Sometimes a learning model can have a lot of errors.

for example a regularized linear regression:

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

makes very large errors. Why?

→ maybe less training data
→ maybe λ is not correctly chosen, try incr. or decr.
→ maybe try using smaller sets of features
→ maybe try polynomial features (x_1^2, x_2^2, $x_1 x_2$, etc.)

Some people misjudge the reason behind the errors which wastes a lot of time.

Diagnostics :-

It is a test used to check the real reason due to which the errors are being caused.

It can take time to run these tests, but are very helpful.

Things to keep in mind when deciding a model is :-

→ Is it overfitting ?

→ Does it generalize to other examples ?

→ Can it plot data if more features are added?

The technique that is most used for testing against these types of problems is splitting the dataset .

$\boxed{\quad 70\% \quad}$ $\boxed{30\%}$

70% of the dataset is used for training

30% is used for testing

$x^{(1)}, y^{(1)}$

$x^{(2)}, y^{(2)}$

$\vdots$

$x^{(mtrain)}, y^{(mtrain)}$

$x^{(1)}_{test}, y^{(1)}_{test}$

$x^{(2)}_{test}, y^{(2)}_{test}$

$\vdots$

$x^{(mtest)}_{test}, y^{(mtest)}_{test}$

mtrain = no. of training examples

mtest = no. of testing examples

First train the data on a regularized cost function

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

Then check the data on the test set.

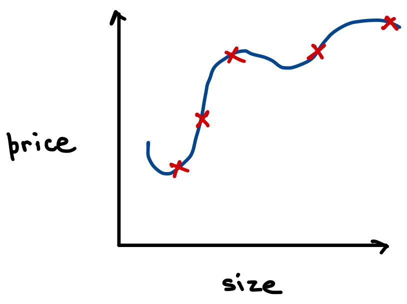
$$J(\vec{w}, b) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(f_{\vec{w},b}(x_{test}^{(i)}) - y^{(i)}\right)^2$$

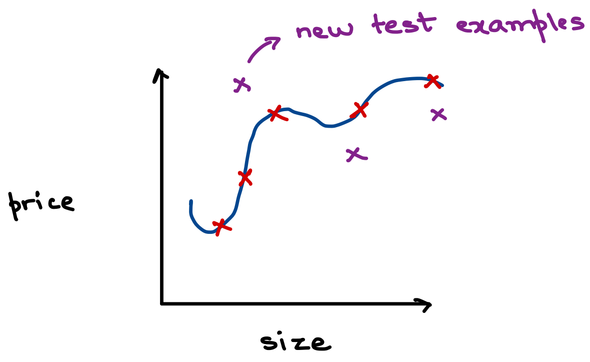
no w_j required because model is not being trained during testing

Also, check the cost function on training set

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

We'll notice that the training set will have much lower cost because the model has already trained itself according to the training set, but the test set might have a high cost due to it not being trained.





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

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

If J_{train} is low and J_{test} is high, that means the model is not adequate and doesn't generalize for untrained data

You can also check for classification problems

Minimize $J(\vec{w}, b)$:

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

Compute test error:

$$J(\vec{w}, b) = \frac{-1}{m_{test}} \sum_{i=1}^{m_{test}} \left[y^{(i)} \log (f_{\vec{w},b}(x_{test}^{(i)}) + (1 - y^{(i)}) \log (1 - f_{\vec{w},b}(x_{test}^{(i)})) \right]$$

Compute train error:

$$J(\vec{w},b) = \frac{-1}{m_{train}} \sum_{i=1}^{m_{train}} [y^{(i)} \log (f_{\vec{w},b}(x^{(i)})) + (1-y^{(i)}) \log (1-f_{\vec{w},b}(x^{(i)}))]$$

Again we check if cost for test is not high compared to cost for train.

There is also one more way to check how well the algorithm works in classification and that is by checking what fraction of test set and training set the algorithm has misclassified.

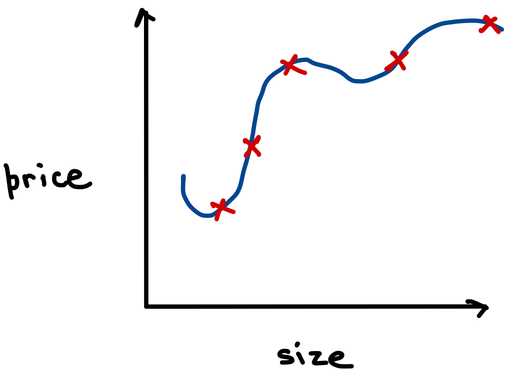
An algorithm was classified like this :-

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

Just count how many $\hat{y} \neq y$ (true label)

There will be a part of the test set and there will be a part of the training set that has been misclassified

We can conduct more tests to determine what kind of model should we use, straight line, quadratic, cubic ? etc.



Once parameters $\vec{w}$ and b are fit according to the training set, they are not a legible indicator of how well a data is going to work for general data.

$J_{train}(\vec{w}, b)$ is likely lower than the actual training error.

$J_{test}(\vec{w}, b)$ is a better indicator of how well the model will work in general because the test data has not been trained yet.

But, $J_{test}(\vec{w}, b)$ is also not the best way

Model Selection

degree of polynomial

We'll get different w, b when we train polynomials of different degree

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

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

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

$\downarrow$

$w^{<3>}, b^{<3>} \rightarrow J_{test}(w^{<3>}, b^{<3>})$

$\vdots$

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

$\downarrow$

$w^{<10>}, b^{<10>} \rightarrow J_{test}(w^{<10>}, b^{<10>})$

Now, we'll find for which order polynomial does J_{test} give the lowest error.

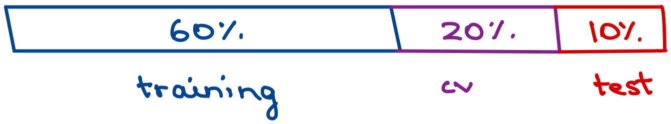
For example, if $J_{test}(w^{<5>}, b^{<5>})$ gives the lowest error then degree 5 order polynomial is the best choice.

The problem with this approach is that it is overly optimistic and thus inaccurate. In the above case we are purposely choosing which order polynomial will give us the best model according to the test set. It might work for the test set but not in general.

The J_{test} might be lower than actual error because we are choosing parameters according to the test set and not thinking about the general dataset.

To solve this problem, we take something called cross validation (cv) set.

We will break our set into three parts.



The training set will be always used for training. The cross-validation set is used for setting the parameters according to the order of the polynomial. Finally, the test set will always be used to check if the model will work well for the general data or not.

$$(x^{(1)}, y^{(1)}) \qquad x^{(1)}, y^{(1)} \qquad x^{(1)}, y^{(1)}$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$(x^{(m_{train})}, y^{(m_{train})}) \qquad (x^{(m_{cv})}, y^{(m_{cv})}) \qquad x^{(m_{test})}, y^{(m_{test})}$$

First we'll train the model according to the training set and validate on the cv set

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

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

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

$\vdots$

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

$\downarrow$

$J_{cv}(w^{<10>}, b^{<10>})$

Now, we pick for which order polynomial, the $\vec{w}$ and b give the least error.

For, example if $J(w^{<4>}, b^{<4>})$ gives the least error then $w^{<4>}$ and $b^{<4>}$ will be the parameters.

We will verify the validity of the model by testing the $w^{<4>}$ and $b^{<4>}$ (the chosen parameters) on the test set.

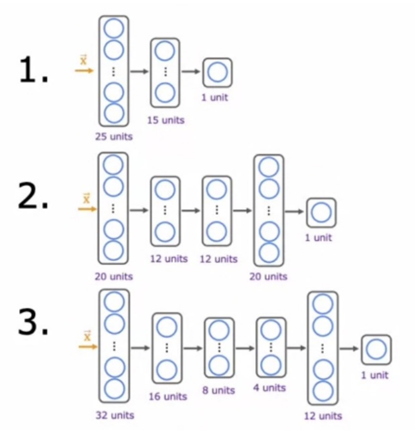
If $J_{test}(w^{<4>}, b^{<4>})$ is high then the model is not very accurate.

Choosing the correct neural network architecture

We can use the training set, cv and test set for choosing the no. of layers, the no. of neurons in a layer, etc. in a neural network architecture.

For eg. If we had three neural network models for the handwriting problem and we have to choose.

check w, b on cv set



$\rightarrow J_{cv}(w^{<1>}, b^{<1>})$

$\rightarrow J_{cv}(w^{<2>}, b^{<2>})$

$\rightarrow J_{cv}(w^{<3>}, b^{<3>})$

For eg. if we get $J_{cv}(w^{<2>}, b^{<2>})$ to be the lowest error, we can check how good the model works in general by running $J_{test}(w^{<2>}, b^{<2>})$