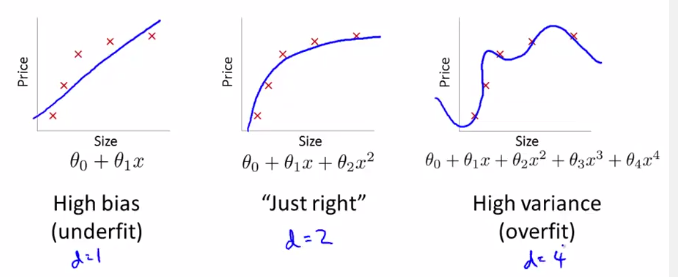
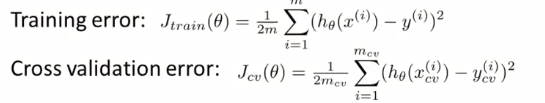
***Bias vs. Variance***

1. **DIAGNOSING BIAS VS. VARIANCE**

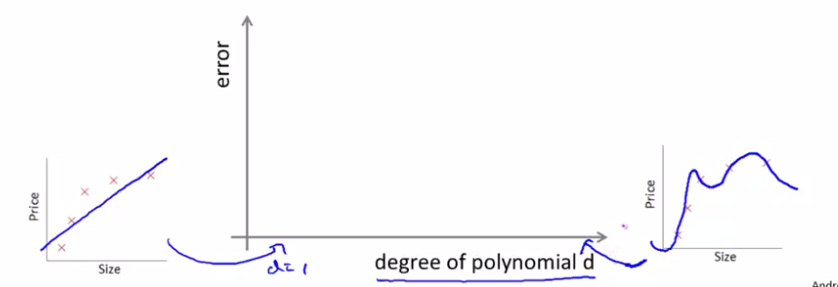
* If running a learning algorithm + it doesn't do as well as you were hoping, almost all the time it will be b/c you have either a **high bias problem** or a **high variance problem** (either an **underfitting** problem or an **overfitting** problem)
* In this case, it's very important to figure out which of these 2 problems it is, bias or variance, or a bit of both
* Knowing which of these 2 things is happening gives a very strong indicator for whether there are
* If you fit a too-simple hypothesis, it looks like a straight line through data that that underfits it.
* If you fit a too-complex hypothesis, it might fit a training set perfectly, but overfit new data
* We want a hypothesis of some intermediate level of complexity, maybe a 2 degree polynomial, not too low and not too high of a degree
* i.e. want a degree that gives you best generalization error.



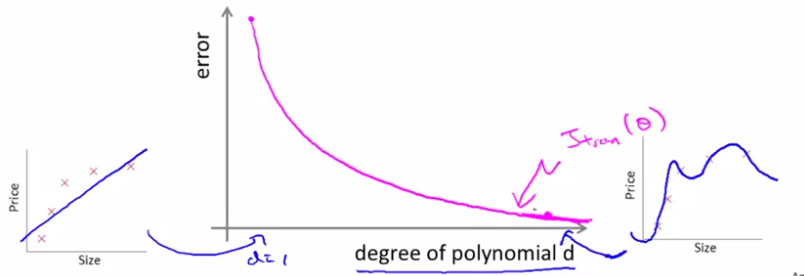
* Remember the **training error** and **cross validation error**



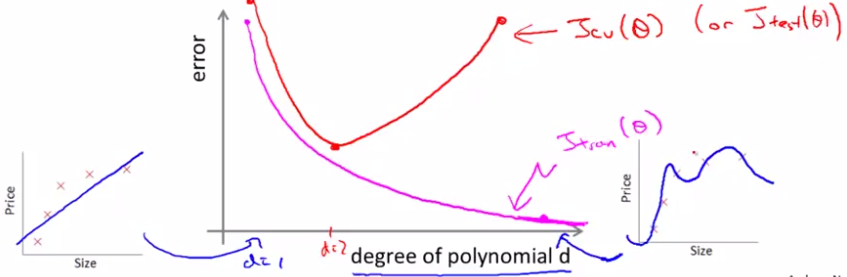
* i.e. the **average squared error** as measured on either set
* Now let's plot the hypothesis error as a function of the degree of polynomial d



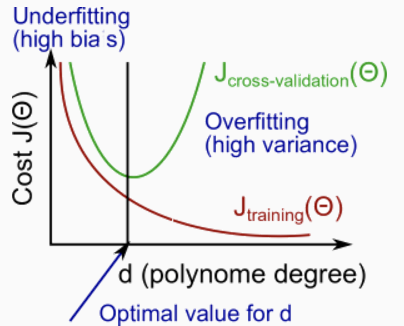
* Where maybe d = 1, we’re fitting very simple functions, when d = 4 or more, we’re fitting very complex, high-order polynomials that might fit the training set w/ much more complex functions
* Let's start with the training error.
* As we increase the degree of the polynomial, we're going to fit our training set better + better and so, if d = 1, we’ll have a very high training error.
* If we have a very high-degree polynomial, our training error is going to be very low, maybe even 0 if it fit the training set that well.
* As we increase of d, typically the training error, J\_train(Ө), decreases



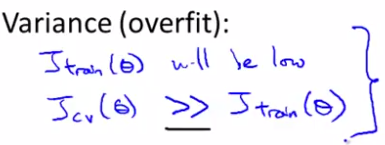
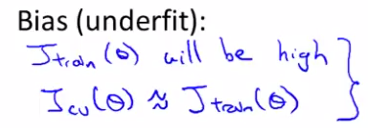
* Next, let's look at the CV error (if we were to plot the test set error, we'd get a pretty similar result to the CV error)
* If we fit an intermediate-degree polynomial, d= 2, we’re going to have a much lower CV error b/c we find a much better fit to the data.
* Conversely if d were too high, d = 4, then we're again *overfitting* + end up w/ a high value for J\_cv(Ө)



* This plot also helps us to better understand the notions of bias + variance.
* If you have a learning algorithm + the CV or test set error is high, we need to figure out if the algorithm suffering from high bias or high variance.
* When CV error is high, this corresponds to 1 of 2 regions
* The region w/ the lower degree d polynomial corresponds to a **high bias problem** (fitting an overly-low order polynomial when we really needed a higher-order polynomial to fit the data)
* The region w/ the higher degree d polynomial corresponds to a high **variance problem** (if d was too large for the data set that we have)



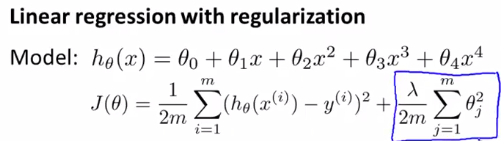
* For the high-bias case (under-fitting), what we find is that both the CV *AND* training error are going to be high.
* If you see this combo, that's a sign your algorithm may be suffering from high bias.
* If your algorithm is suffering from high variance, the training error is going to be low (fitting training set very well) but the CV error is much bigger
* The key that distinguishes these 2 cases is:
* If you have a **high bias** problem, the training set + CV set error will *both* be
* If you have a **high variance** problem, the training set error will usually be lower than the CV error



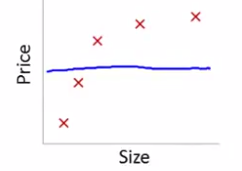
* By diagnosing whether a learning algorithm is suffering from high bias or high variance, we get much better guidance for what might be promising things to try in order to improve the performance of the learning algorithm

1. **REGULARIZATION AND BIAS/VARIANCE**

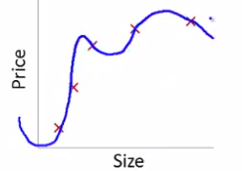
* Suppose we're fitting a high-order polynomial + to prevent over-fitting use regularization to try to keep the values of the parameters (weights) small.



* Consider 3 cases.
* Very large value of the regularization parameter λ, such as 10,000
* In this case, all parameters, Ө1-Өm, would be *heavily* penalized, so we’d end up w/ most parameter values being closer to 0 + the hypothesis hӨ(x) will be approximately = Ө0
* We end up w/ a hӨ(x) is more or less a flat, constant straight line, which high bias + badly underfits the data set



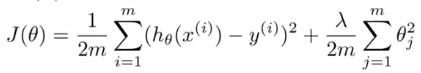
* The other extreme is if we have a very small value of λ, such as near 0
* In that case, given we're fitting a *high order* polynomial, this is typical over-fitting scenario.
* We're fitting a high-order polynomial pretty much *without regularization* or w/ very minimal regularization (penalization), so we end up w/ high-variance in an overfitting situation.



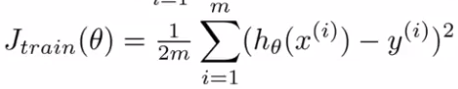
* It's only if we have some intermediate value of λ (neither too large nor too small) that we end up w/ parameters Ө that give us a reasonable fit to the data.
* So, how can we automatically choose a good value for the regularization parameter λ?
* Just to reiterate, here's our model:



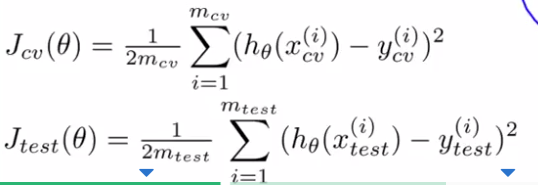
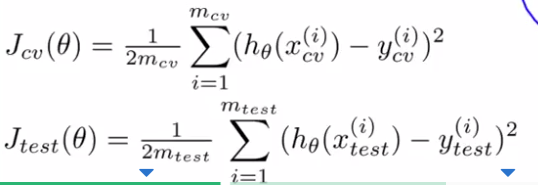
* And here's our learning algorithm's objective 🡺 to minimize:



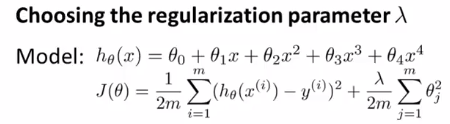
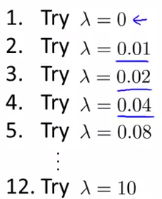
* Then define J\_train(Ө) to be different to be the optimization objective 🡪 take away the regularization term.



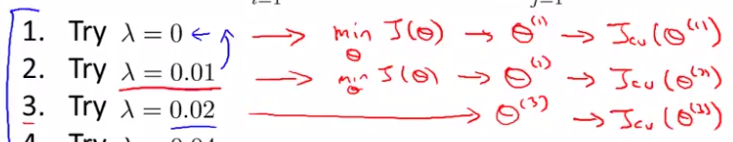
* When using regularization, J\_train(Ө) is no longer the same as J(Ө), the original cost function
* It is just the **sum of squared errors (SSE)**, or the **average squared error,** on the training set w/out taking into account regularization.
* Similarly define the CV + test set errors to be the average SSE on the CV + test sets

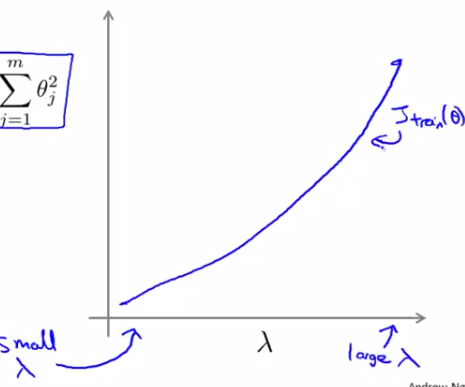
* So our 3 set errors are just 1/2 of the average squared error of each set, w/out an extra regularization term.
* We can automatically choose a regularization parameter λ is to maybe have some range of values of λ to try out, such as 0.01, 0.02, 0.04, + so on, maybe in multiples of 2, until some larger value like 10

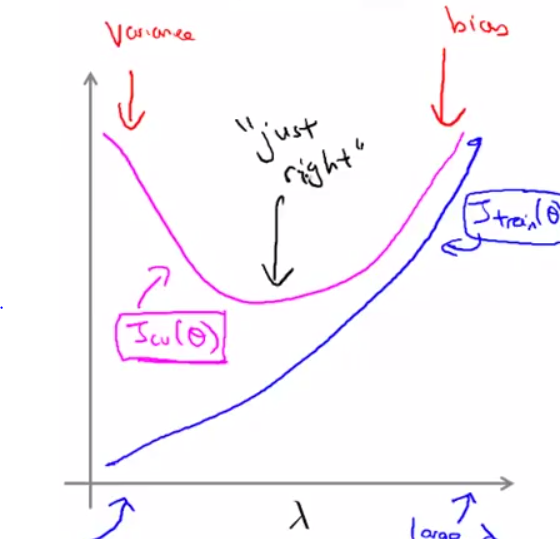
* This gives maybe 12 different model, and what we can do is take the 1st first model w/ λ = 0 + minimize the cost function J(Ө) to get some parameter Ө1
* Then take the 2nd model w/ λ = 0.01, minimize J(Ө) again, + get some different parameter vector Ө2
* Go so on up to Ө(12).
* Then take all the hypotheses these parameters + use the CV set to validate them
* We use the models fit w/ different values of the regularization parameter + evaluate them w/ the CV set to measure the average square error of each of the parameter vectors in the CV set.

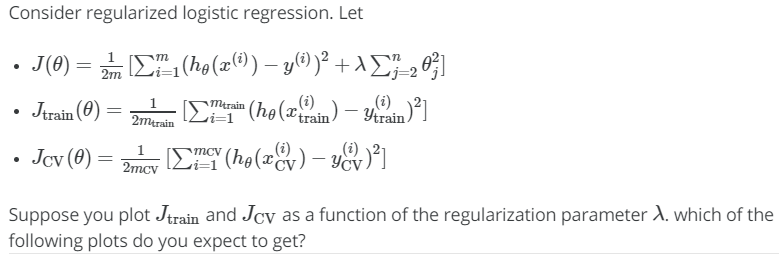
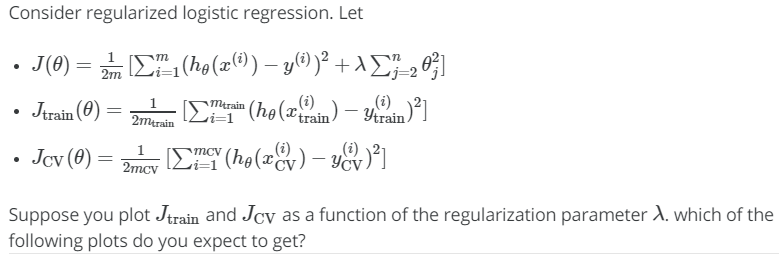
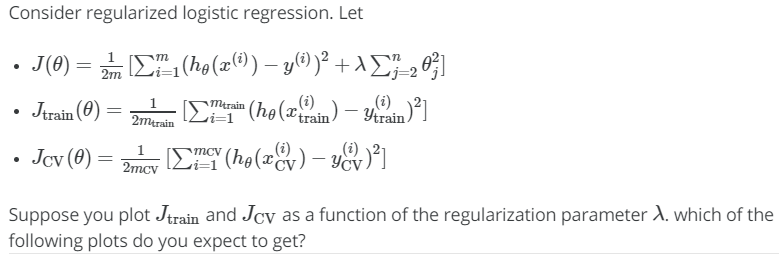


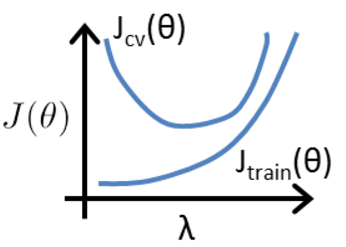
* Then pick whichever model gives the lowest error on the CV set, say Ө5,
* Finally, if we wanted to report the test set error, is to take this parameter Ө5 + look at how well it does on the test set.
* Since we fit our parameter to the CV set, we set aside a separate test set to use to get a better estimate of how well our selected parameter vector will generalize to previously unseen examples.
* So that's model selection applied to selecting the regularization parameter λ.
* Now to get a better understanding of how CV + training error vary as we vary λ.
* Plot J\_train + J\_cv as a means of seeing how well does my hypothesis does on the training + CV sets as we vary λ.
* If λ is small, we're not using much regularization + run a larger risk of overfitting, whereas if λ is large, we run the higher risk of having a biased problem
* What we find is that for small values of λ, we can fit the training set relatively way b/c we're not regularizing, as λ basically goes away you're just minimizing pretty much just gray arrows.
* So, when λ is small, you end up w/ a small value for J\_train
* Whereas if λ is large, you have a high bias problem, + you might not fit your training that well, so



* So J\_train(Ө) will tend to increase when λ increases, b/c a large λ corresponds to high bias where we might not even fit a training set well
* A small value of λ corresponds to maybe fitting a very high degree polynomial to the data
* For the CV error, if we have a large value of λ, we may end up underfitting (high bias), + the CV error will be high.
* So, w/ high bias, we won't be doing well in either set
* Whereas w/ high variance w/ smaller λ’s, we may be overfitting data
* By overfitting, the CV error will be high.



* Once again, it will often be some intermediate value of λ that is just right/works best, in terms of having a small CV error or a small test error
* On the real data set, the curves you get may end up looking more messy + noisy than above
* For some data sets, you will really see those trends
* By looking at a plot of the CV error, you can manually select a point for λ that minimizes the CV error
* When trying to pick a regularization parameter λ for a learning algorithm, often plotting a figure like this helps understand better what's going on + helps verify we are indeed picking a good value for λ.
*   



**III. LEARNING CURVES**

**IV. DECIDING WHAT TO DO NEXT REVISITED**