l]}) = \tanh(Z^{[l]})$

➔ **λ càng lớn thì $w^{[l]}$ càng nhỏ**

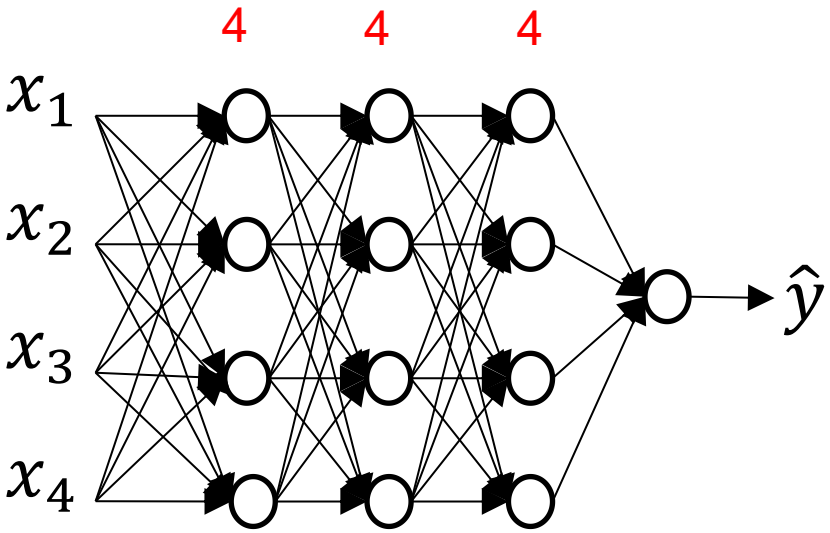
➔ **$Z^{[l]}$ càng nhỏ** ➔ $g(Z^{[l]})$ càng nhỏ.

Dropout regularzation

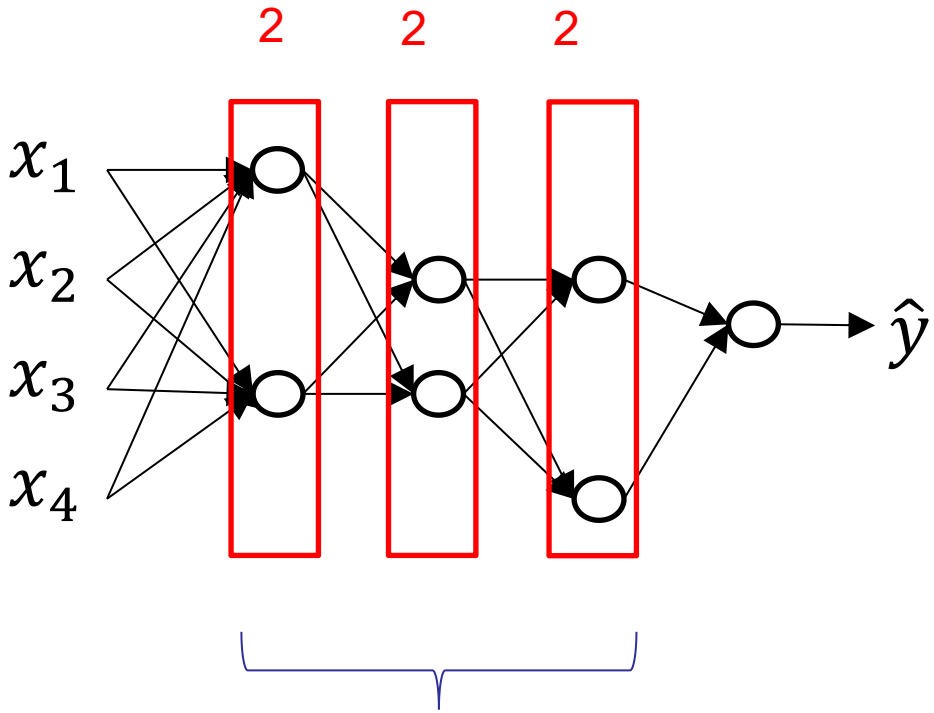
- Bên cạnh L2 regularzation, dropout là một kỹ thuật regularzation khác nhằm hạn chế overfiting.
- Dropout là kỹ thuật bỏ đi một số unit trong mỗi lớp nhằm đơn giản hoá mạng neural:
  + Bỏ đi một số units => giảm $W^{[l]}$.
- Kỹ thuật này do Google phát triển và đã được cấp bằng sáng chế.

https://jmlr.org/papers/v15/srivastava14a.html

Minh hoạ



Dropout



Bỏ đi khoảng 50% (p=0.5) cho mỗi layer

Ví dụ

Giả sử, đang xét lớp thứ 3 trong mạng neural (l=3), xác suất giữ là 0.8 (keep_prob=0.8) ➔ khoảng 20% unit sẽ bị tắt.

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

Giả sử, a3 có 50 unit, khi thực hiện bỏ khoảng 20% unit xong, a3 còn lại khoảng 40 unit (20%*50 = 10).
Như vậy, ở lớp tiếp theo:
$z^{[4]} = W^{[4]}*a^{[3]} + b^{[4]}$, lớp $z^{[4]}$ sẽ chỉ còn khoảng 80% unit, do ta nhân với $a^{[3]}$ ($a^{[3]}$ đã dropout 20%)

Một số kỹ thuật khác nhằm làm giảm overfiting

- Tăng dữ liệu (data augmentation).
  + Chi phí đắt.
- Early stopping.

Ví dụ về Tăng dữ liệu ảnh

4

Ví dụ về Tăng dữ liệu TEXT

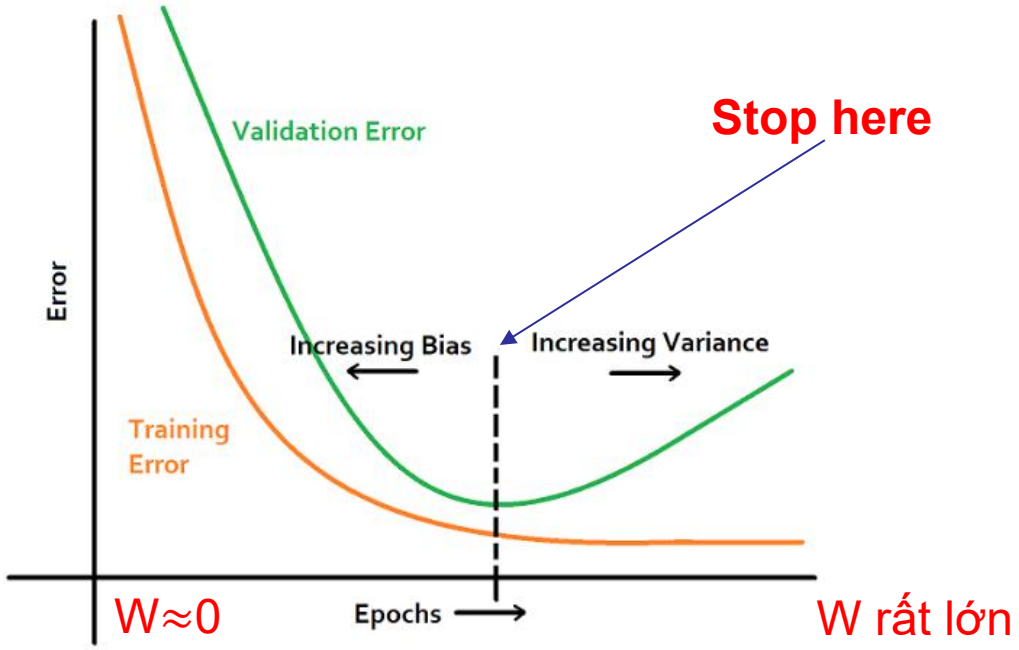
Operation	Sentence
None	A sad, superior human comedy played out on the back roads of life.
SR	A ***lamentable***, superior human comedy played out on the ***backward*** road of life.
RI	A sad, superior human comedy played out on ***funniness*** the back roads of life.
RS	A sad, superior human comedy played out on ***roads*** back ***the*** of life.
RD	A sad, superior human out on the roads of life.

DỮ LIỆU CHÍNH

DỮ LIỆU TĂNG CƯỜNG

Early stopping

– Là một kỹ thuật dừng vòng lặp của gradient descent sớm để tránh trường hợp lặp quá mức dẫn đến overfiting.



KHỞI TẠO TRỌNG SỐ

Chuẩn hoá dữ liệu

– Chuẩn hoá dữ liệu huấn luyện dựa vào

+ Trung bình (mean): $\mu = \frac{1}{m}\sum_{i=1}^{m} X^{(i)}$

+ Phương sai (variance): $\sigma = \frac{1}{m}\sum_{i=1}^{m} X^{(i)} \otimes\otimes 2$

– Lưu ý: Khi chuẩn hoá tập train, thì tập dev/test phải được thực hiện tương tự (cùng μ và σ).

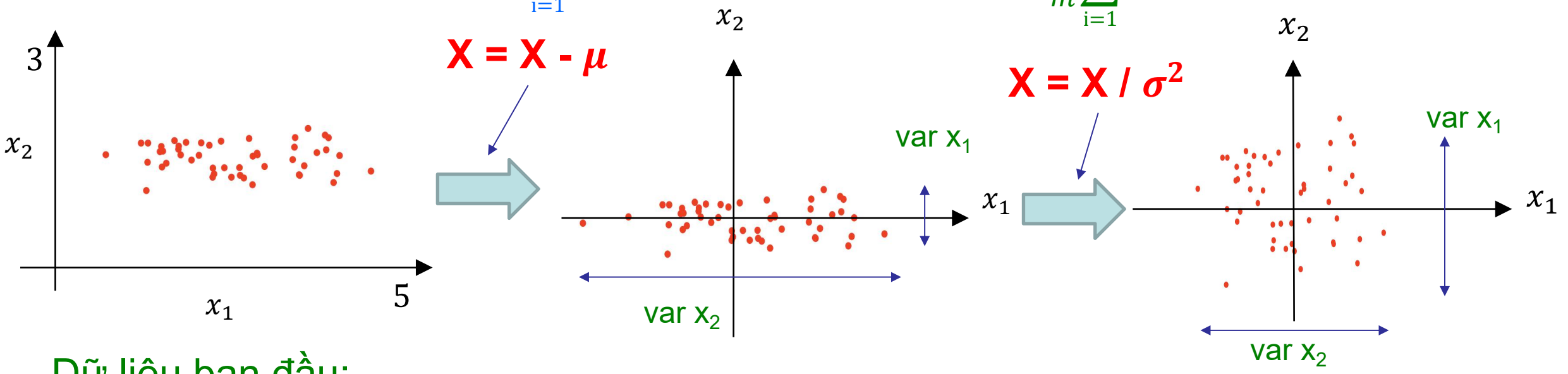
Chuẩn hoá dữ liệu (tt)

$$\mu = \frac{1}{m}\sum_{i=1}^{m} X^{(i)}$$

X = X - μ

$$\sigma = \frac{1}{m}\sum_{i=1}^{m} X^{(i)} \otimes\otimes 2$$

X = X / σ^2



Dữ liệu ban đầu:
X = [x_1, x_2]

var x_1 < var x_2

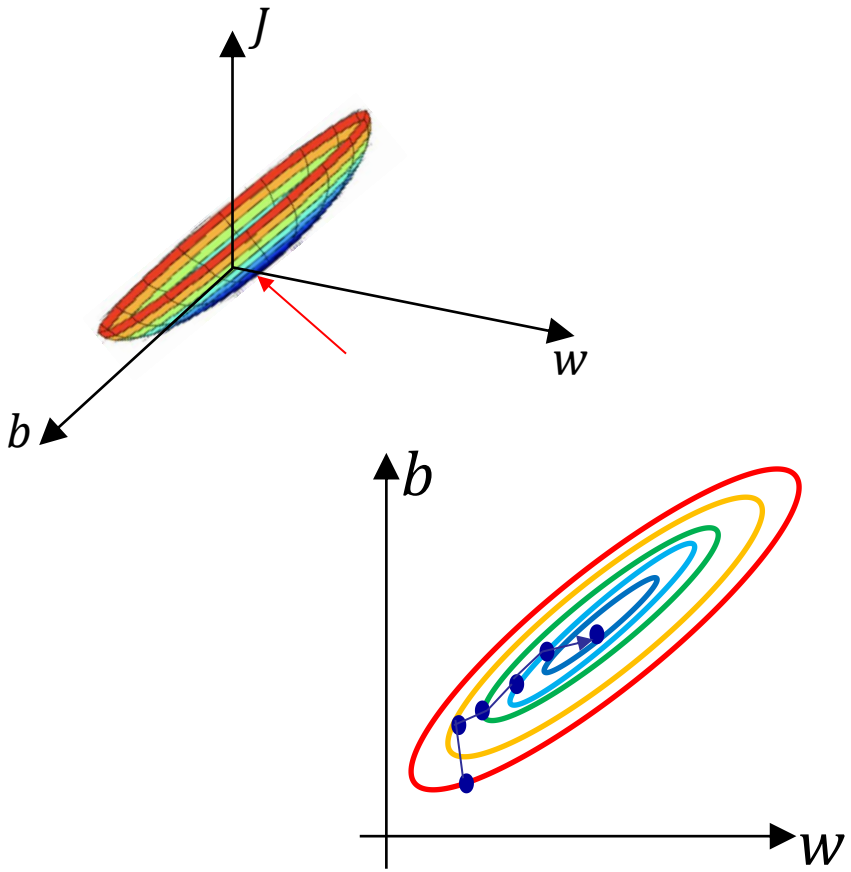
var x_1 ≈var x_2

Tại sao cần chuẩn hoá dữ liệu ?

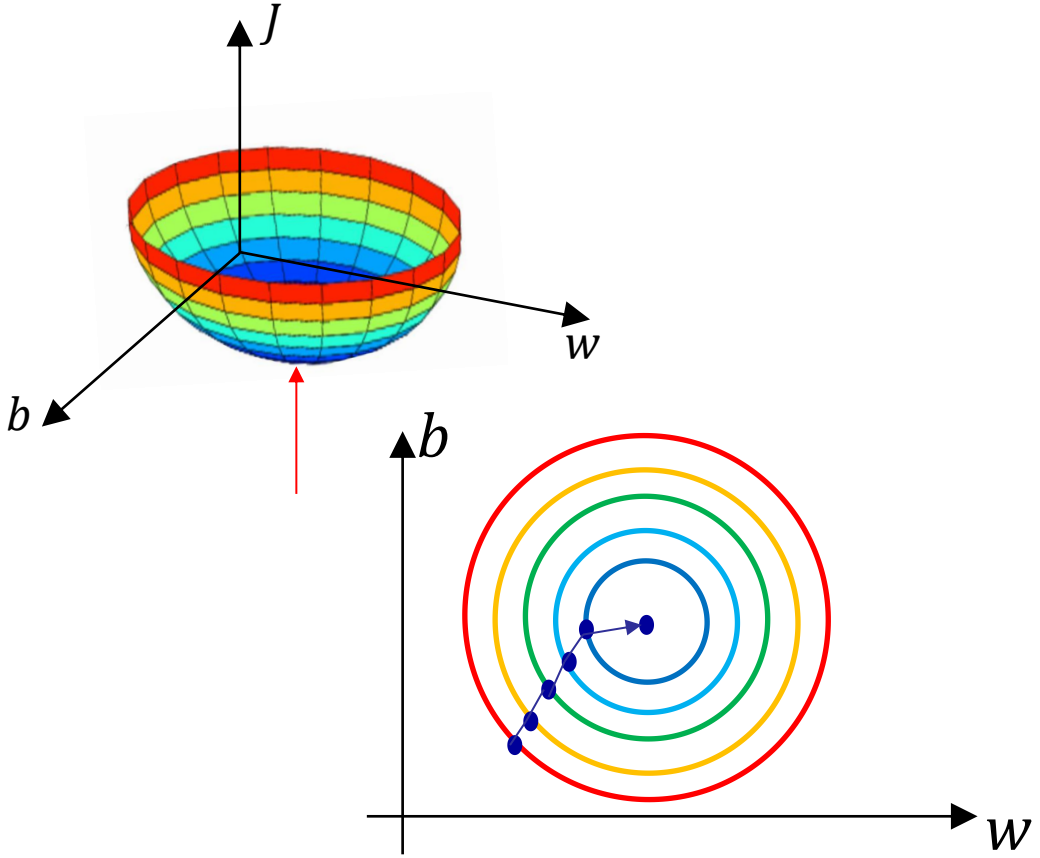
- Với dữ liệu đầu vào: $X = [x_1, x_2]$.
  - + x_1 có miền giá trị 1 đến 1000.
  - + x_2 có miền giá trị 1 đến 10.
  - ➔ giá trị w_1 và w_2 rất khác nhau.
- Nếu như không chuẩn hoá, thì giá trị của w_1 và w_2 sẽ có sự chênh lệch lớn ➔ quá trình tính toán trên hàm chi phí J chậm.
- Như vậy, chuẩn hoá dữ liệu sẽ giúp cho quá trình gradient descent được cải thiện.

Minh hoạ

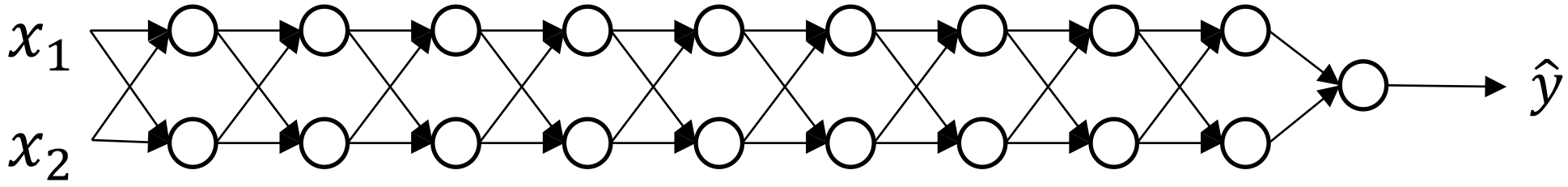
Unnormalized



Normalized



Vanishing/Exploding Gradient descent



$W^{[1]}$ $W^{[2]}$ $W^{[3]}$ $W^{[4]}$ ... $W^{[L]}$

$$\hat{y} = W^{[L]} W^{[L-1]} W^{[L-2]} \ldots\ldots W^{[3]} W^{[2]} W^{[1]} X$$

$$Z^{[1]} = W^{[1]} X \mid a^{[1]} = g(Z^{[1]})$$

$$Z^{[2]} = W^{[2]} * a^{[1]} \mid a^{[2]} = g(Z^{[2]})$$

$W^{[1]} * W^{[2]} * \ldots * W^{[L]}$ = **W mũ L**

Nhận xét

- Càng đi sâu xuống nhiều lớp, thì giá trị hàm kích hoạt (activation function) sẽ thay đổi như sau:
  + Nếu W càng lớn thì giá trị $\hat{y}$ càng lớn (exploding).
  + Nếu W càng nhỏ thì giá trị $\hat{y}$ càng nhỏ (vanishing).
- Nếu khởi tạo $W^{[L]} > I$, thì giá trị hàm kích hoạt càng lớn nếu càng đi về các lớp sau.
- Nếu khởi tạo $W^{[L]} < I$, thì giá trị hàm kích hoạt càng nhỏ dần nếu càng đi về các lớp sau.

(I là ma trận đơn vị)

Ví dụ

Exploding

$$W^{[L]} = \begin{pmatrix} 1.5 & 1 \\ 1 & 1.5 \end{pmatrix}$$

Giả sử mạng neural 10 lớp.

$$W^{[10]} \approx \begin{pmatrix} 1.5^{10} & 1 \\ 1 & 1.5^{10} \end{pmatrix}$$

$$\approx \begin{pmatrix} 57.66 & 1 \\ 1 & 57.66 \end{pmatrix}$$

Vanishing

$$W^{[L]} = \begin{pmatrix} 0.5 & 1 \\ 1 & 0.5 \end{pmatrix}$$

Giả sử mạng neural 10 lớp.

$$W^{[10]} \approx \begin{pmatrix} 0.5^{10} & 1 \\ 1 & 0.5^{10} \end{pmatrix}$$

$$\approx \begin{pmatrix} 0.00097 & 1 \\ 1 & 0.00097 \end{pmatrix}$$

Khởi tạo trọng số cho mạng neural để tránh Vanishing/exploding GD

Khởi tạo trọng số $W^{[l]}$:

$$W^{[1]} = np.random.randn(shape(W)) \otimes Variance$$

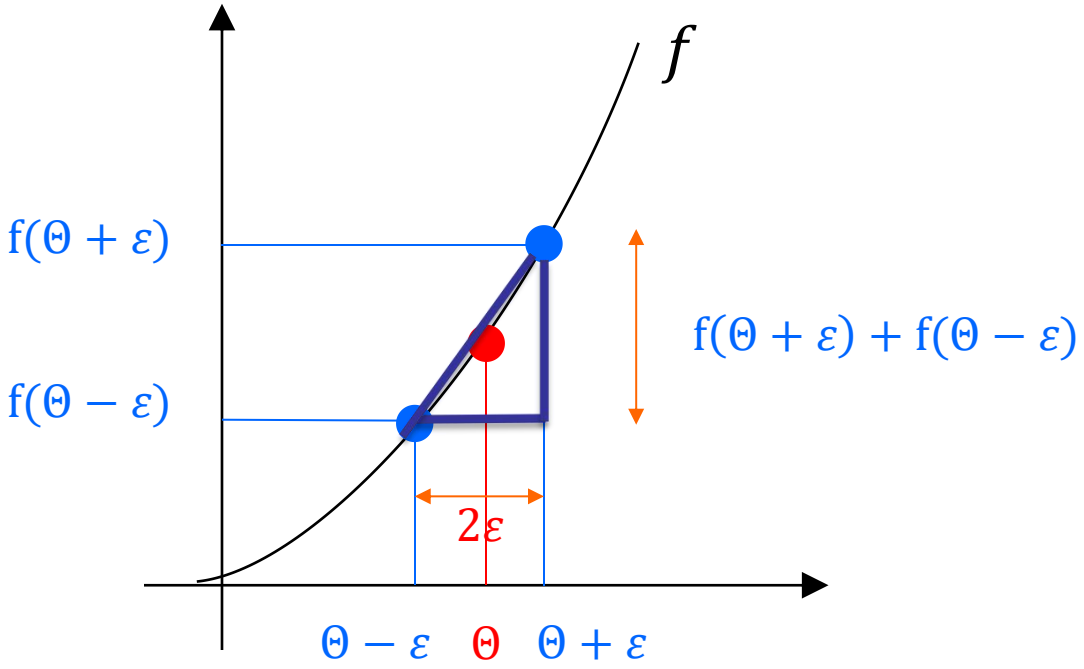
Làm cho trọng số W quá lớn hoặc quá nhỏ so với 1 (phương sai không quá lớn)

Variance cho ReLU: $np.\text{sqrt}\left(\frac{2}{n^{[l-1]}}\right)$

Variance cho TanH: $np.\text{sqrt}\left(\frac{1}{n^{[l-1]}}\right)$ *Xavier 's initialization*

Variance cho TanH (phiên bản khác): $np.\text{sqrt}\left(\frac{2}{n^{[l-1]} * n^{[l]}}\right)$ *Bengio et al.*

Gradient Checking – Lý thuyết



$$\frac{f(\Theta+\varepsilon)+f(\Theta-\varepsilon)}{2\varepsilon} \approx g(\Theta)$$

Gradient checking

- Lấy các trọng số $W^{[1]}$, $b^{[1]}$, ... $W^{[L]}$, $b^{[L]}$ và chuyển thành vector Θ.
  + $J(W^{[1]}, b^{[1]}, ... W^{[L]}, b^{[L]}) = J(\Theta)$.
- Lấy các trọng số $dW^{[1]}$, $db^{[1]}$, ... $dW^{[L]}$, $db^{[L]}$ và chuyển thành vector $d\Theta$.
  + $dJ(W^{[1]}, b^{[1]}, ... W^{[L]}, b^{[L]}) = dJ(\Theta)$.

Gradient Descent checking example

For i from 1 to L:

$$d\Theta_{Approx}[i] = \frac{J(\Theta_1, \Theta_2, ..., \Theta_i + \varepsilon) - J(\Theta_1, \Theta_2, ..., \Theta_i - \varepsilon)}{2\varepsilon} \approx d\Theta[i]$$

$$\partial = \frac{||d\Theta_{Approx}^{[i]} - d\Theta^{[i]}||_2}{||d\Theta_{Approx}^{[i]}||_2 + ||d\Theta^{[i]}||_2}$$

If $\partial \approx 10^{-7}$ then Greate gradient descent

else if $\partial \approx 10^{-7}$ then OK! But some bug happen!

else if $\partial \approx 10^{-3}$ then Bug everywhere

Một vài lưu ý với Gradient checking

- Chỉ dùng cho debug, không dùng để huấn luyện.
  + Tính $d\Theta_{Approx}^{[i]}$ tốn rất nhiều tài nguyên.
- Nếu gradient descent trả về bug, thì kiểm tra lại các thành phần của layer thứ i đó.
  + Các thành phần như: W, b, dW, db.
- Nhớ sử dụng regurlazation nếu như hàm chi phí có sử dụng.
- Không hoạt động với kỹ thuật Dropout.
- Thực hiện grad check ngay sau khi khởi tạo ngẫu nhiên trọng số cho W và b, và sau khi huấn luyện qua một số lần.

TỔNG KẾT

1. Các kỹ thuật chia dữ liệu để huấn luyện mạng neural.

   Chia dữ liệu thành train/dev/test (8-1-1 với dữ liệu nhỏ và 9-0.5-0.5 với dữ liệu lớn).

2. Các loại loại trong Deep learning.

   High bias (underfiting) và high variance (overfiting).

3. Các kỹ thuật giảm overfiting (high variance).

   Chuẩn hoá mạng (regurlazation), tăng dữ liệu (data augmentation), early stopping.

4. Các kỹ thuật tăng cường tốc độ cho gradient descent.

   Normalizing input data, Dropout, Parameter Initialization, Grad checking.

TÀI LIỆU THAM KHẢO

1. Khoá học *Neural Network and Deep learning*, deeplearning.ai.
2. Ian Goodfellow, Yoshua Bengio, Aaron Courvile, *Deep learning*, MIT Press, 2016.
3. Andrew Ng., *Machine Learning Yearning*. Link: https://www.deeplearning.ai/machine-learning-yearning/
4. Vũ Hữu Tiệp, *Machine Learning cơ bản*, NXB Khoa học và Kỹ thuật, 2018.