Debugging a learning Algo

If we use a certain formula and get unacceptably large errors in prediction we can do these steps:

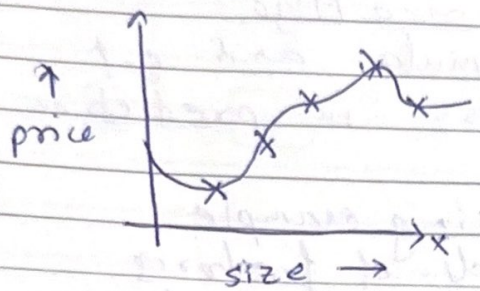
- Get more training example
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features
- Try increasing λ
- Try decreasing λ

But in reality, when applying these things on a DL model, some may turn out to be fruitful while others do not.

To help to choose / improve a DL, we run a diagnostic

Diagnostics take a bit of time to implement, but reduce the risk of spending time with ~~no~~ of thing which do not help

Evaluating your Model



Model fits training data well but will fail to generalize to new examples not in training set. (overfitting)

$$f_{\vec{w}b}(\vec{x}) = w_1 x + w_2 x^2 + \cdots + w_n x^n + b \qquad x = x_1$$

here we are using only 1 feature and can see its not good.

x_1 = size in ft^2
x_2 = no. of bedrooms
x_3 = no. of floors
x_4 = age of home in year.

We can evaluate our model by dividing our data set

features	result	
	} 70% training set	(x^1, y^1) (x^2, y^2) ⋮ $(x^{m_{train}}, y^{m_{train}})$
	} 30% test set	(x^1_{test}, y^1_{test}) ⋮ $(x^{m_{test}}_{test}, y^{m_{test}}_{test})$

Ex. Train/test procedure for linear Regression.

Fit Parameters

$$J(\vec{w}, b) = \left[\frac{1}{2m_{train}} \sum_{i=1}^{m_{train}} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m_{train}} \sum_{j=1}^{n} w_j^2 \right]$$

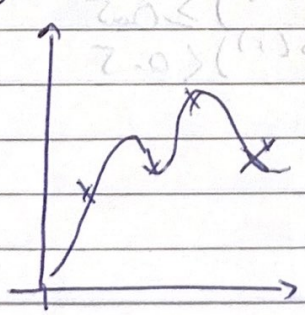
Compute test error:

$$J_{test}(\vec{w}, b) = \frac{1}{2m_{test}} \left[\sum_{i=1}^{m_{test}} f_{\vec{w},b}\left(\vec{x}^{(i)}_{test}\right) - y^{(i)}_{test} \right)^2 \right]$$

Compute training error

$$J_{train}(\vec{w}, b) = \frac{1}{2m_{train}} \left[\sum_{i=1}^{m_{train}} \left(f_{\vec{w},b}\left(\vec{x}^{(i)}_{train}\right) - y^{(i)}_{train} \right)^2 \right]$$

Both ~~test~~ & training error does not contain the regularization term.



$J_{train}(\vec{w}, b)$ will be low

$J_{test}(\vec{w}, b)$ will be high

Train/test procedure for classification problem

Fit parameters

$$J(\vec{w},b) = -\frac{1}{m_{train}} \sum_{i=1}^{m_{train}} \left[y^{(i)} \log\left(f_{\vec{w},b}(\vec{x}^{(i)})\right) + (1-y^{(i)}) \log\left(1 - f_{\vec{w},b}(\vec{x}^{(i)})\right) \right] + \frac{\lambda}{2m_{train}} \sum_{j=1}^{m} w_j^2$$

Compute test/train error is same as fit parameters without the regularization term.

There is another that is commonly used for classification problems.

↳ fraction of test set & fraction of training set the algorithm has misclassified

$$\hat{y} = \begin{cases} 1 \text{ if } f_{\vec{w},b}(\vec{x}^{(i)}) \geq 0.5 \\ 0 \text{ if } f_{\vec{w},b}(\vec{x}^{(i)}) < 0.5 \end{cases}$$

count $\hat{y} \neq y$

Model Selection

Once parameters $\vec{w}, b$ are fit to the training set, the training error is most likely lower than the actual generalization error

$J_{test}(\vec{w}, b)$ is a better estimate of how well the model will generalize to new data compared to $J_{train}(\vec{w}, b)$

order

d = 1 1. $f_{\vec{w},b} = w_1 x + b \rightarrow w^{<1>}, b^{<1>} \rightarrow J_{test}(w^{<1>}, b^{<1>})$

d = 2 2. $f_{\vec{w},b} = w_1 x + w_2 x^2 + b \rightarrow w^{<2>}, b^{<2>} \rightarrow J_{test}(w^{<2>}, b^{<2>})$

d = 3 3. $f_{\vec{w},b} = w_1 x + w_2 x^2 + w_3 x^3 + b$

$\vdots$

d = 10 10. $f_{\vec{w},b} = w_1 x + w_2 x^2 + w_3 x^3 + \cdots + w_{10} x^{10} + b$

We find the error of test set for each polynomial and take the lowest error model.

This method is flawed as J_{test} < generalization error and J_{test} is an optimistic estimate of generalization error

Model Selection - ML

size	price		
	training set 60%	→	$(x^{(1)}, y^{(1)})$ ⋮ $(x^{(m_{train})}, y^{(m_{train})})$
	cross validation 20%	→	$(x_{cv}^{(1)}, y_{cv}^{(1)})$ ⋮ $(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$
	test set 20%	→	$x_{test}^{(1)}, y_{test}^{(1)}$ ⋮ $(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Training error:

$$J_{train}(\vec{w}, b) = \frac{1}{2m_{train}} \left[\sum_{i=1}^{m_{train}} \left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)} \right)^2 \right]$$

Cross validation error:

$$J_{cv}(\vec{w}, b) = \frac{1}{2m_{cv}} \left[\sum_{i=1}^{m_{cv}} \left(f_{\vec{w},b}(\vec{x}_{cv}^{(i)}) - y_{cv}^{(i)} \right)^2 \right]$$

Test error:

$$J_{test}(\vec{w}, b) = \frac{1}{2m_{test}} \left[\sum_{i=1}^{m_{test}} \left(f_{\vec{w},b}(\vec{x}_{test}^{(i)}) - y_{test}^{(i)} \right)^2 \right]$$

d = 1 1. $f_{\vec{w},b}(\vec{x}) = w_1 x + b \rightarrow w^{<1>}, b^{<1>} \rightarrow J_{cv}(w^{<1>}, b^{<1>})$

d = 2 2. $f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + b \rightarrow w^{<2>}, b^{<2>} \rightarrow J_{cv}(w^{<2>}, b^{<2>})$

d = 3 3. $f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + w_3 x^3 + b$

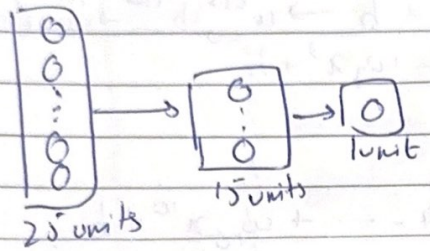
⋮

d = 10 10. $f_{\vec{w},b}(\vec{x}) = w_1 x + w_2 x^2 + \cdots + w_{10} x^{10} + b$

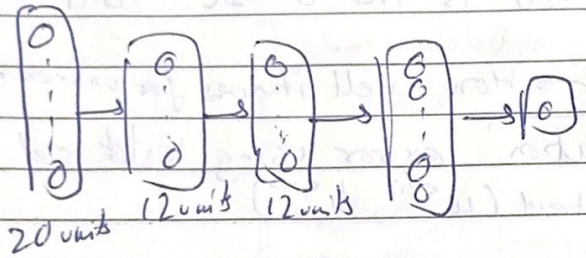
We then see which J_{cv} is the lowest and pick that model

Estimate the generalization (→ How well it does for unknown examples) error using test set $J_{test}(w^{<4>}, b^{<4>})$

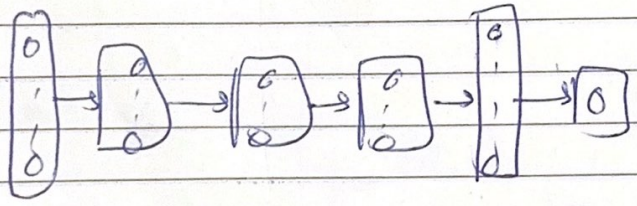
Model Selection - NN

1. 

$w^{[1]}, b^{[1]}$ J_{cv}

2. 

$w^{[2]}, b^{[2]}$ J_{cv}

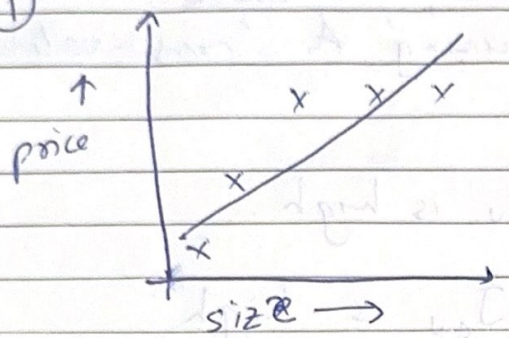
3. 

$w^{[3]}, b^{[3]}$ J_{cv}.

Pick model with lowest J_{cv}.

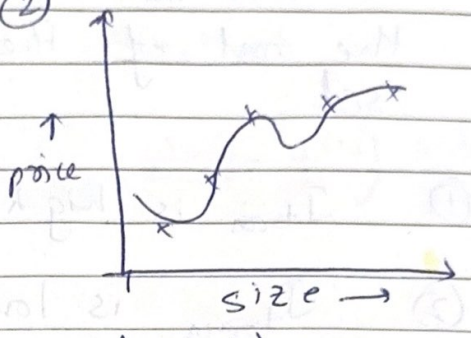
Estimate generalization error using $J_{test}(w, b)$

Bias & Variance

① 

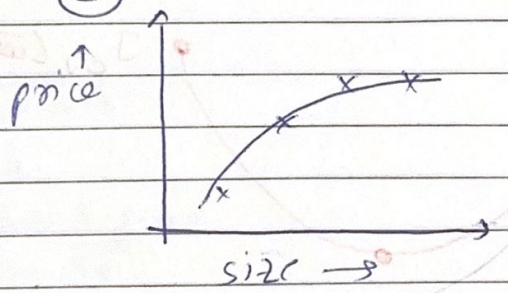
$f(\vec{w}, b) = w_1 x + b$

high bias (underfit)

② 

$f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

high variance (overfit)

③ 

$f_{(\vec{w},b)}(x) = w_1 x + w_2 x^2 + b$

(just right)

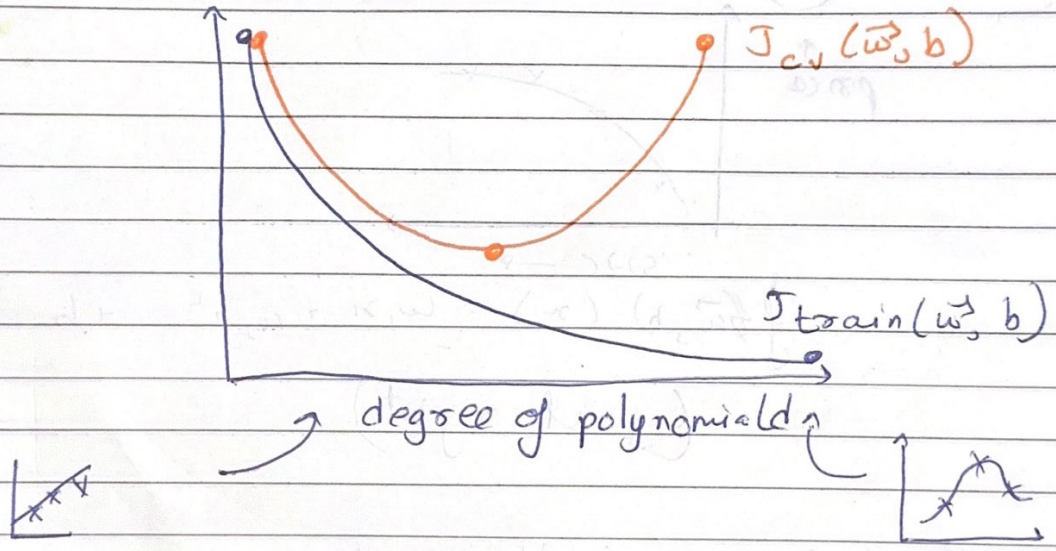
This f is for 1 variable, so it is easy to plot. we need another method for multiple features

To check the performance, we check the cost of the 'training' & 'cross-validation' set.

d=1 ① J_{train} is high, J_{cv} is high.

d=4 ② J_{train} is low, J_{cv} is high

d=2 ③ J_{train} is low, J_{cv} is low



∴ High bias (underfit) → J_{train} will be high
($J_{train} \approx J_{cv}$)

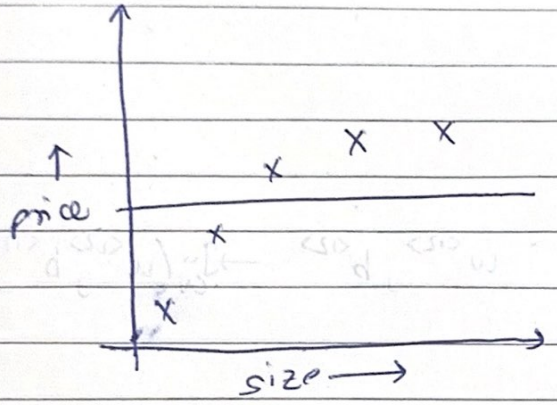
High variance (overfit) → J_{train} may be low
$J_{cv} >> J_{train}$

High bias & variance → J_{train} is high & $J_{cv} >> J_{train}$

Regularization & bias/variance

Model: $f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

$$J(\vec{w},b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$



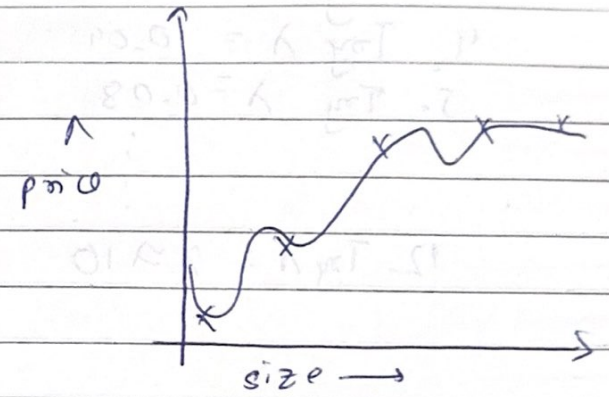
Large λ

larger the value λ, more motivated will be the regularization term to keep $\vec{w}$ small

High bias (underfit)

$J_{train}(\vec{w},b)$ is large

$J_{CV}(\vec{w},b)$ is high



Small λ

small λ, less general will be the final graph, or less motivation to make it generalizes

High variance (overfit)

$J_{train}(\vec{w},b)$ is small

$J_{CV}(\vec{w},b)$ is high

An intermediate λ value will give a proper graph

To choose the proper λ value, we take various values for λ and the equation with the least J_{cv} will give the proper λ value.

Ex. 1. Try $\lambda = 0 \rightarrow \min_{\vec{w}, b} J(\vec{w}, b) \rightarrow w^{<1>}, b^{<1>} \rightarrow J_{cv}(w^{<1>}, b^{<1>})$

2. Try $\lambda = 0.01 \rightarrow w^{<2>}, b^{<2>} \rightarrow J_{cv}(w^{<2>}, b^{<2>})$

3. Try $\lambda = 0.02$

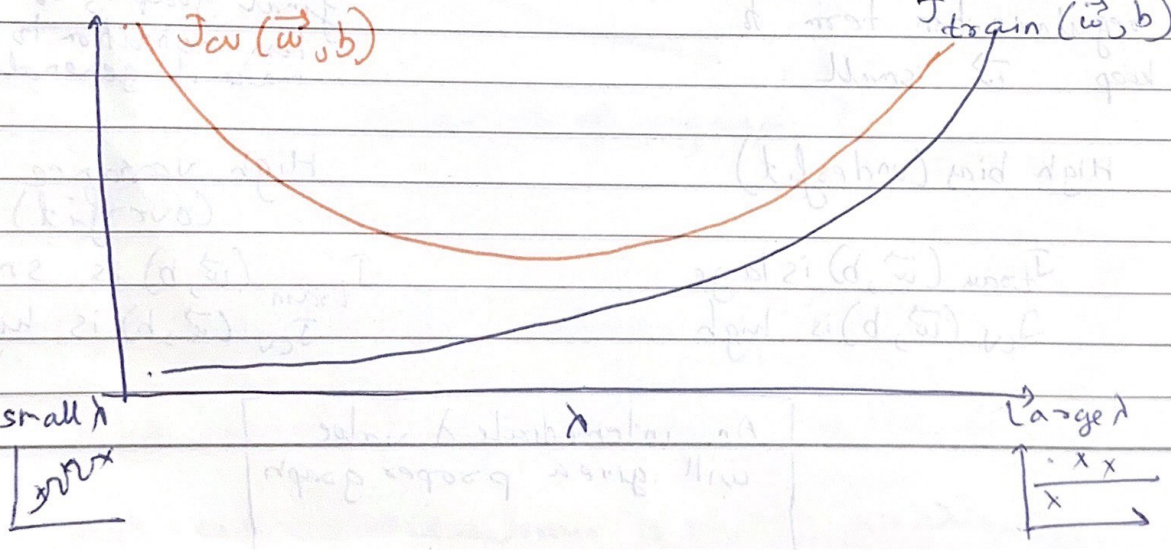
4. Try $\lambda = 0.04$

5. Try $\lambda = 0.08$

⋮

12. Try $\lambda = \lambda \times 10 \quad w^{<12>}, b^{<12>} \rightarrow J_{cv}(w^{<12>}, b^{<12>})$

We take the best λ value and calculate generalization error using J_{test}



Summary:→ If baseline loss is equal/near to training cost & high loss diff btw training & CV then overfitting

If loss difference btw baseline & test is high, then overfitting underfitting

Establishing baseline level of performance

Whenever evaluating an algorithm, it is always good to measure it against another performance like:

- Human level performance
- Competing algorithms performance
- Guess based on experienced

Ex.

① Human error = 10.6 %
② Training error = 10.8%
③ Cross validation error = 14.8%

If we only see ② & ③, we see as underfitting, but checking ①, we see that it actually is an overfitting error.

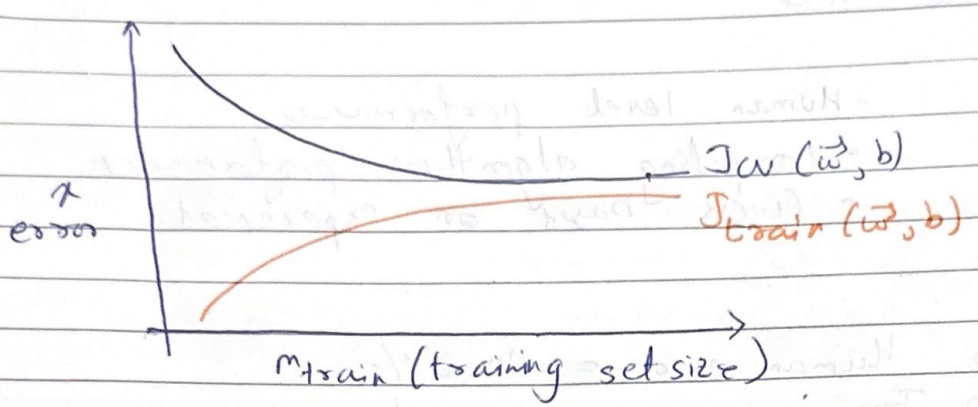
Ex Human error ⟵ 10.6% ↓0.2 He 10.6 ↑4.4
Training error (J_{train}) → 10.8% ↓4.0 Te 15.0 ↓
Cross Validation Error (J_{cv}) → 14.8% ↑ Cve 15.5 0.5↓
high variance high bias

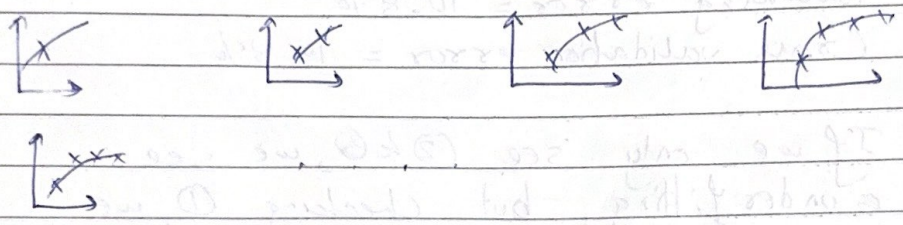
He → 10.6 ↑4.4
Te → 15.0 ↓ high variance
Cve → 19.7 ↑4.7 high bias.

Learning Curves

Lets take $f_{\vec{w},b}(x) = w_1 x + w_2 x^2 + b$.

J_{train} = training error
J_{cv} = cross validation error.

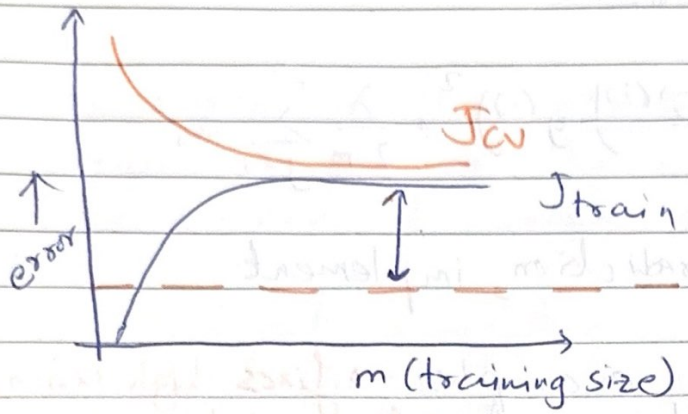




. . . .

If we have small no. of ~~too~~ training examples, it is easy to keep the error low. However, if we increase the number of examples, it become more tricky to keep error low

High Bias (underfitting)



Note. After a point, adding more data will not make J_{cv} & J_{train} come closer as the model cannot go to fit those examples

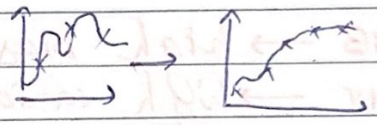
↓

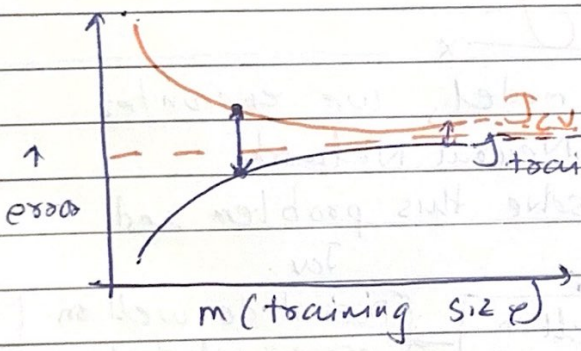
model becomes too simple for the task at hand.

~~Note:~~

if a learning algorithm suffers from high bias, getting more training data will not (by itself) help much.

High Variance (overfitting)





model produces un-realistically low error

if a learning algorithm suffers from high variance, getting more data is likely to help.

Debugging a Learning Algorithm.

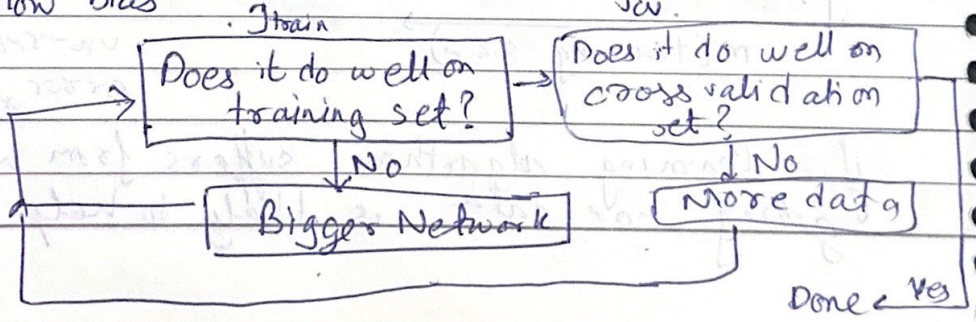
$$J(\vec{w},b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

If It makes errors in prediction, implement

HV ◦ Get more training examples → fixes high variance
HV ◦ Try smaller feature set → fixes high variance
HB ◦ Try getting more features → fixes high bias
HB ◦ Try adding polynomial features (x_1^2, x_2^2, x_1, x_2, etc) → fixes high bias
HB ◦ Try decreasing λ → fixes high bias
HV ◦ Try increasing λ → fixes high variance

HB → high bias = underfitting
HV → high variance = overfitting

When trying to find a proper model, we encounter the bias variance tradeoff. Neural Network that are big enough can solve this problem and achieve a low bias



Draw back is that bigger the network, more strain is put on the hardware
The number of unique data is also limited.

To prevent overfitting in NN, we use regularization.
In tensor flow, we have to add 'kernal_regularizer' to the layer declaration.

```
layer1 = Dense(units=25, activation='selu',
               kernal_regularizer=L2(0.01))
```