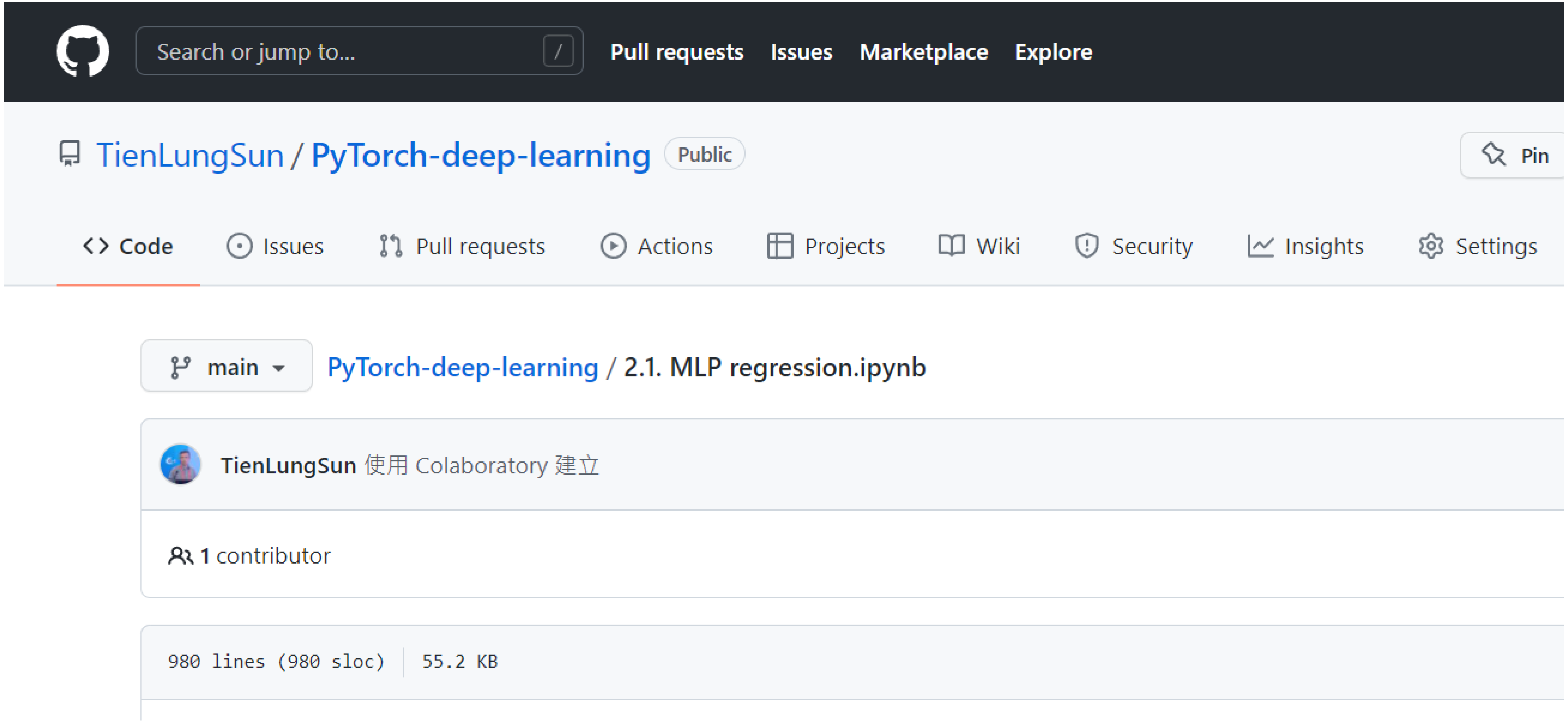
Regression

2.1 MLP regression.ipynb



How machine learns from data?

- Regression
- Classification

$$y = f(x)$$

- Define a function to be learned: $y = f(x)$
- Define a loss function $\mathcal{L}$ to describe the error between $\hat{y} = f(x)$ and y
- Find the optimal parameters of f that minimize $\mathcal{L}$

Statistics vs Machine Learning

Statistics versus machine learning

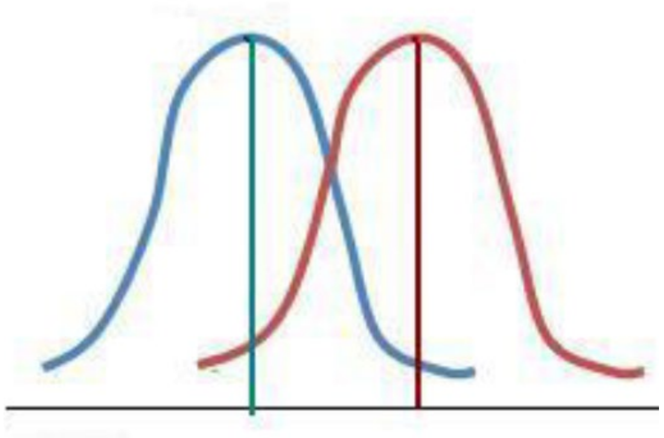
Statistics draws population inferences from a sample, and machine learning finds generalizable predictive patterns.

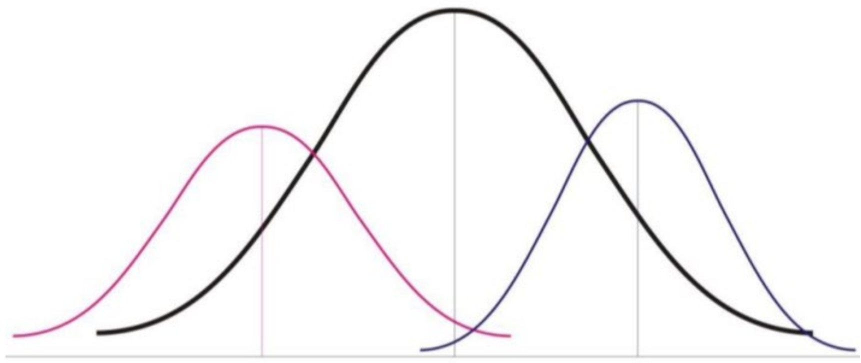
$$y = f(x)$$

Ij, H. (2018). Statistics versus machine learning. *Nat Methods*, *15*(4), 233.

Statistics draw population inferences

動作	range (delta D)
A1	8.20
A1	6.42
A1	8.26
A1	14.55
A1	11.70
A1	11.75
A1	12.72
A1	11.96
A1	6.03
A1	9.20
A1	14.25
A1	13.93
A2	14.17
A2	19.91
A2	9.40
A2	8.48
A2	10.61
A2	13.86
A2	7.15
A2	7.66
A2	6.09
A2	13.18
A2	9.15
A2	13.29





Evaluation of generalizable prediction performance with test data

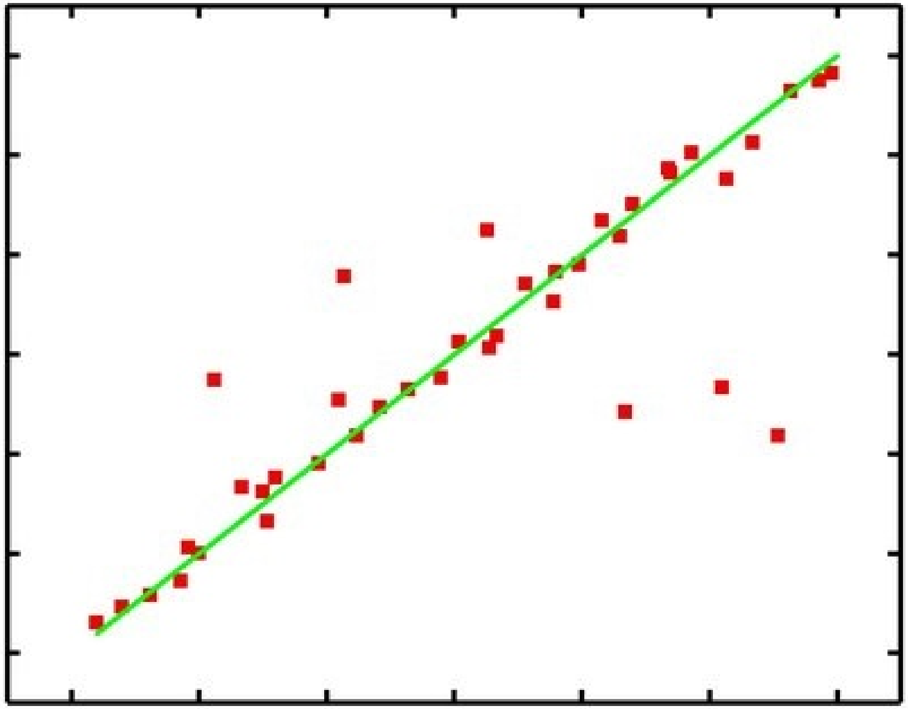
Split input data to train and test data

In [5]:

```
from sklearn.model_selection import train_test_split
trainX, testX, trainY, testY = train_test_split(numpyX, numpyY, test_size=0.20, random_state=0)
print(trainX.shape, testX.shape, trainY.shape, testY.shape)
```

```
(1600, 7) (400, 7) (1600, 1) (400, 1)
```

Statistics vs Machine Learning



(b) Linear Regression

Statistics – R^2 of the variance explained

ML – MSE, MAE of test data which are unseen by the model

Input data

y	x1	x2	...
0.578			
0.64			
-0.96			
0.23			
…			

$$y = 0.323{x_1}^2 + 0.586x_1x_2 + 0.4x_3 + 0.8972{x_5}^3 + 0.267{x_3}^2x_5x_6 + 0.78{x_7}^2$$

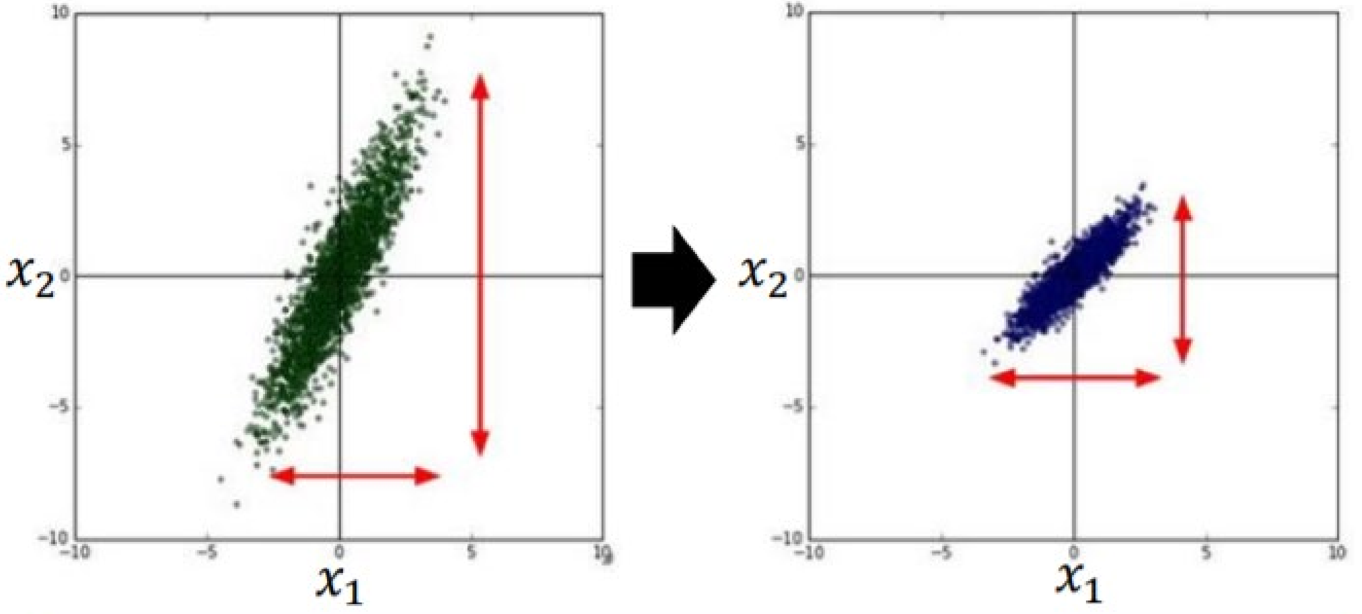
$$x_1 \sim \mathcal{N}(0.5, 0.3)$$

$$x_2 \sim \mathcal{N}(0.8, 0.5)$$

$\dots$

Feature scaling

$$y = b + w_1 x_1 + w_2 x_2$$

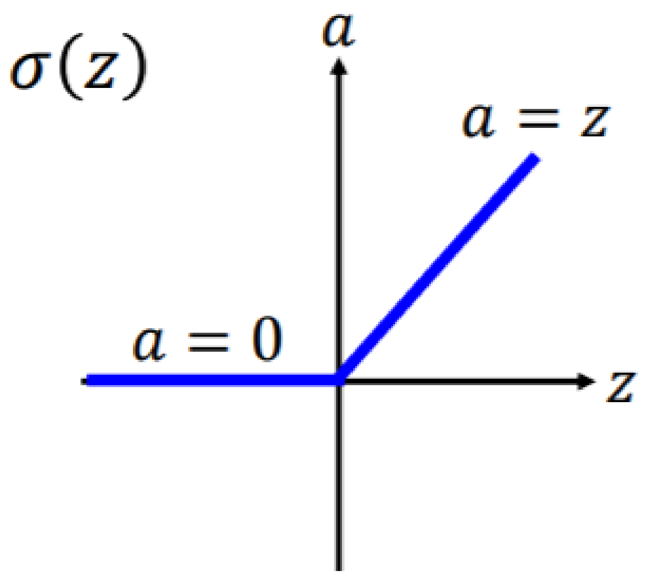


Make different features have the same scaling

Reference: 李弘毅 ML Lecture 3-1 https://youtu.be/yKKNr-QKz2Q

ReLU activation function

- Rectified Linear Unit (ReLU)



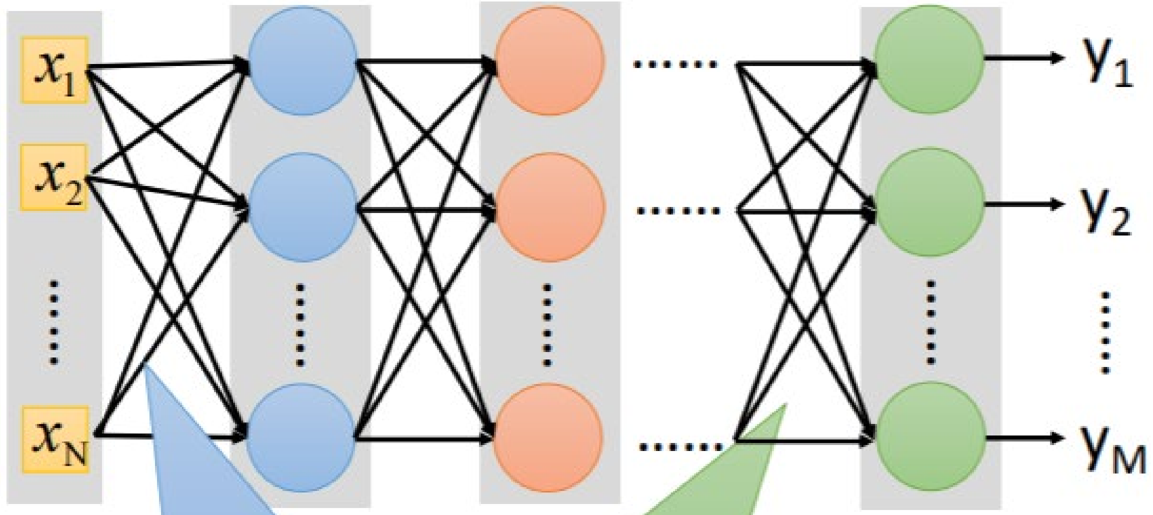
[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Vanishing gradient problem



Smaller gradients

Learn very slow

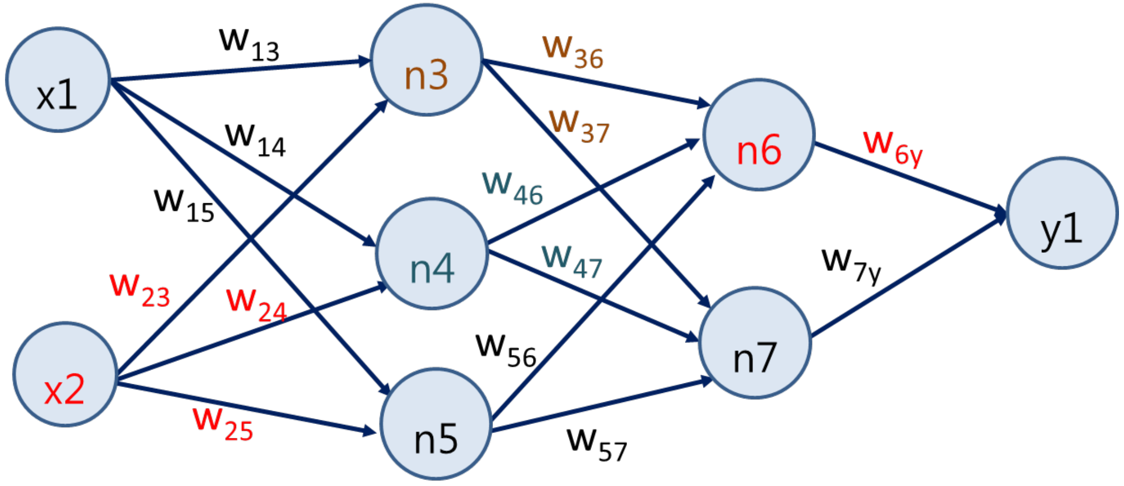
Almost random

Larger gradients

Learn very fast

Already converge

based on random!?



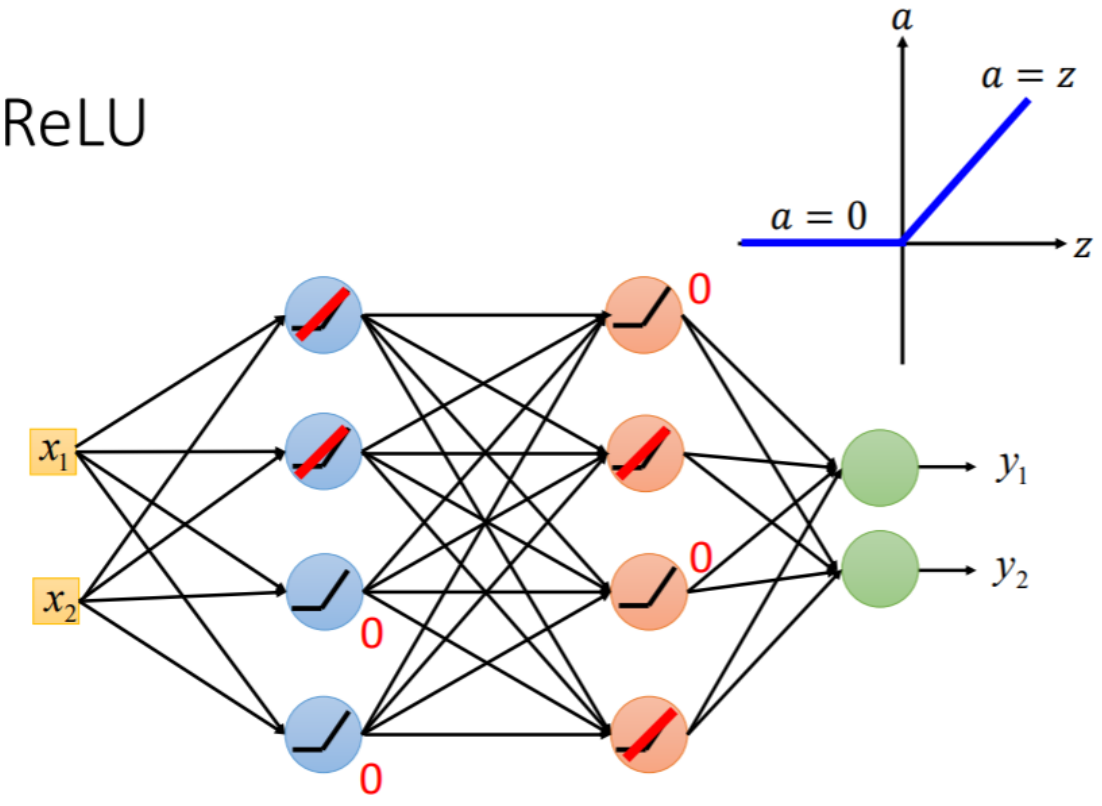
$$\frac{\partial L}{\partial w_{6y}} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial w_{6y}}$$

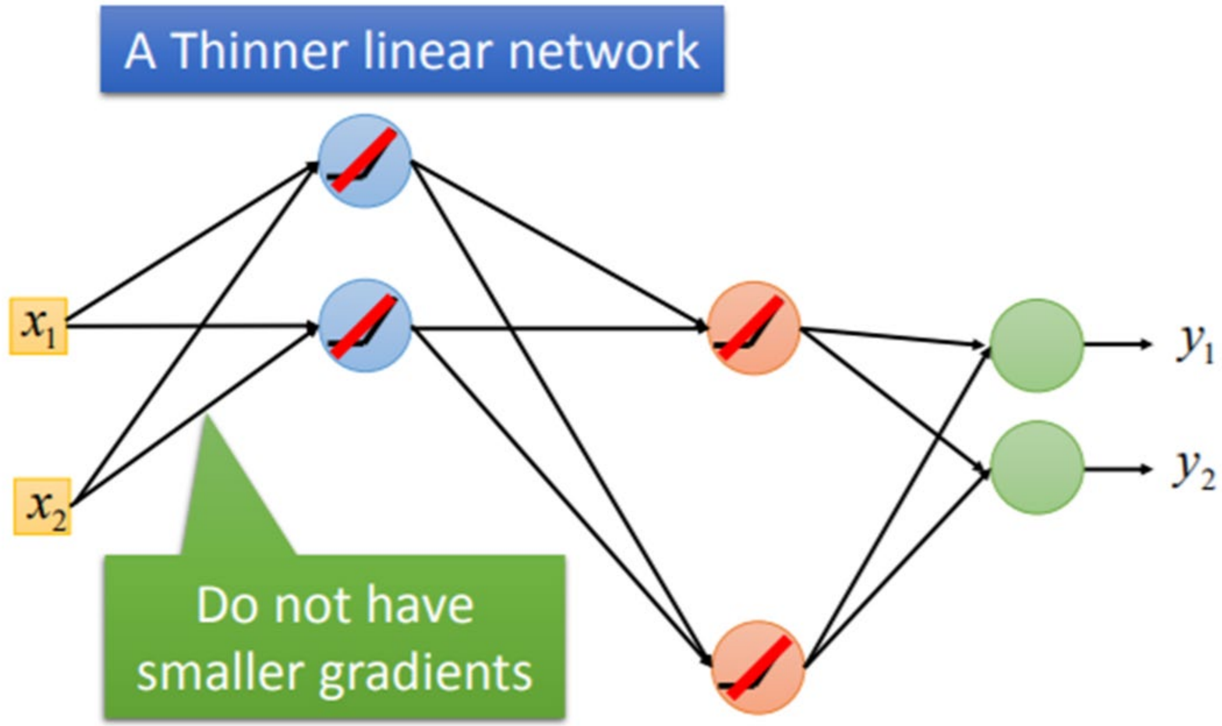
$$\frac{\partial L}{\partial w_{57}} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial n_7}\frac{\partial n_7}{\partial w_{57}}$$

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

ReLU results in a thinner linear network

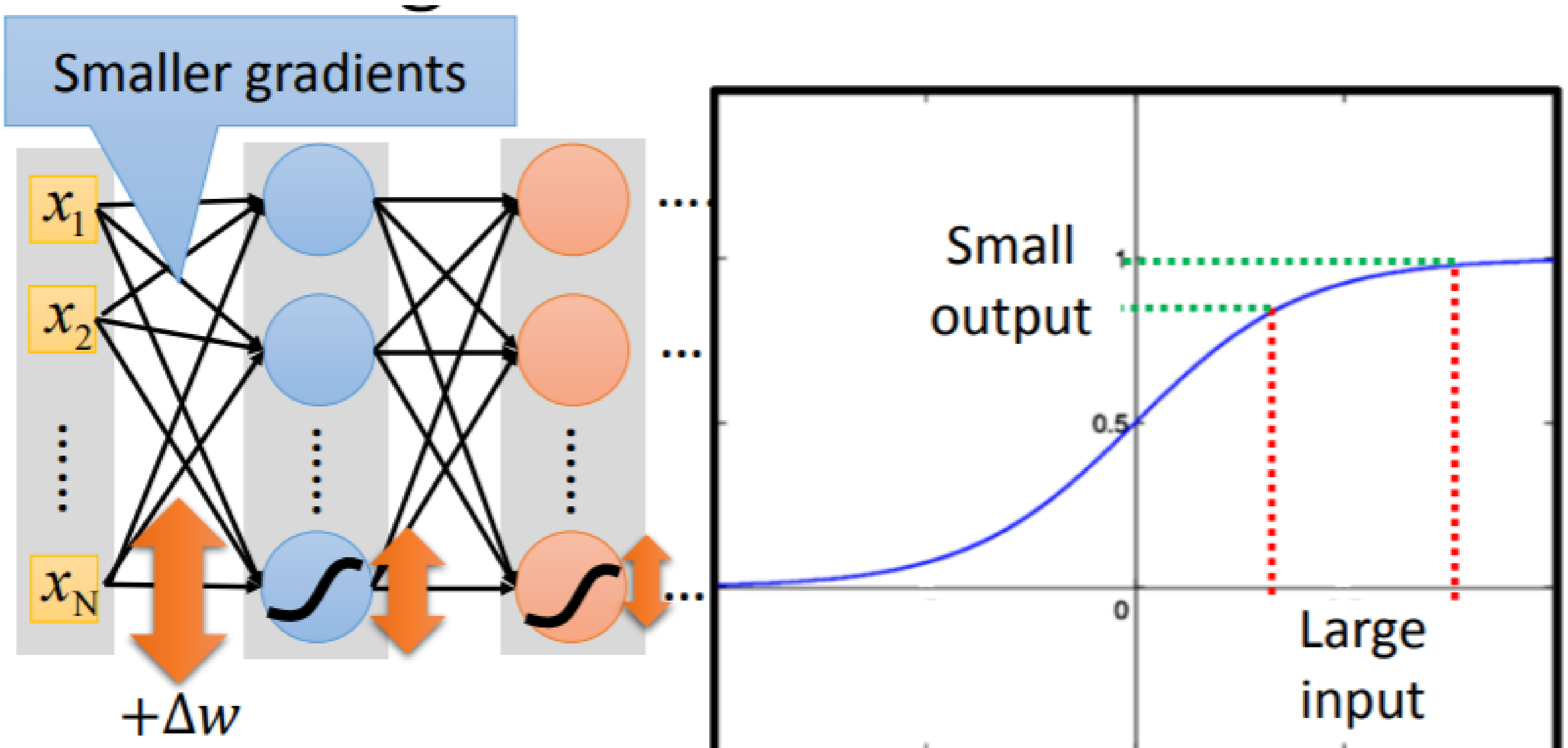
ReLU





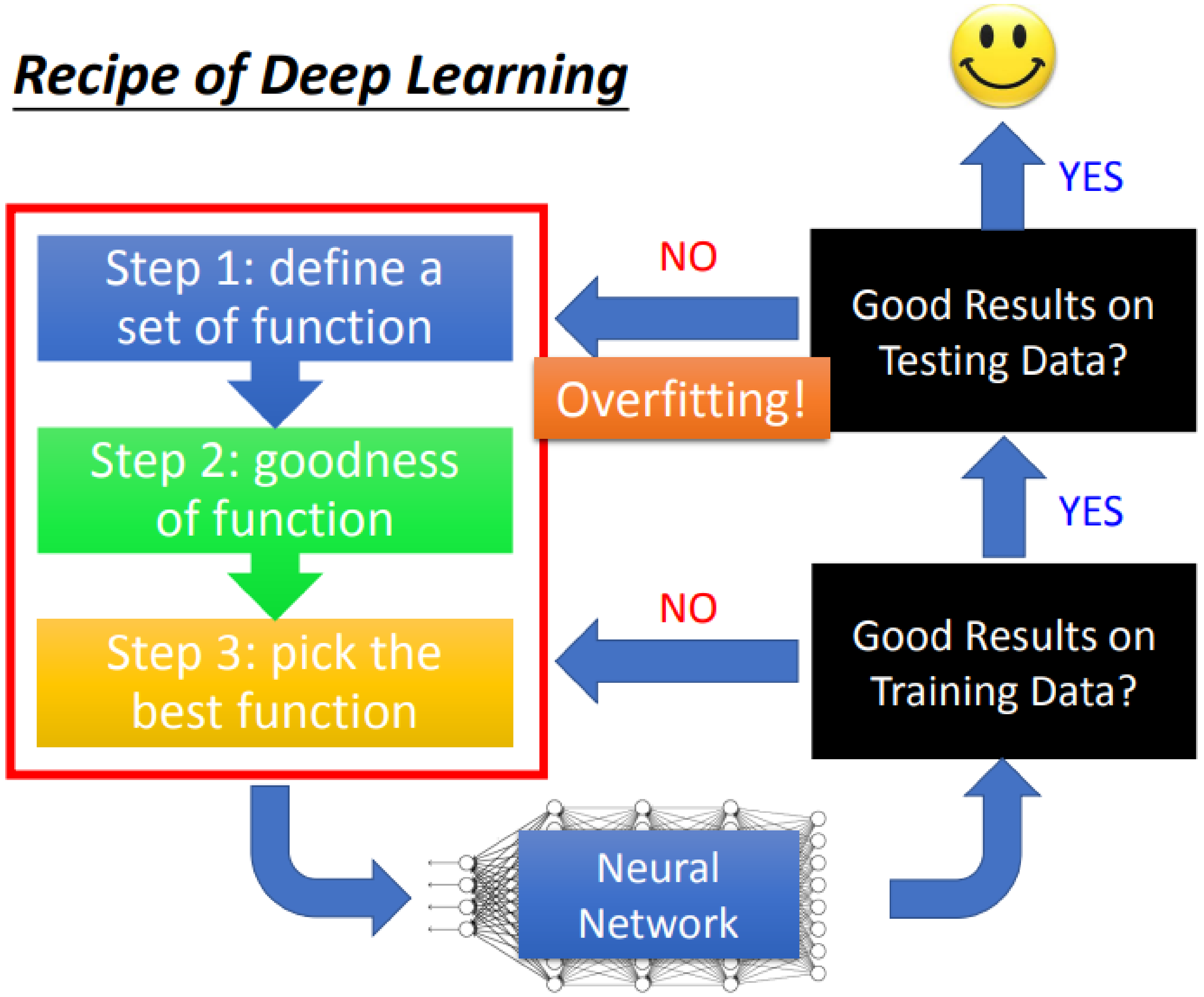
Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Sigmoid is hard to get the power of deep



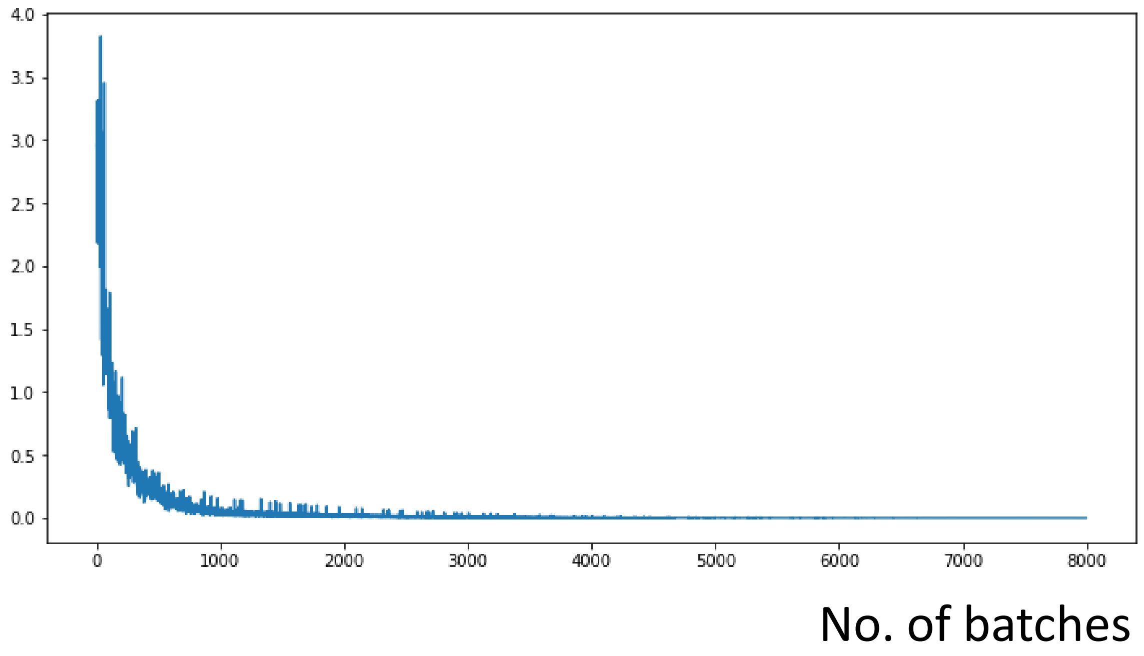
Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

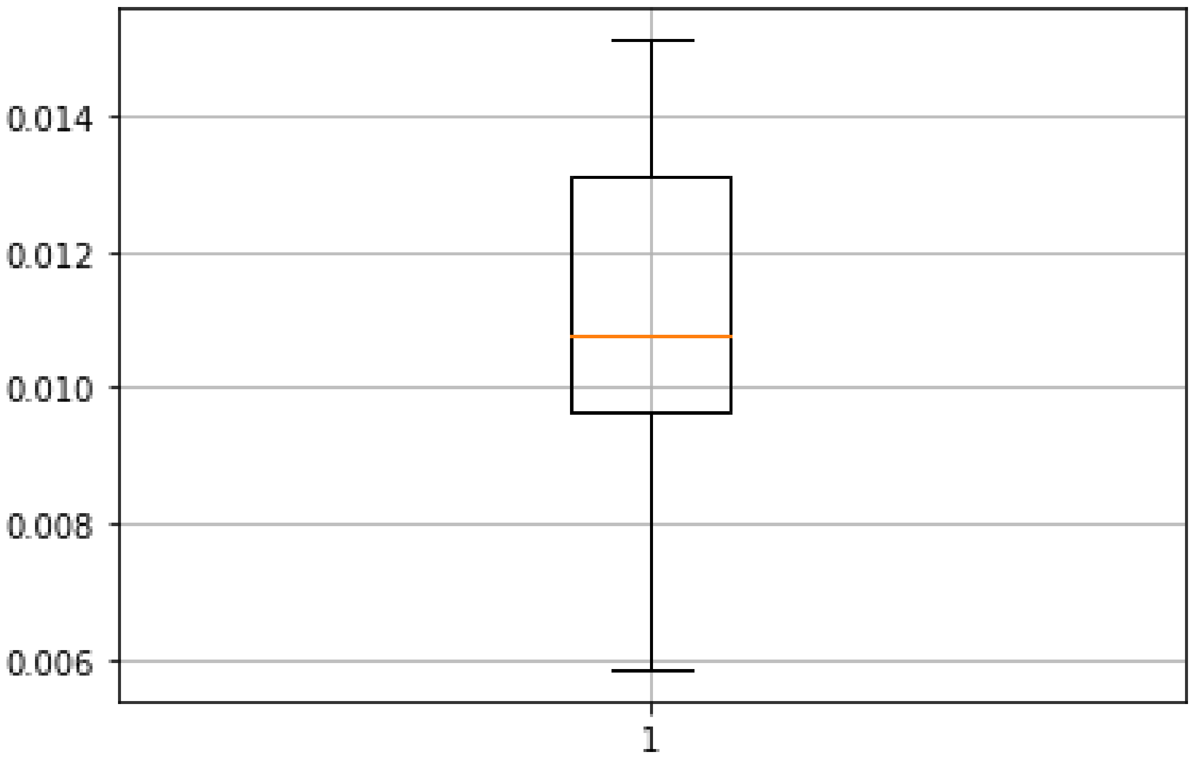
ML finds generalizable predictive patterns



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

ML finds generalizable predictive patterns





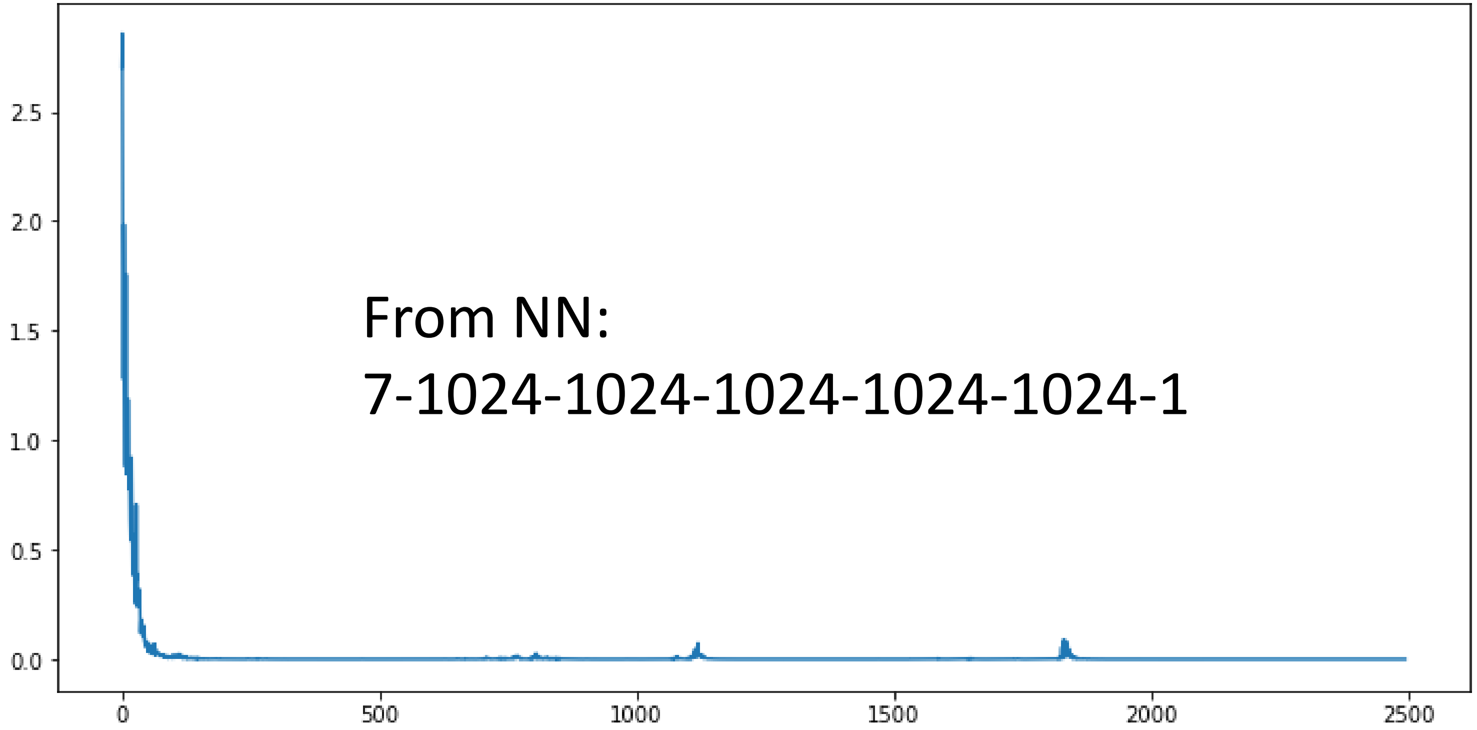
Good results on training data?

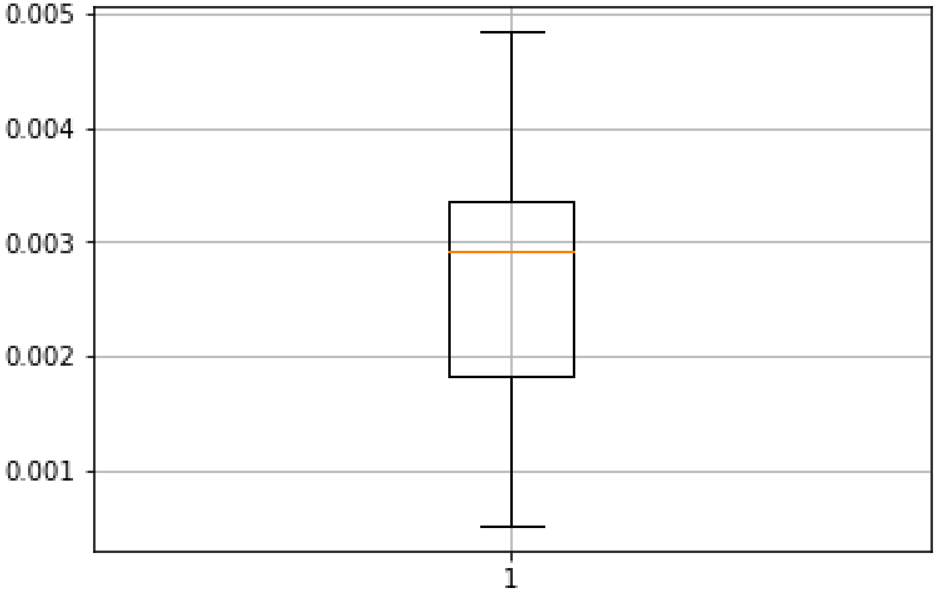
Good results on test data?

Try different NNs

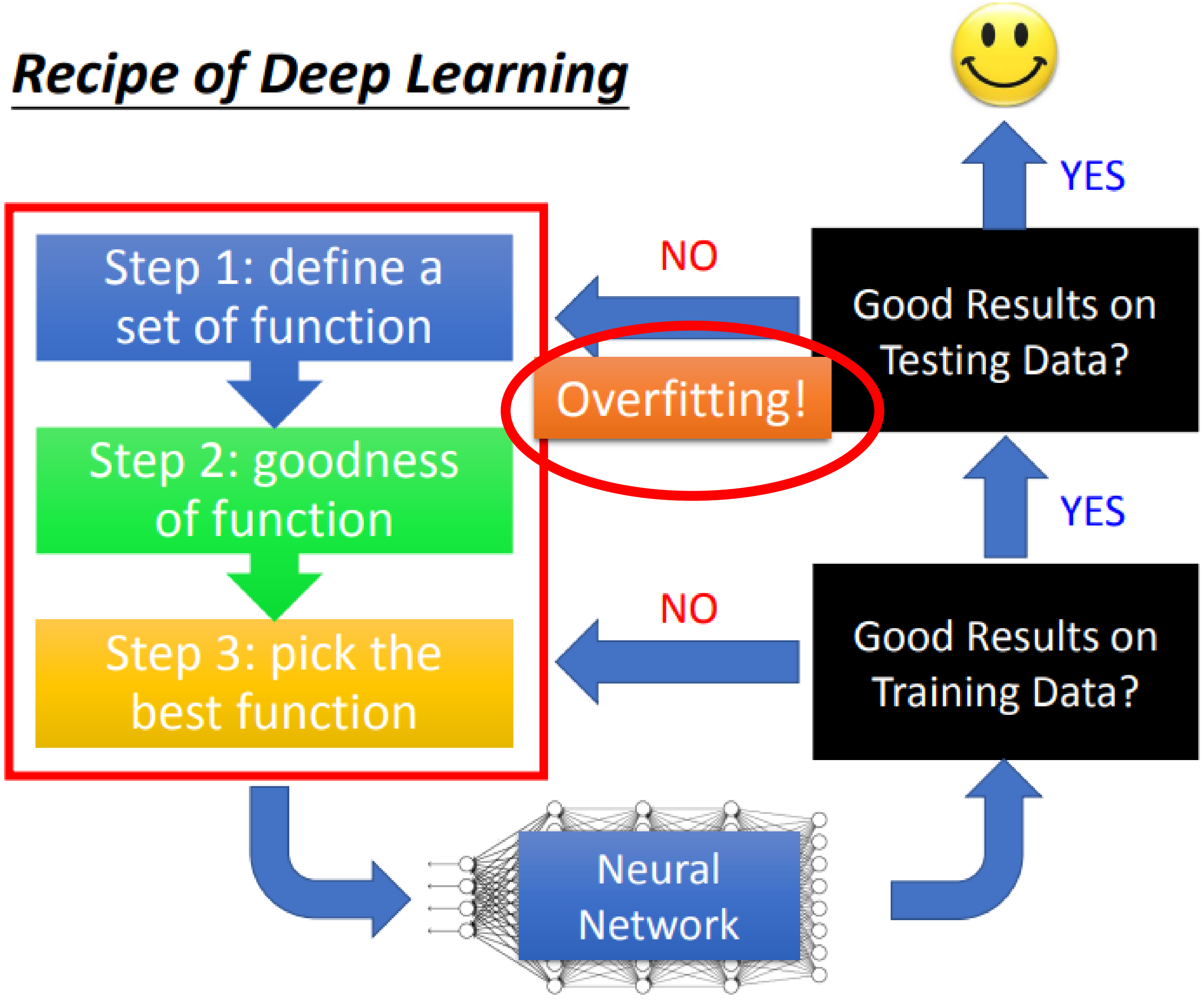
- Good results on training data?
- Good results on test data?

7-56-56-1 (current one)
7-256-256-256-1
7-512-512-512-512-1
7-1024-1024-1024-1024-1024-1



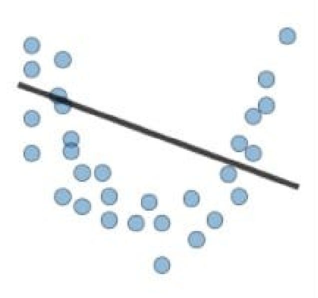
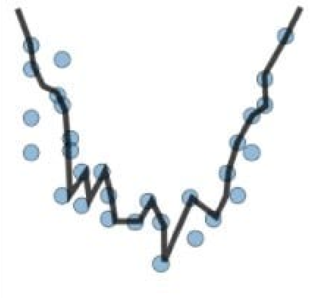
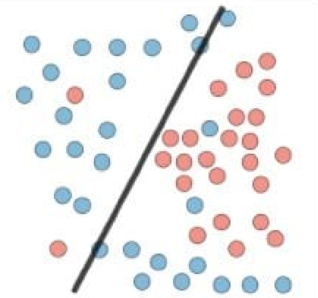
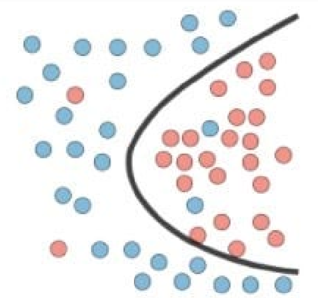
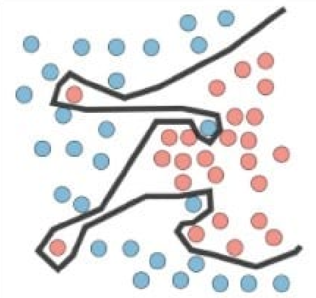
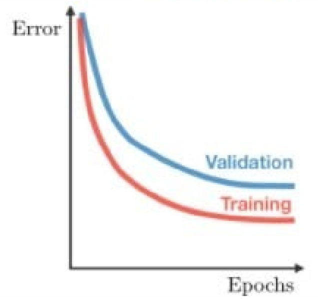


Overfitting



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Overfitting

	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	• Complexify model • Add more features • Train longer		• Perform regularization • Get more data

https://www.facebook.com/buggyprogrammers/photos/a.113230983855158/213140590530863/

Overfitting

2.2. Overfitting.ipynb

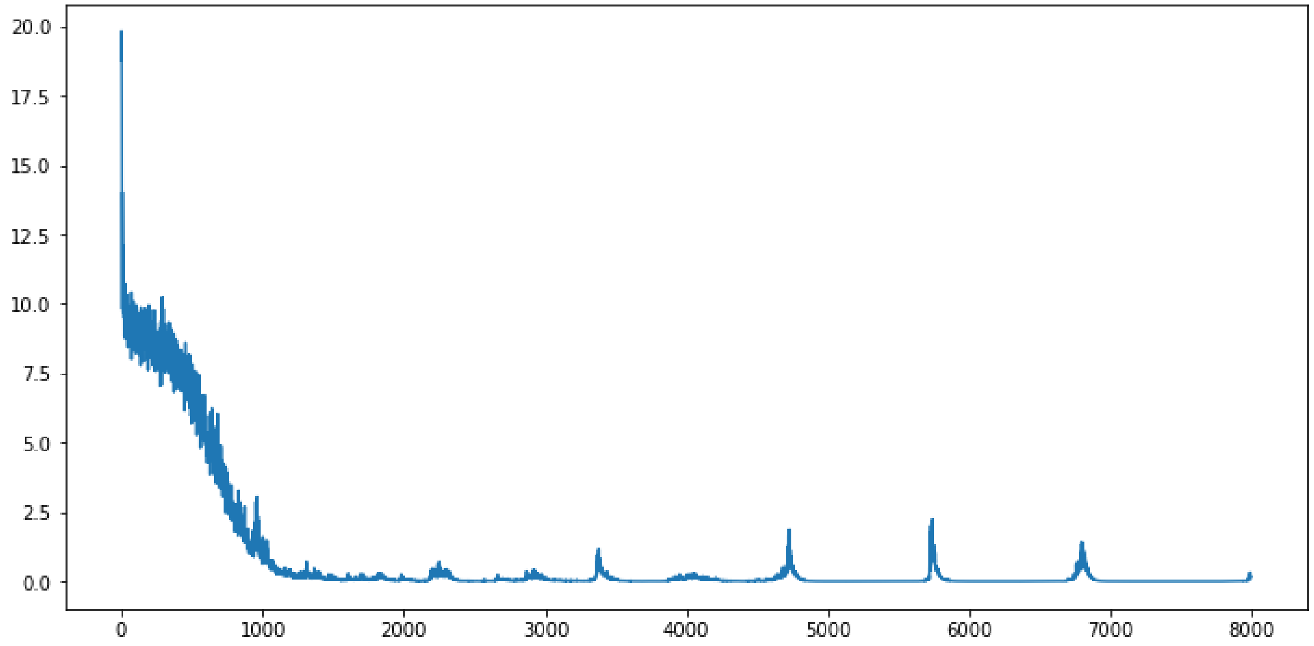
Add noises to y

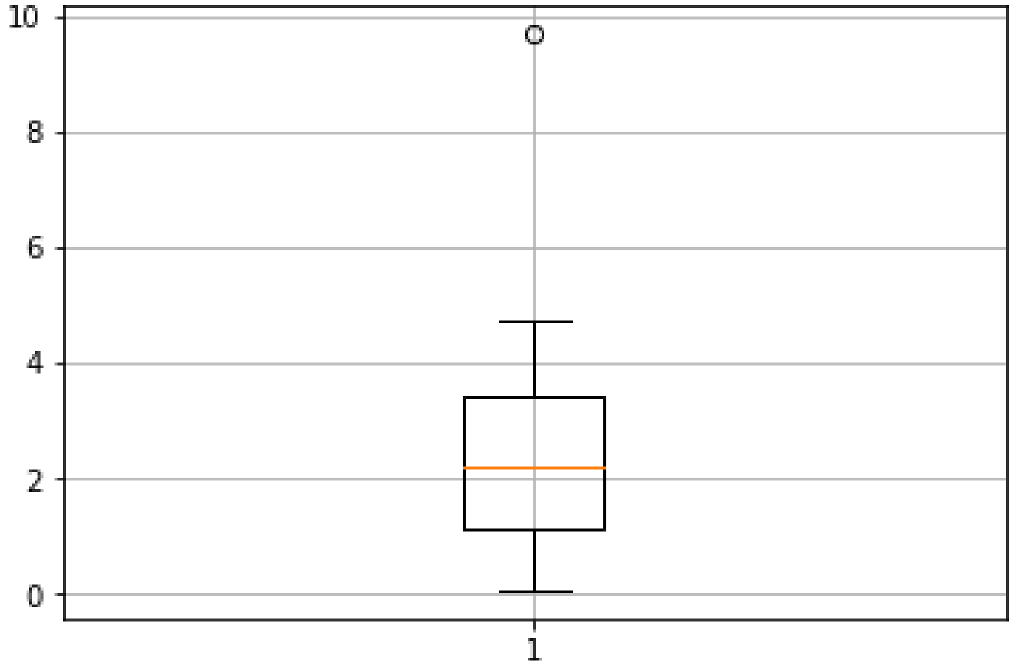
$$y = 0.323{x_1}^2 + 0.586x_1x_2 + 0.4x_3 + 0.8972{x_5}^3 + 0.267{x_3}^2x_5x_6 + 0.78{x_7}^2 + \mathcal{N}(2,3)$$

Use a complicated NN

7-1024-1024-1024-1024-1024-1

Overfitting

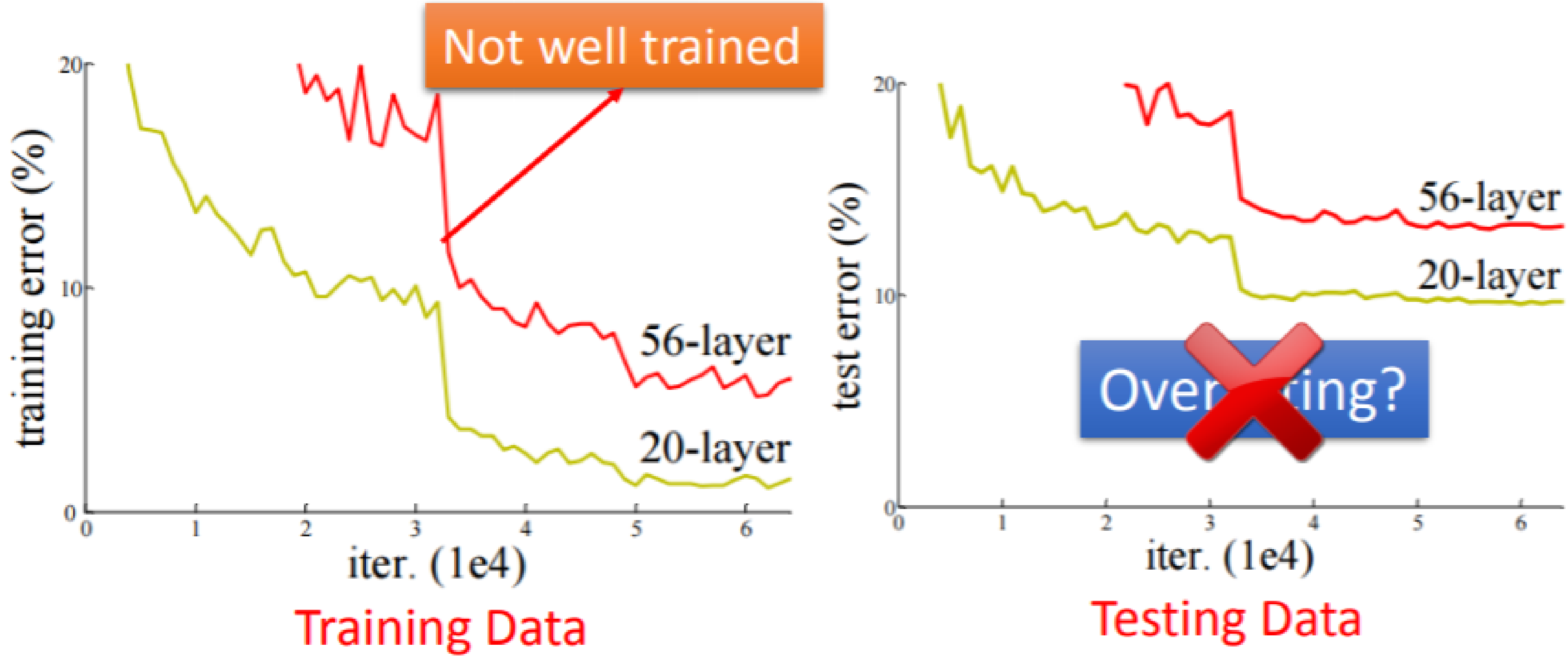




Good results on training data

Bad results on test data

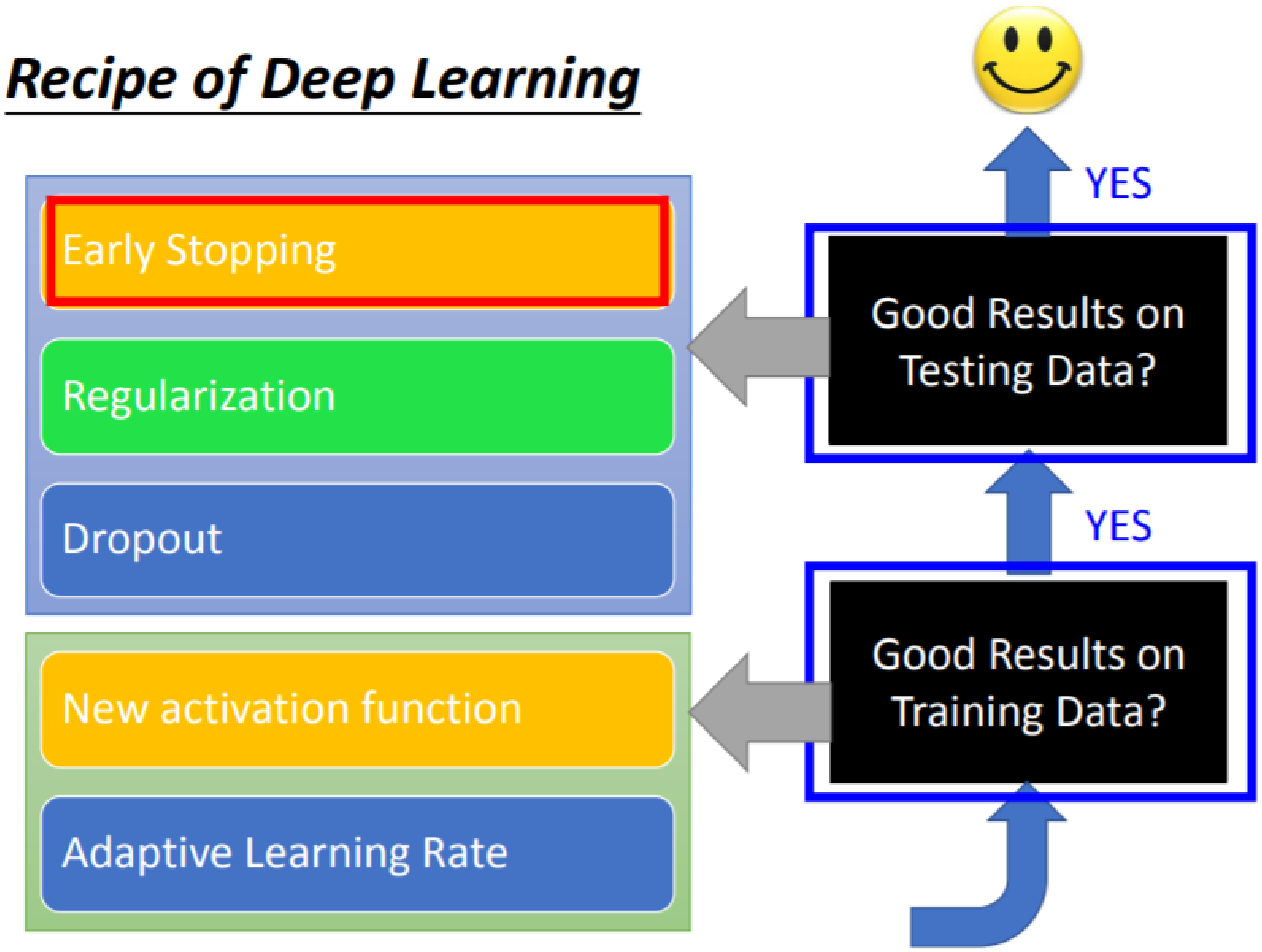
Do not always blame overfitting



Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385

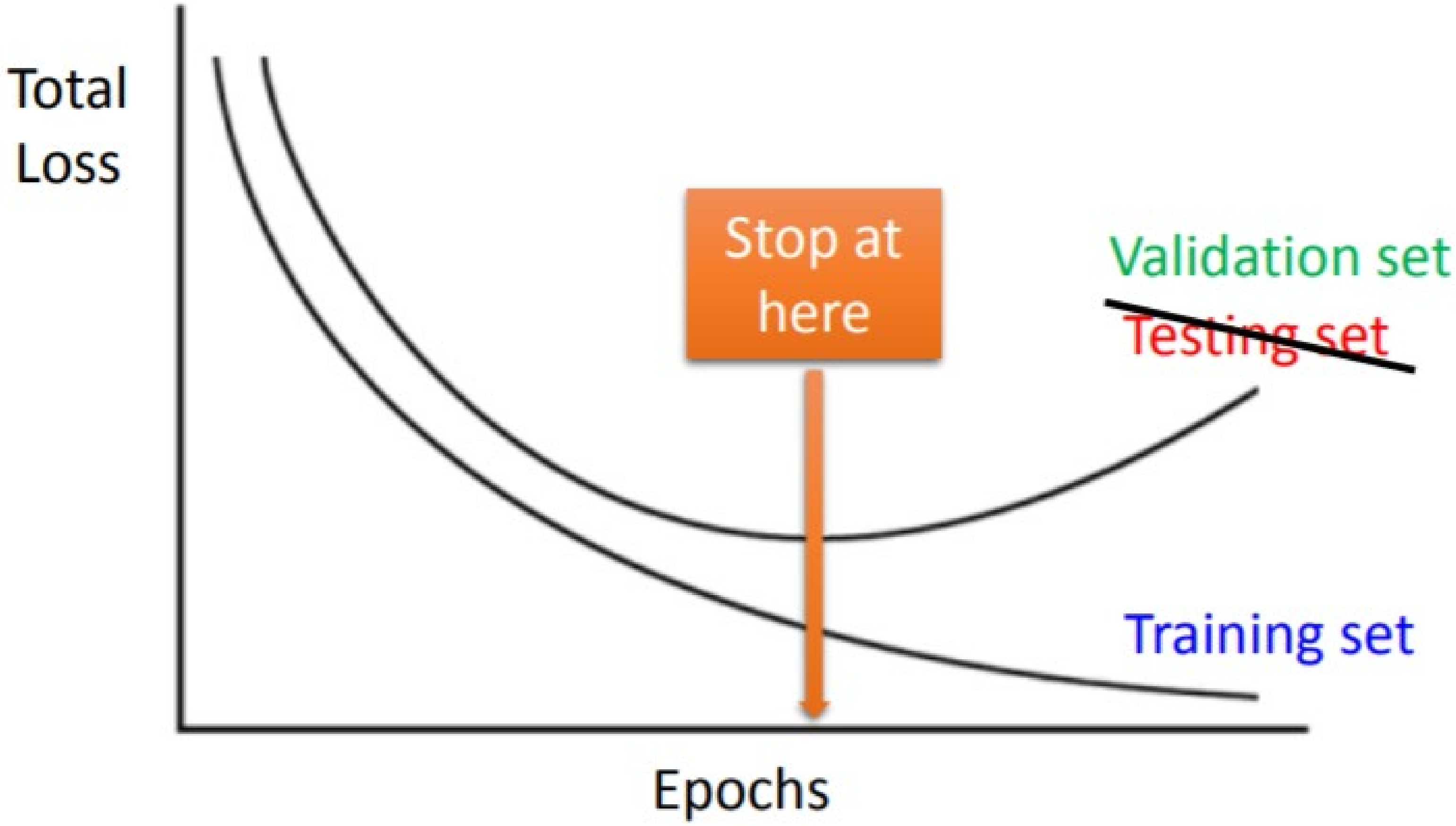
Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

What to do if overfitting?



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Early stopping



Keras: http://keras.io/getting-started/faq/#how-can-i-interrupt-training-when-the-validation-loss-isnt-decreasing-anymore

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Regularization – L2

- Find a set of weight not only minimizing original cost but also close to zero

$$\mathrm{L}'(\theta) = \underbrace{L(\theta)}_{\text{Original loss}} + \lambda \frac{1}{2} \underline{\|\theta\|_2} \rightarrow \text{Regularization term}$$

$$\theta = \{w_1, w_2, \ldots\}$$

Original loss

$$L(\theta) = \sum_{n=1}^{N} (\hat{y}^n - y^n)^2$$

L2 regularization:

$$\|\theta\|_2 = (w_1)^2 + (w_2)^2 + \ldots$$

(usually not consider biases)

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

L2 regularization

$$L'(\theta) = L(\theta) + \lambda \frac{1}{2}\|\theta\|_2$$ Gradient: $$\frac{\partial L'}{\partial w} = \frac{\partial L}{\partial w} + \lambda w$$

Update: $$w^{t+1} \rightarrow w^t - \eta \frac{\partial L'}{\partial w} = w^t - \eta\left(\frac{\partial L}{\partial w} + \lambda w^t\right)$$

$$= \underbrace{(1 - \eta\lambda) w^t}_{\text{Closer to zero}} - \eta \frac{\partial L}{\partial w}$$

Weight Decay

Vanilla gradient decent update $$w^{t+1} \rightarrow w^t - \eta \frac{\partial L}{\partial w}$$

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

L1 regularization

L1 regularization:

$$\|\theta\|_1 = |w_1| + |w_2| + \ldots$$

$$\mathrm{L}'(\theta) = L(\theta) + \lambda \frac{1}{2}\|\theta\|_1 \qquad \frac{\partial \mathrm{L}'}{\partial w} = \frac{\partial \mathrm{L}}{\partial w} + \lambda\,\mathrm{sgn}(w)$$

Update:

$$w^{t+1} \to w^t - \eta \frac{\partial \mathrm{L}'}{\partial w} = w^t - \eta\left(\frac{\partial \mathrm{L}}{\partial w} + \lambda\,\mathrm{sgn}(w^t)\right)$$

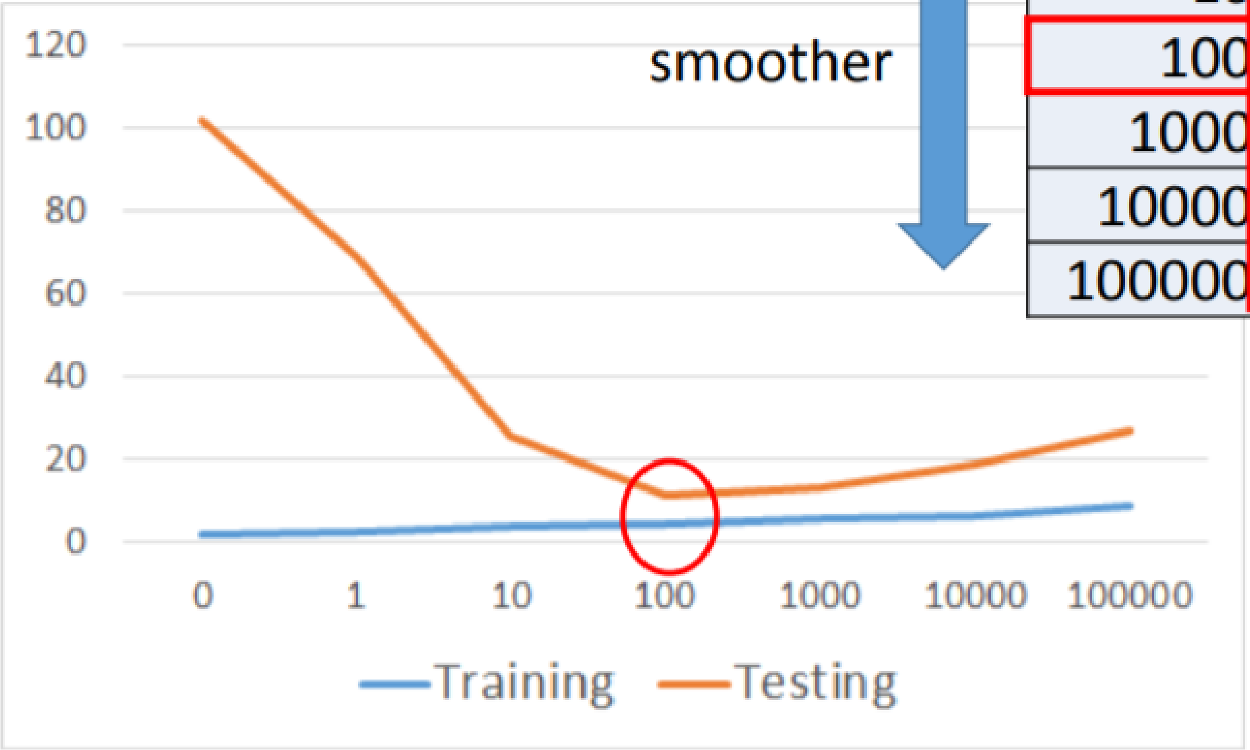
$$= w^t - \eta \frac{\partial \mathrm{L}}{\partial w} - \underline{\eta\lambda\,\mathrm{sgn}(w^t)}$$ Always delete

$$= (1 - \eta\lambda) w^t - \eta \frac{\partial \mathrm{L}}{\partial w}$$ …… L2

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Regularization

Regularization



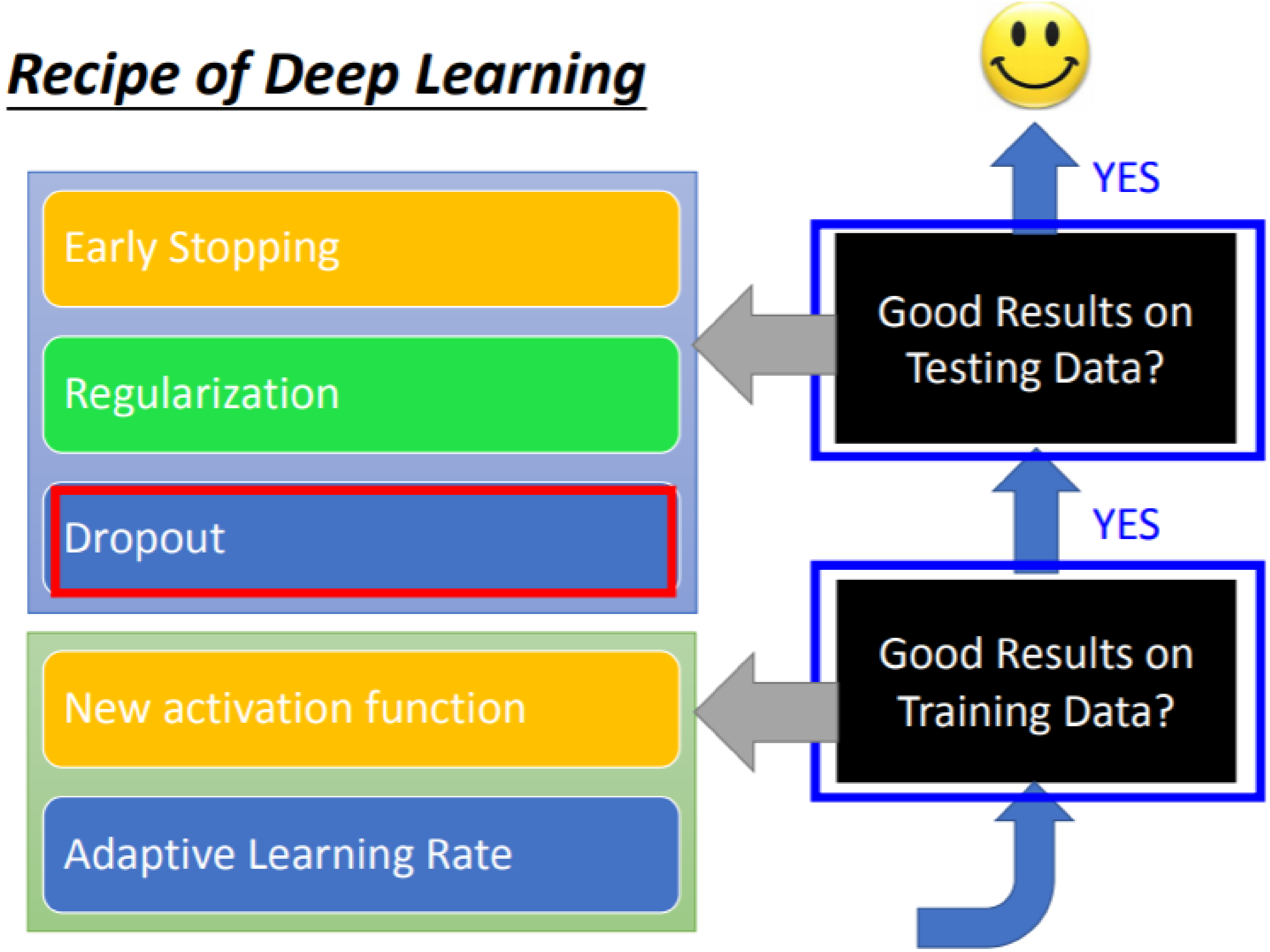
λ	Training	Testing
0	1.9	102.3
1	2.3	68.7
10	3.5	25.7
100	4.1	11.1
1000	5.6	12.8
10000	6.3	18.7
100000	8.5	26.8

How smooth?

Select λ obtaining the best model

Reference: 李弘毅 ML Lecture 1 https://youtu.be/CXgbekl66jc

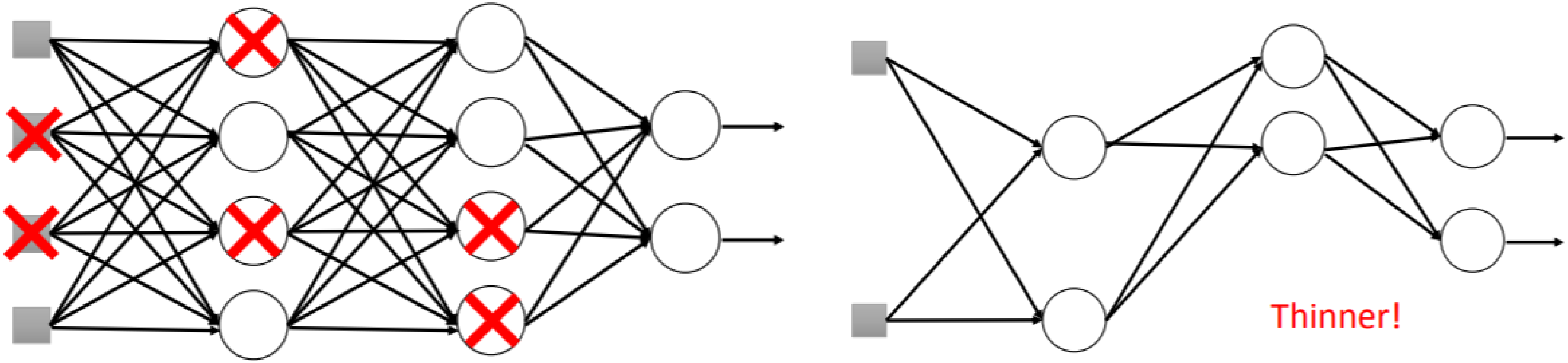
Drop out



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Drop out

Each time before updating θ, each neuron has $p\%$ to dropout. So the *NN* structure is changed (become thinner). That is, for each mini-batch, we resample the dropout neurons.

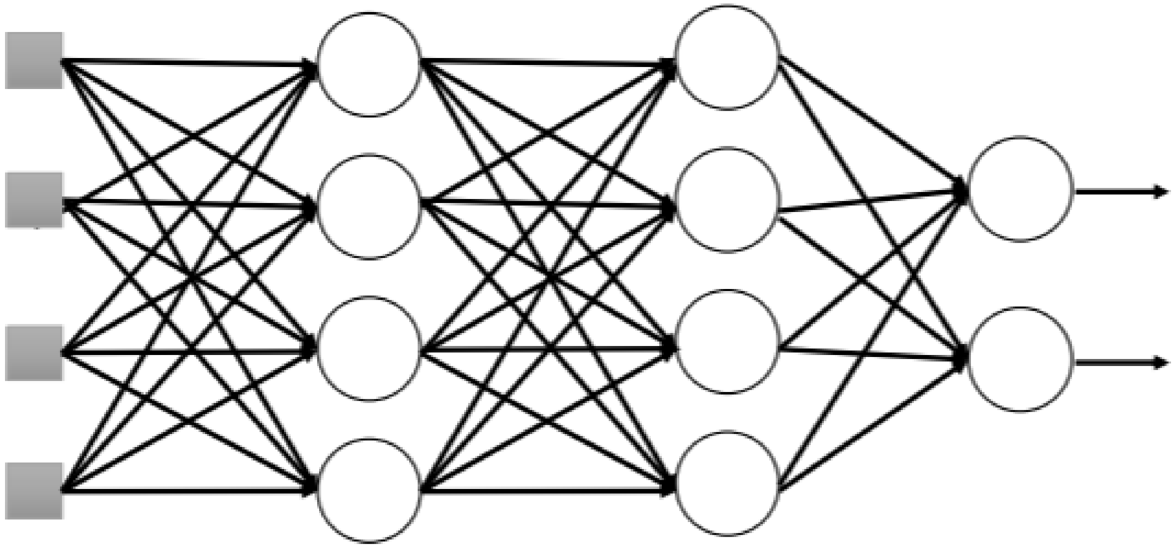


Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Drop out

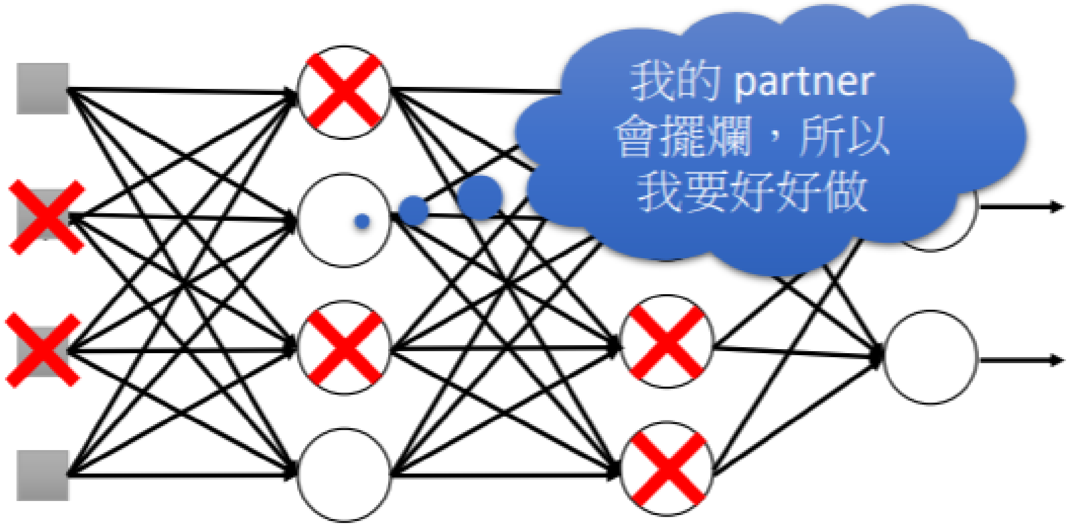
No neuron drop out at test stage. All weights time $1 - p\%$

Testing:



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Why drop out makes NN perform better?

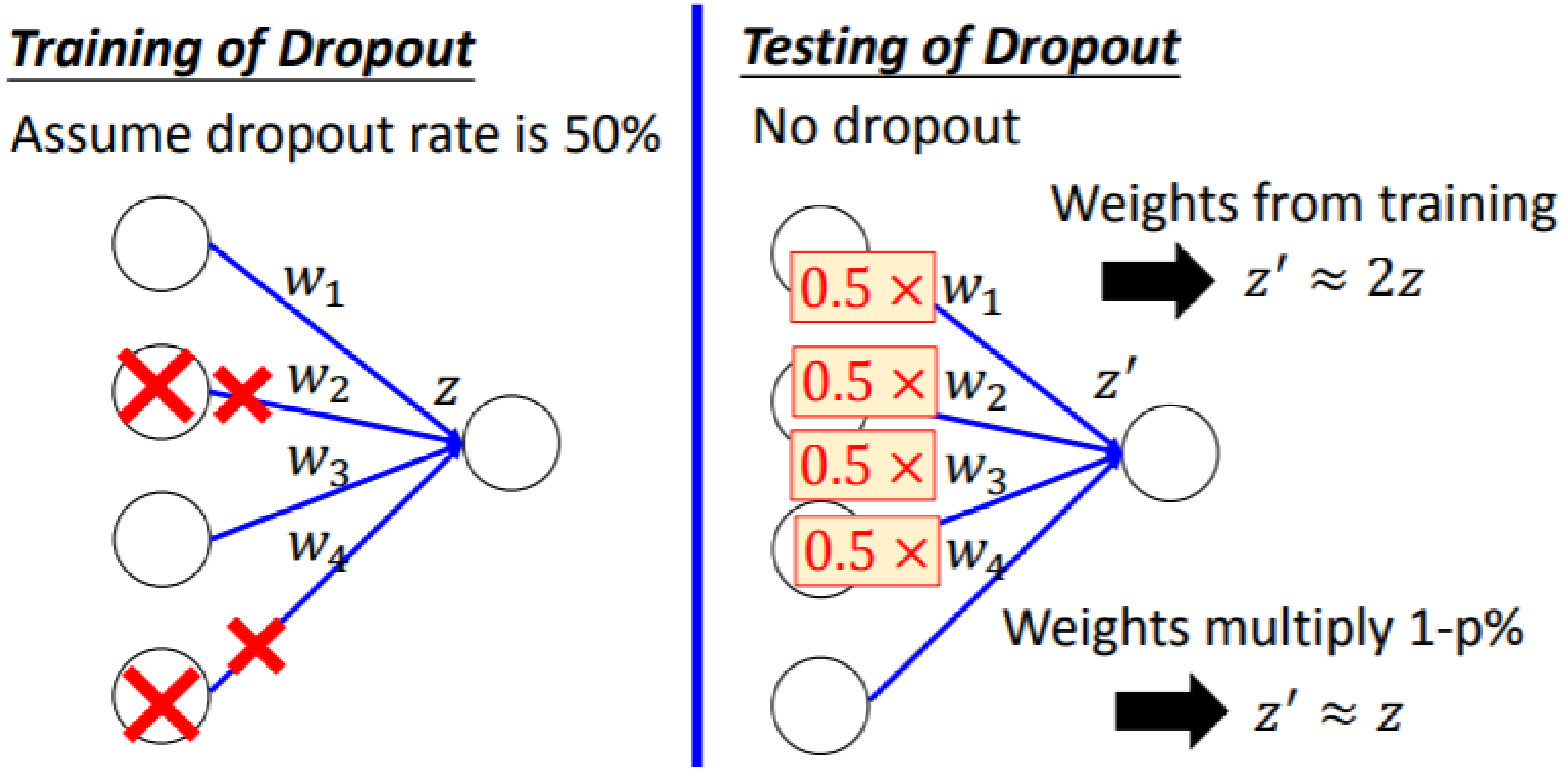


- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Why multiply weights by (1-p)% during testing?

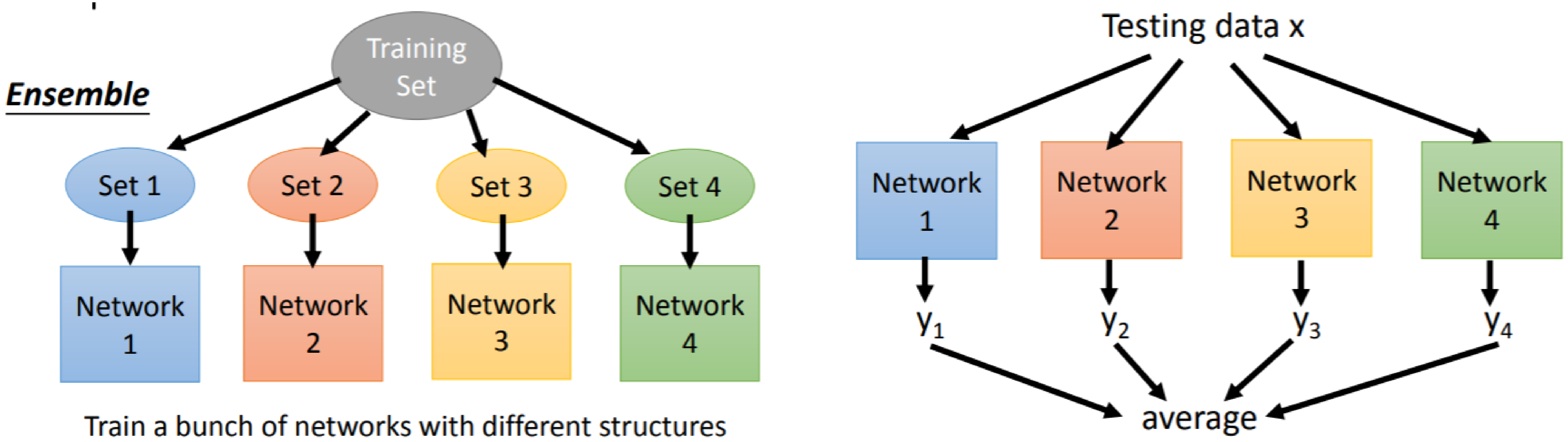
- Why the weights should multiply (1-p)% (dropout rate) when testing?



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

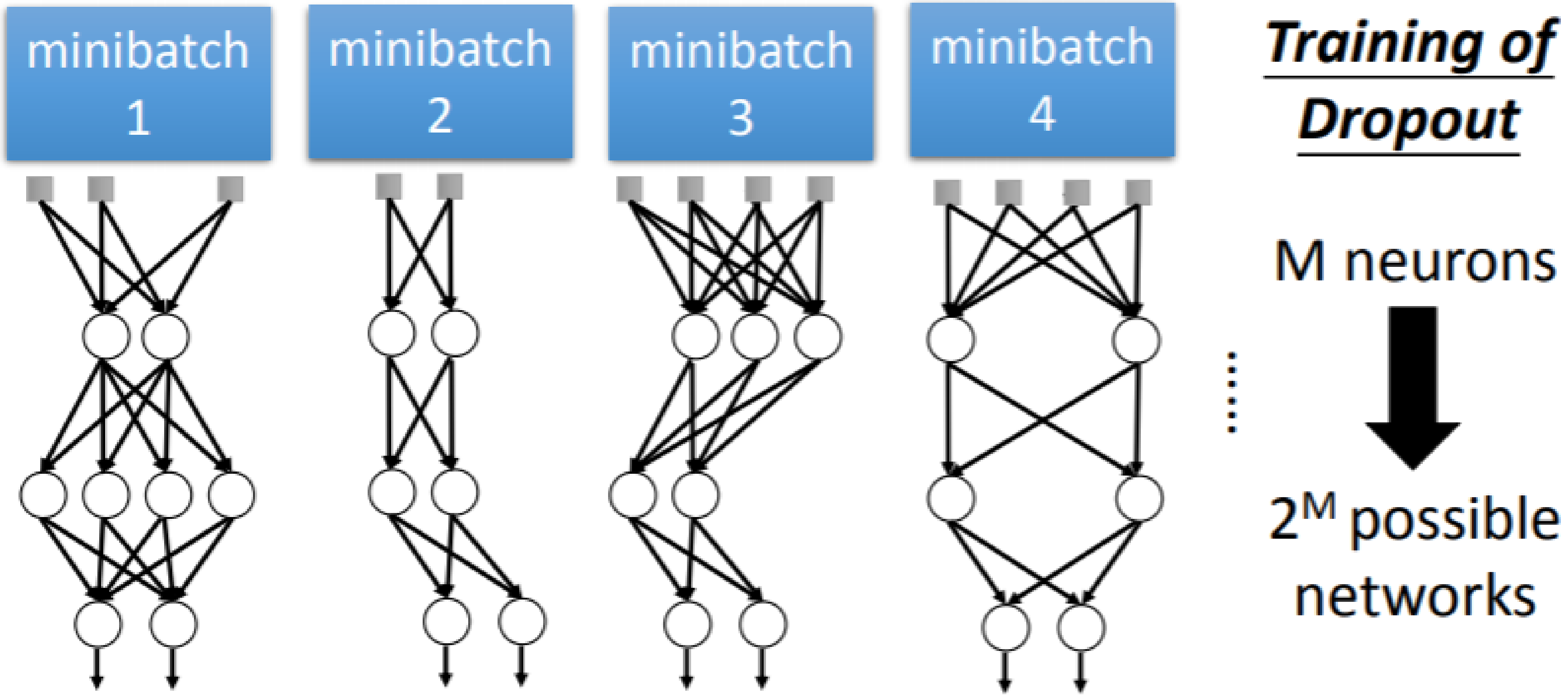
Why drop out makes NN performs better?

Drop out can be seen as an ensemble method.



Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

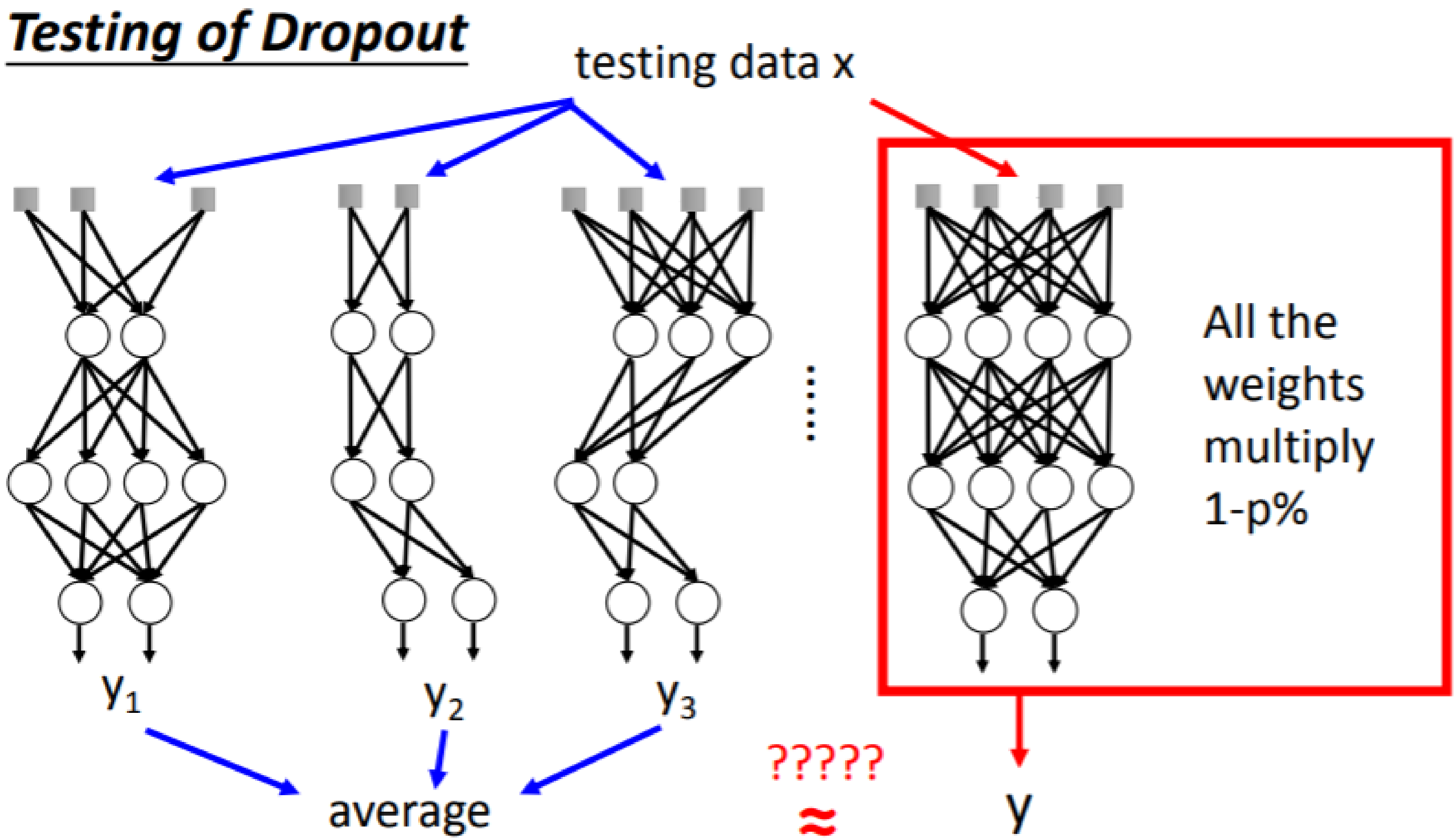
Why drop out makes NN performs better?



- ➢Using one mini-batch to train one network
- ➢Some parameters in the network are shared

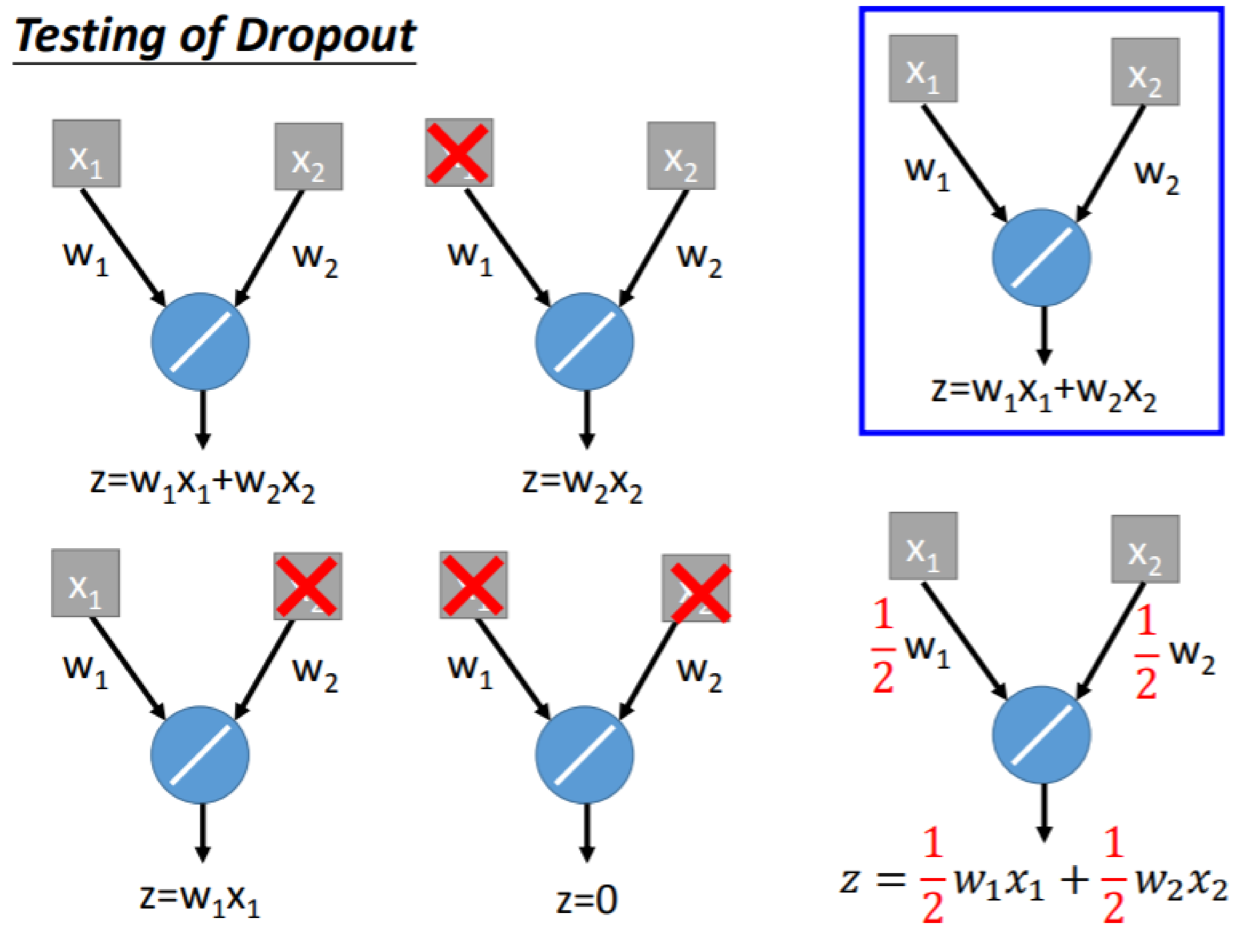
Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Why multiply weights by (1-p)% during testing?



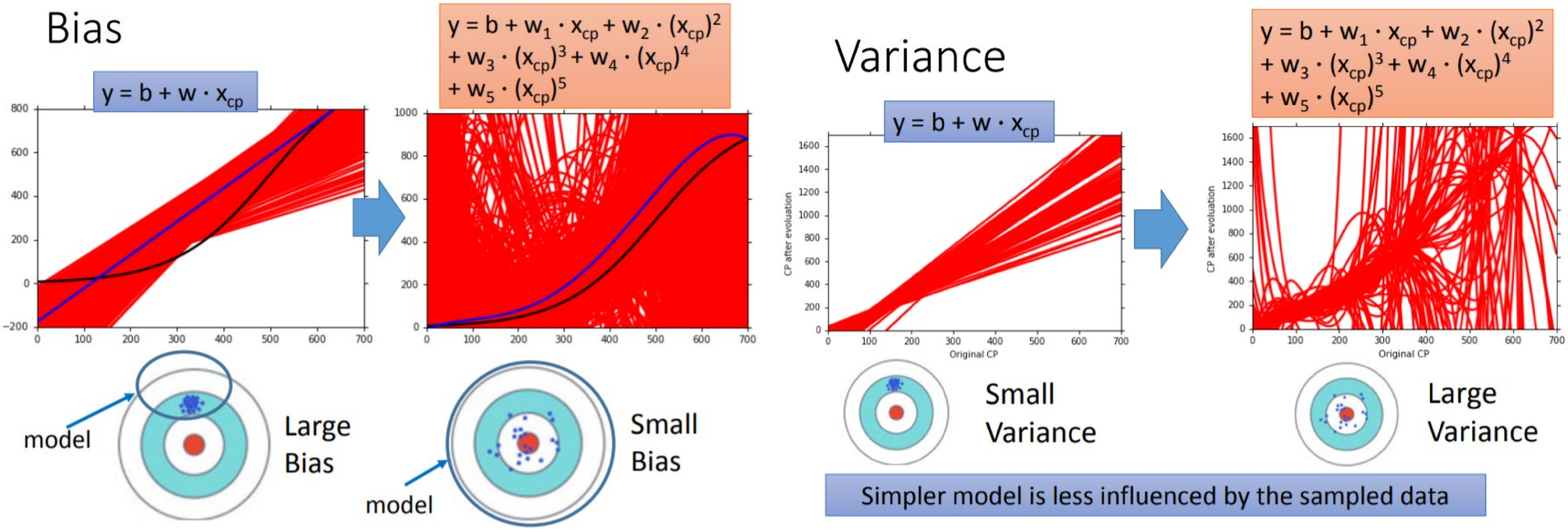
Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Why multiply weights by (1-p)% during testing?



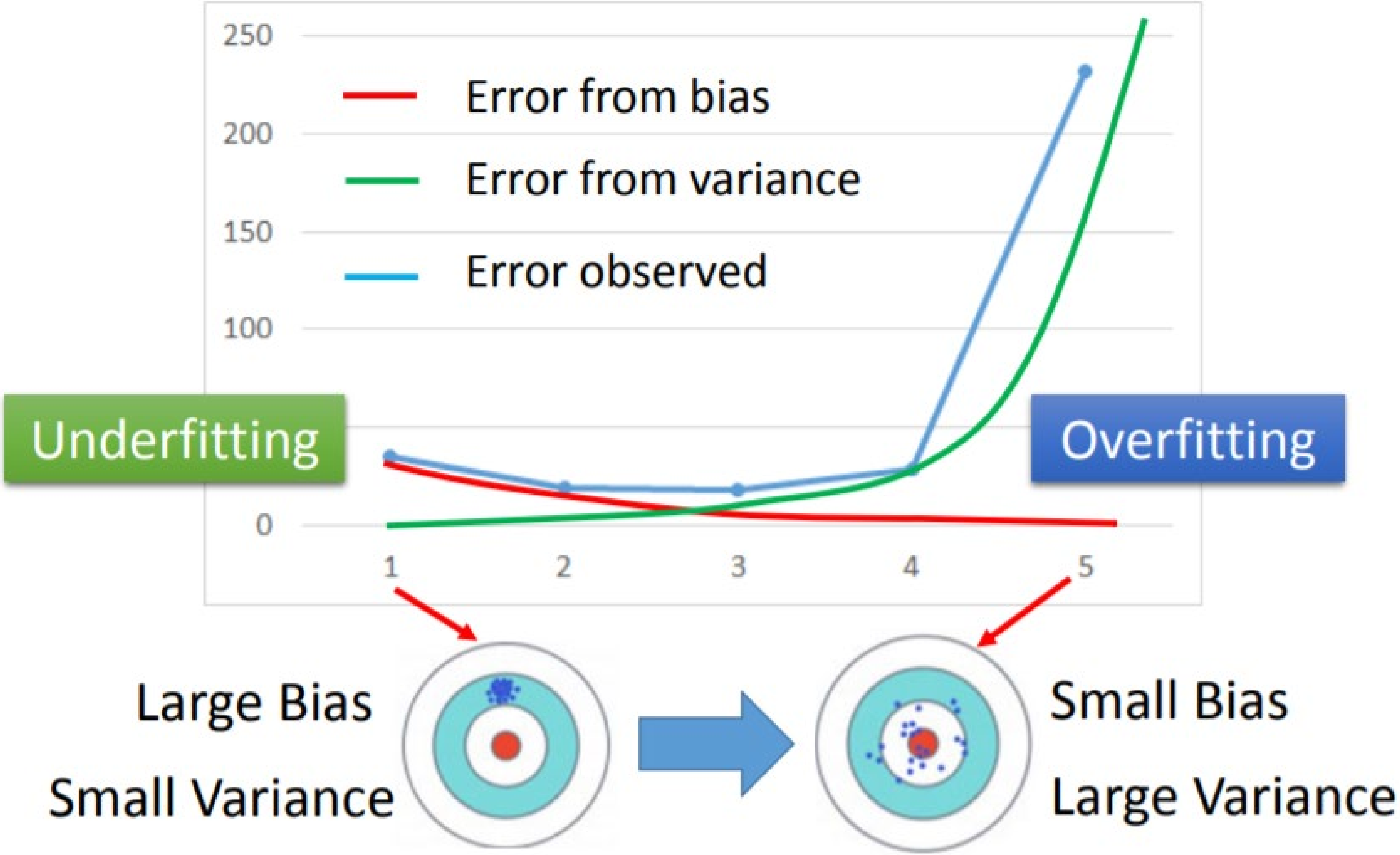
Reference: 李弘毅 ML Lecture 9-1 https://youtu.be/xki61j7z-30

Simpler model is less influenced by the sampled data and has smaller variance



Reference: 李弘毅 ML Lecture 2 https://youtu.be/D_S6y0Jm6dQ

Errors of ML model

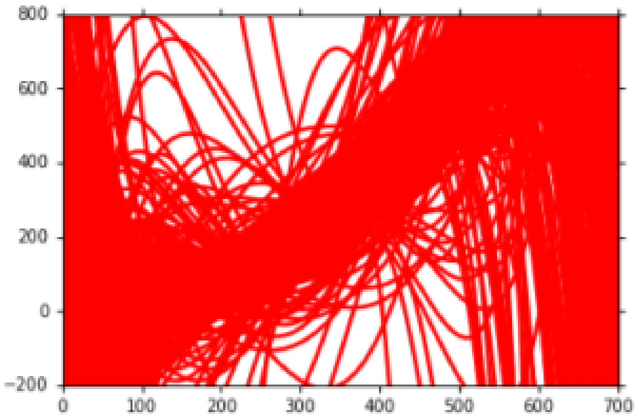


Reference: 李弘毅 ML Lecture 2 https://youtu.be/D_S6y0Jm6dQ

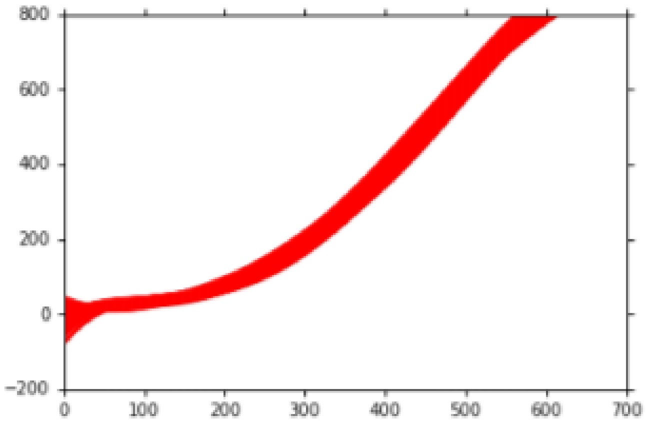
How to reduce model's variances?

① More data

Very effective, but not always practical

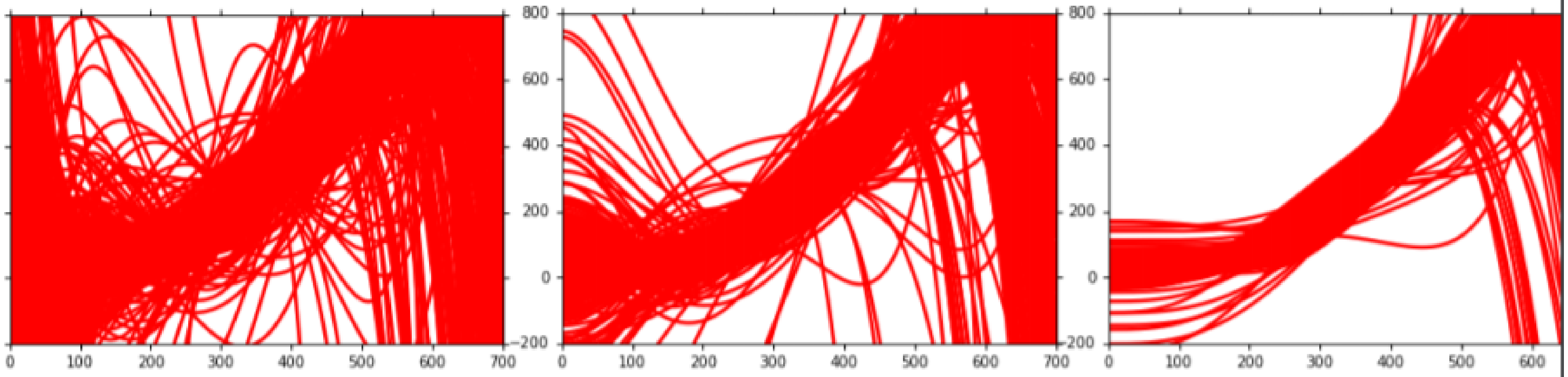


10 examples



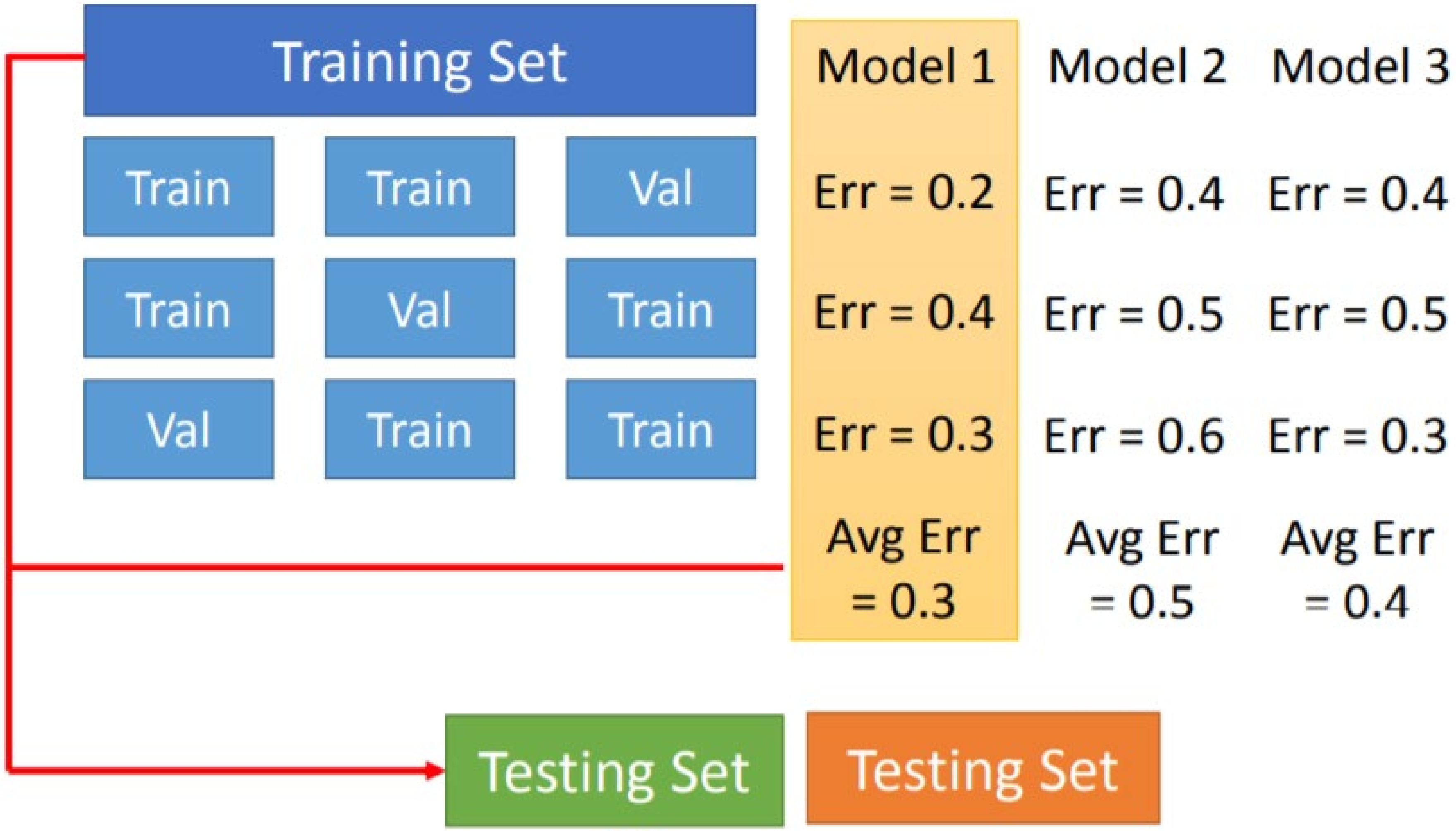
100 examples

② Regularization



Reference: 李弘毅 ML Lecture 2 https://youtu.be/D_S6y0Jm6dQ

Training with cross validation



Reference: 李弘毅 ML Lecture 2 https://youtu.be/D_S6y0Jm6dQ