Advice for Applying Machine Learning

فيه بعض الحاجات الى من خلالها بقدر انى اوفر وقت كبير جدا انى ال algorithm يقدر يساعدنى فى أنه يعمل predict كويس للحاجه زى انى ممكن اختر features معينه مش كل ال features بجانب انى ممكن أقلل او اكبر ال lambda او انى اخليها بدل ما هى linear equation تكون polynomial equation بس لازم اختيار حاجه من الحاجات ديه او غيرها ميكونش randomly يعنى المفروض تكون عارف بتختار ديه ليه.

Some of these help us to minimize error are

- Getting more training examples
- Trying smaller sets of features
- Trying additional features
- Trying polynomial features
- Increasing or decreasing λ

من المهم جدا تقسم الداتا الى عندك ل training and testing set ولو الداتا sorted لازم تخليها randomly unordered عشان الموديل بتاعك يقدر ي gernlize ال fitting.
انا بقسم الداتا عشان ممكن يكون الافتراض بتاعى الى طلع من ال training examples يكون weights بتاعته مش شغاله كويس مع ال testing set لذلك بنقسم الداتا الى عندنا ل training and testing set
:The new procedure using these two sets is then

1. Learn Θ and minimize Jtrain(Θ) using the training set
2. Compute the test set error Jtest(Θ)

Model Selection and Train/Validation/Test Sets

فيه طريقه تانيه احسن من انى اشتغل علطول من بعد ما اعمل traing على testing set وهى انى أقسم الداتا training, validation and testing set لذلك ممكن استخدم ال approach ده بعد اما أقسم الداتا انى اعمل train باستخدام different degree equstion مثلا لحد d=10 الى هو معادله من الدرجه العاشره وبعدها اختار أقل معادله طلعت error وابتدى اخد قيم ال thetas الخاصه بيها واشتغل على validtion dataset وبعدها أقدر بقا اشوف ده هايعمل ايه مع ال test .dataset

One way to break down our dataset into the three sets is:

- Training set: 60%
- Cross validation set: 20%
- Test set: 20%

We can now calculate three separate error values for the three different sets using the following method:

1. Optimize the parameters in Θ using the training set for each polynomial degree.
2. Find the polynomial degree d with the least error using the cross validation set.

3. Estimate the generalization error using the test set with Jtest(Θ(d)), (d = theta from polynomial with lower error);

This way, the degree of the polynomial d has not been trained using the test set.

Bias vs. Variance

High bias means underfitting, high variance means overfitting.

ال algorithm بتاعى ممكن يكون مش شغال كويس فى ال generalization بعد اما اعمل training حتى لو كان بي fit data very will ده مش معناه انه هيشتغل كويس مع ال unseen data بس الفكرة هنا هو ازاى اعرف الموديل بتاعى فيه high bias or high variance يعنى under fitting or over fitting وده بقدر اعرفه لما بشتغل على ال training dataset واطلع مثلا 5.= error واجى فى ال cross validation data set احسب ال cost function القيها بقت اعلى من الى طلعت فى ال training cost function هنا بيقا الموديل بتاعى فيه high variance يعنى over fitting والعكس لو كان ال cost of cross validtion اقل من ال cost of training data set ده معناه انه high bais يعنى under fitting.

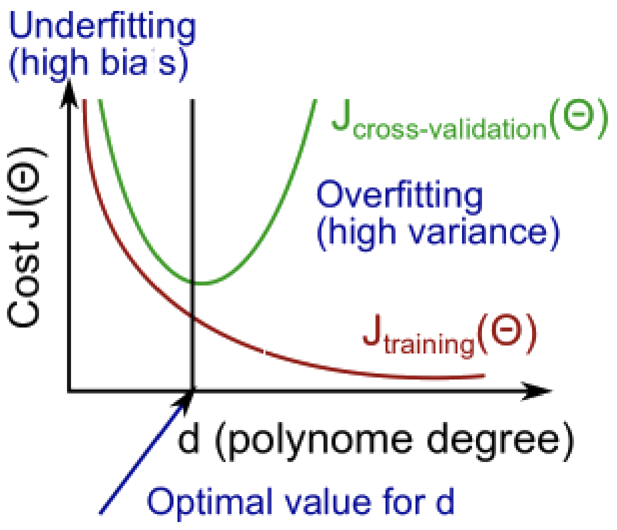
- We need to distinguish whether **bias** or **variance** is the problem contributing to bad predictions.

لذلك احنا محتاجين نلاقى نقطة وسط بينهم الى هى تعبر عن ال optimization hypothesis يعنى ميكونش بي overfit or underfit وده بيساعدنى انى يكون الموديل بتاعى generalization with unseen data.

ما بجى اشتغل واحاول انى اصغر ال cost function عن طريق مثلا انى ازود فى ال degree بتاعت المعادلة ده بيزود نسبة ال over fitting وفى نفس الوقت مع ال training examples بيقل دايما لكن مع ال cross validation بيحصل انى بعد درجه معينه من ال degree ال cost بتبتدى تزيد.

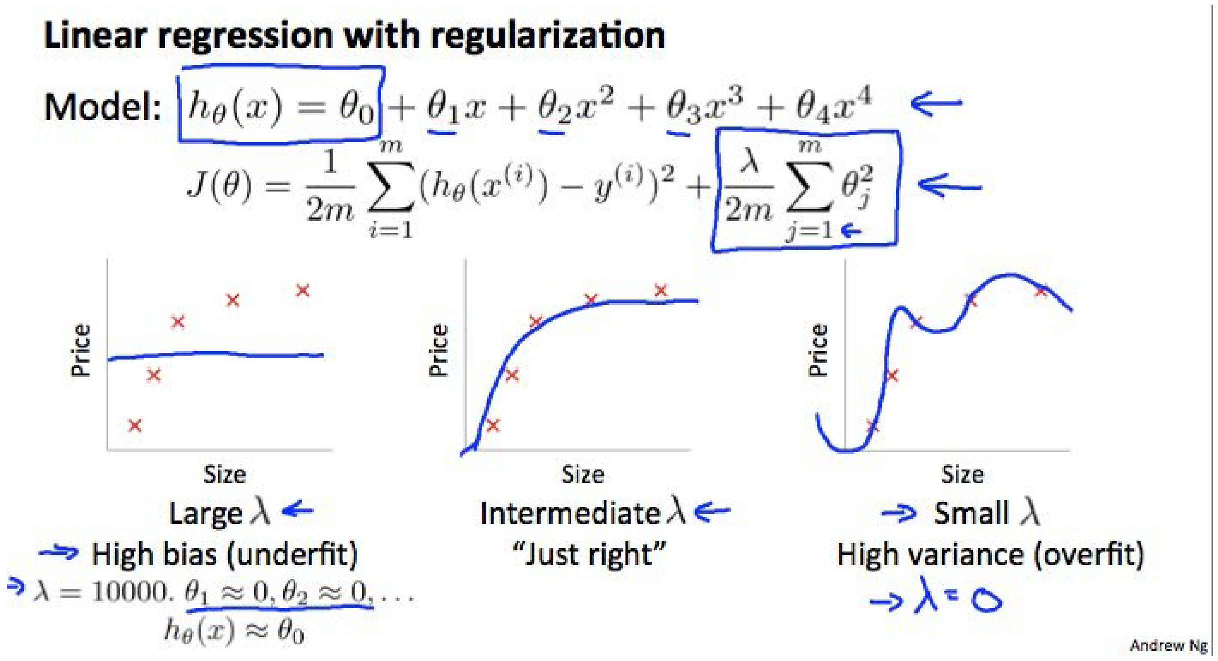
The training error will tend to **decrease** as we increase the degree d of the polynomial.

At the same time, the cross validation error will tend to **decrease** as we increase d up to a point, and then it will **increase** as d is increased, forming a convex curve.



High bias (underfitting): both Jtrain(Θ) and JCV(Θ) will be high. Also, JCV(Θ)≈Jtrain(Θ).

High variance (overfitting): Jtrain(Θ) will be low and JCV(Θ) will be much greater than Jtrain(Θ).


Linear regression with regularization
Model: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$
Price
Size
Large λ
High bias (underfit)
$\lambda = 10000.\ \theta_1 \approx 0, \theta_2 \approx 0, \dots$
$h_\theta(x) \approx \theta_0$
Intermediate λ
"Just right"
Small λ
High variance (overfit)
$\lambda = 0$
Andrew Ng

In the figure above, we see that as lambda λ increases, our fit becomes more rigid. On the other hand, as lambda λ approaches 0, we tend to over overfit the data. So how do we choose our parameter lambdaλ to get it 'just right' ? In order to choose the model and the regularization term λ, we need to:

1. Create a list of lambdas (i.e. λ∈{0,0.01,0.02,0.04,0.08,0.16,0.32,0.64,1.28,2.56,5.12,10.24});
2. Create a set of models with different degrees or any other variants.
3. Iterate through the lambdaλs and for each lambda λ go through all the models to learn some Θ.
4. Compute the cross validation error using the learned Θ (computed with λ) on the JCV(Θ) **without** regularization or λ = 0.
5. Select the best combo that produces the lowest error on the cross validation set.
6. Using the best combo Θ and λ, apply it on Jtest(Θ) to see if it has a good generalization of the problem.

If a learning algorithm is suffering from **high bias**, getting more training data will not **(by itself)** help much.

If a learning algorithm is suffering from **high variance**, getting more training data is likely to help.

Deciding What to Do Next Revisited

Our decision process can be broken down as follows:

- **Getting more training examples:** Fixes high variance
- **Trying smaller sets of features:** Fixes high variance
- **Adding features:** Fixes high bias
- **Adding polynomial features:** Fixes high bias
- **Decreasing λ:** Fixes high bias
- **Increasing λ:** Fixes high variance.

Diagnosing Neural Networks

- A neural network with fewer parameters is **prone to underfitting**. It is also **computationally cheaper**.
- A large neural network with more parameters is **prone to overfitting**. It is also **computationally expensive**. In this case you can use regularization (increase λ) to address the overfitting.

Using a single hidden layer is a good starting default. You can train your neural network on a number of hidden layers using your cross validation set. You can then select the one that performs best.