ICCS482 Deep Learning

Lecture 5: underfitting, overfitting and regularization

Sunsern Cheamanunkul, Sep 22, 2020.

Diagnosing problem with NNs

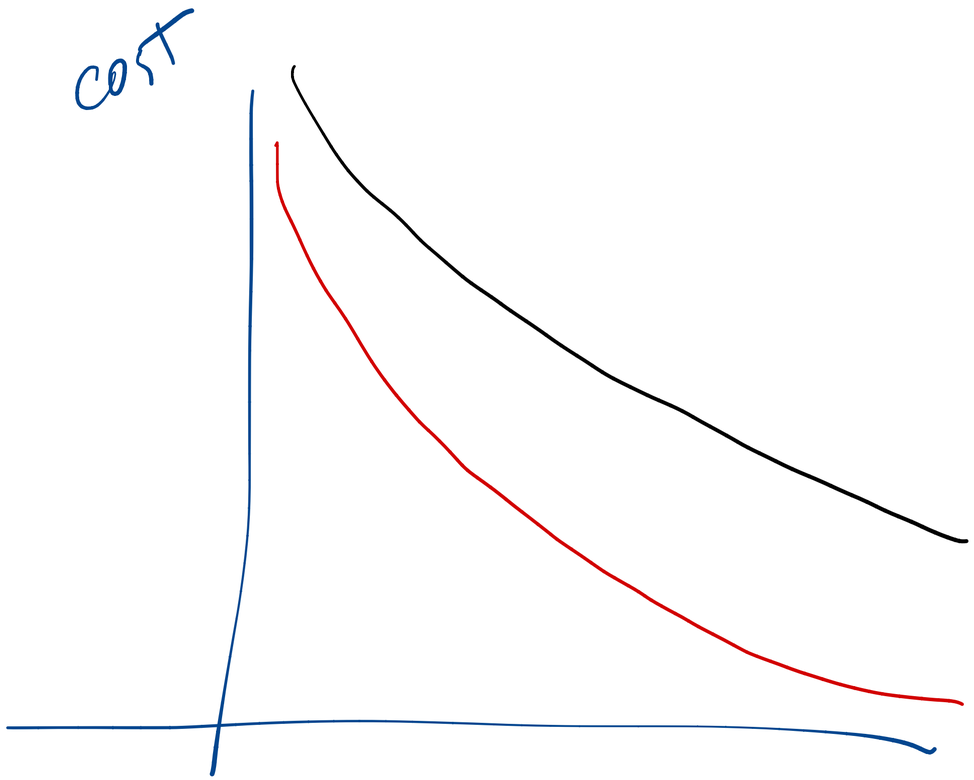
Terms

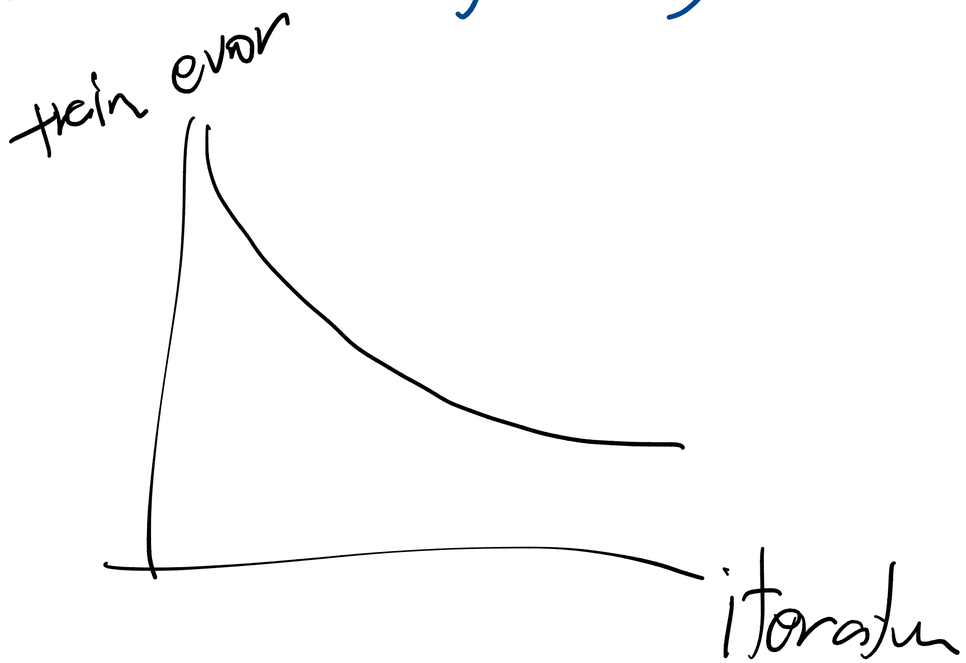
① Training errors — Cost on Training Set

② Test errors — Cost on the test set

Scenario #1

High train error (high cost on train set)

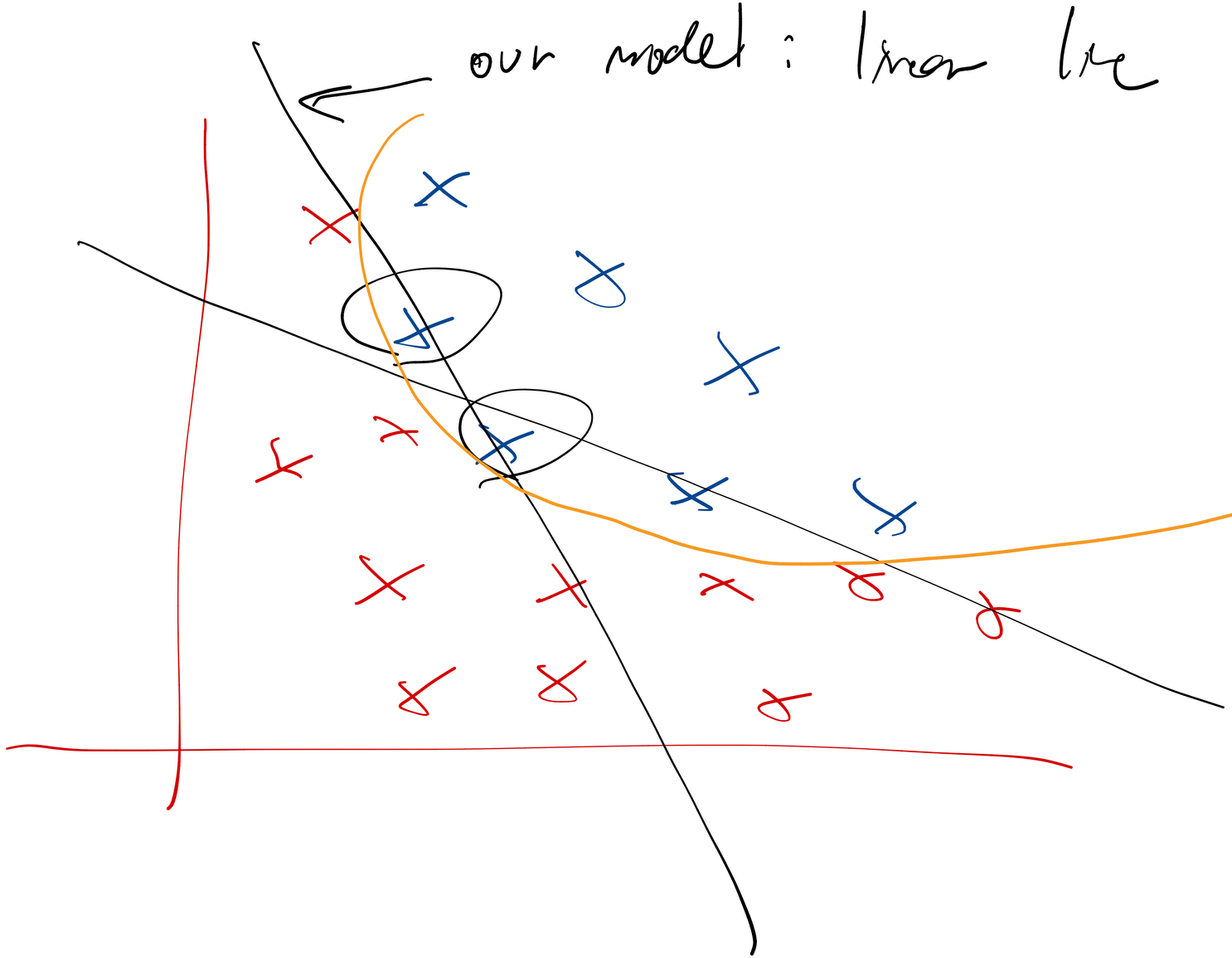




"underfitting"

Why do we have high train error?

1. model has not ~~enough~~ learned enough
2. intrically very hard
   (impossible to find a good model)
3. model is not complex enough
   (too simple)


our model: linear line
better model!
more complex than a straight line.

How to make a model more complex?

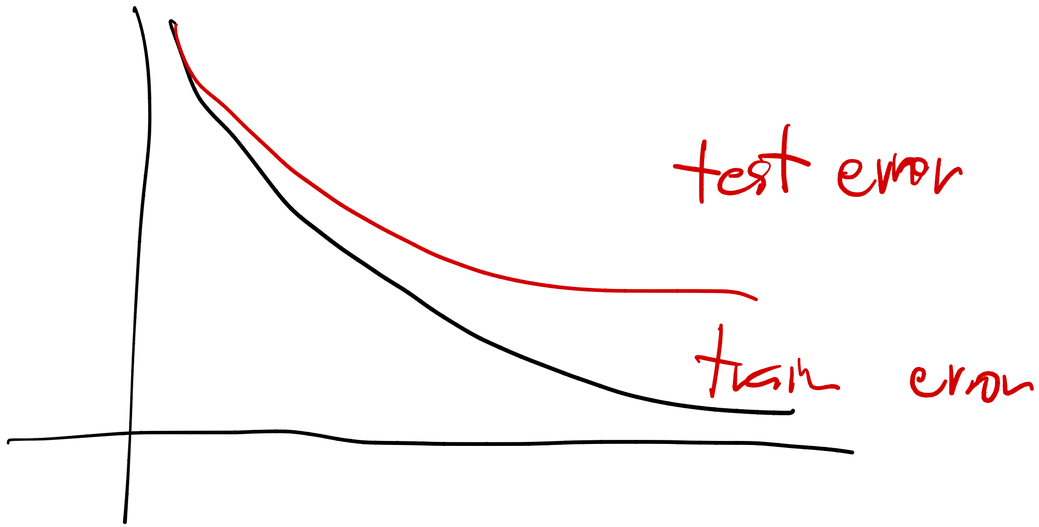
Deep NN → { add more layers
add more nodes in each layer

↓

more parameters!

Scenario #2

Low train error
high test error



"overfitting"

Reasons

① model too complex

② train data and test data doesn't come from the same distribution ~ haven't seen enough.

Scenario #3

high train error
low test error

] maybe something wrong with our train data

this is good!

(should not happen IRL)

scenario #4

high train error

high test error

model is not learning at all

- underfitting?

Underfitting

1. try adding more layers
2. try adding more hidden units
3. Add more parameters

Overfitting — fit train data well
— doesn't work on test

the model generalises poorly!

what to do

① decrease # of params
— # of hidden layers / units

② Add more data — so the model sees more data
need variety!

③ Regularization

– punishing the model in some way

– L_2 regularization:

$$J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{n} \ell(y^{(i)}, a^{(i)}) + \frac{\lambda}{2m} \|w\|^2$$

↑ encourage small weights

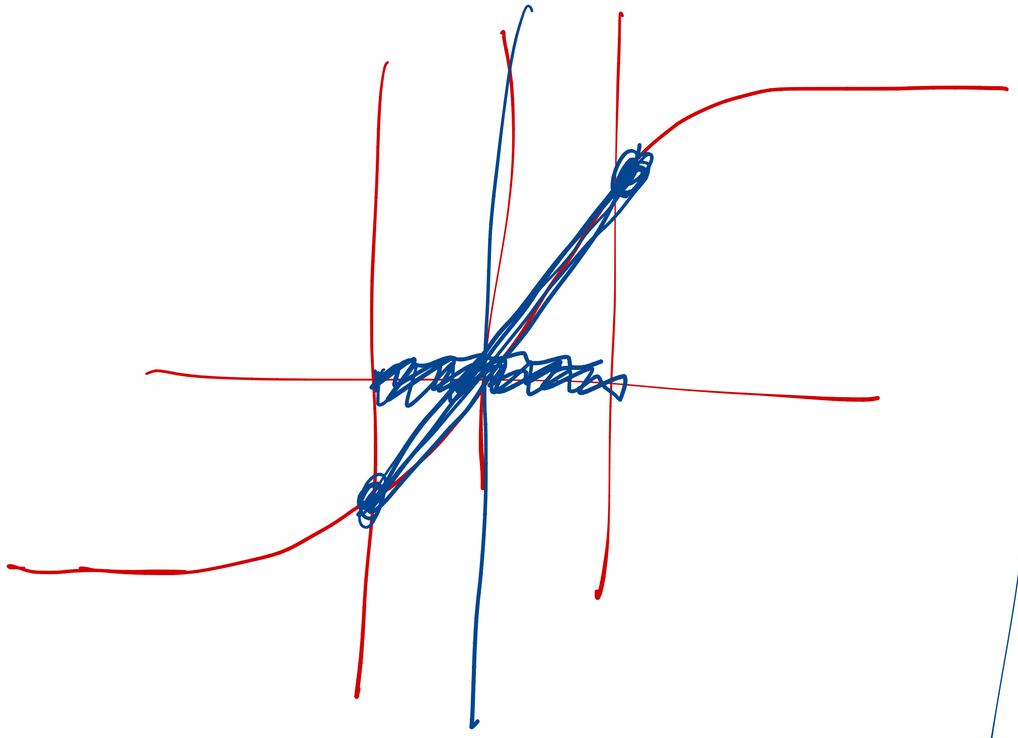
Find $\vec{w}, b$ that minimizes $J(\vec{w}, b)$

$$\vec{w}_1 = \begin{bmatrix} 100 \\ 200 \end{bmatrix} \quad \text{vs} \quad \vec{w}_2 = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$$

this is preferred by L_2 regularizer

Why regularization help with overfitting ?

① make the model less complex



as if we have less

$$z^{[l]} = w^{[l]} \cdot a^{[l-1]}$$

$$a^{[l]} = g(z^{[l]})$$

nodes

other regularizations

L_1 regularization

$$J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{n} \ell(y^{(i)}, y^{(i)}) + \frac{\lambda}{2m} \|w\|_1$$

$$\left\| \begin{bmatrix} 3 \\ -3 \end{bmatrix} \right\|_1 = |3| + |-3| = 6$$

$$\left\| \begin{bmatrix} 3 \\ 1 \end{bmatrix} \right\|_2 = 3^2 + 1^2 = 10$$

we want L_1 of $\vec{w}$ to be small

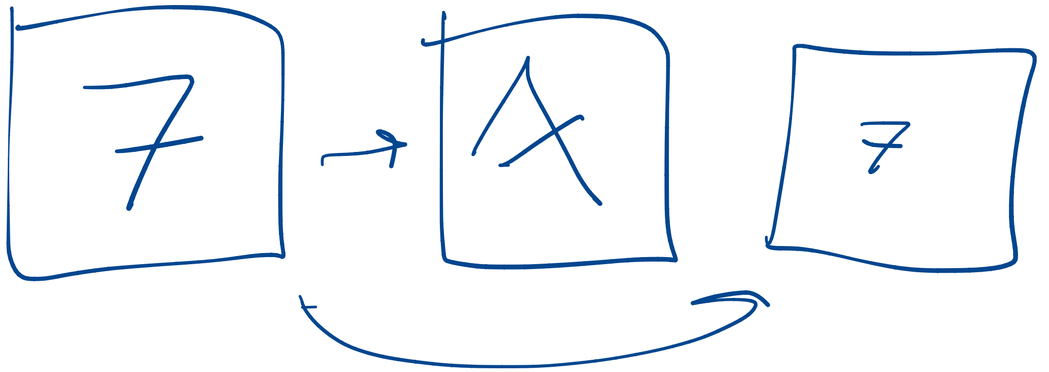
$\longrightarrow$ we want $\vec{w}$ to appear often in $\vec{w}$

Dropout regularization

- randomly zero out nodes
- do not want to trust any single nodes too much
- hope this helps with generalization
- don't drop during test phase

Other methods to fight overfitting

- Add more data!
  - sometimes this is not possible.
  - data augmentation techniques
    - take existing examples and tweak them a little e.g.



- Early Stopping