1 Machine learning diagnostics

When the prediction of a hypothesis trained by a learning algorithm turns out to have unacceptable errors, a series of options are available as promising further actions to take:

1. Get more training examples.
2. Try smaller sets of features.
3. Try additional features.
4. Try polynomial features.
5. Adjusting value of regularization parameter λ.

Rather than randomly try different approaches to improve the performance of the algorithm, it is worthwhile to implement machine learning diagnostics, which apply certain tests to figure out what is / isn't working and can thus provide guidance on what should be done next.

1.1 Evaluate a hypothesis: test set

Training the algorithm with all the examples available might end up overfitting the training set. To avoid such possibility, we can divide all examples into the **training set** and the **test set**. A typical choice is to put 70 % of the examples in the training set and 30% of the examples in the test set. Then a series of parameters θ can be learned from the training set by minimizing $J(\theta)$, after which the functionality of the algorithm can be measured by the test set error $J_{test}(\theta)$. For linear regression, we have

$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$$

Similarly, for logistic regression, we have

$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} (y_{test}^{(i)} \log(h_\theta(x_{test}^{(i)})) + (1 - y_{test}^{(i)}) \log(1 - h_\theta(x_{test}^{(i)}))) \quad (1)$$

For logistic regression (i.e. 2-classification) problem, another way to define the test set error is

$$J_{test}(\theta) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err(h_\theta(x_{test}^{(i)}), y_{test}^{(i)})$$

in which

$$err(h_\theta(x, y) = \begin{cases} 1, \text{ if } y = 0, h_\theta(x) \geq 0.5 \text{ or } y = 1, h_\theta(x) < 0.5 \\ 0, \text{ otherwise} \end{cases}$$

1.2 Model selection: cross-validation set

In order to choose among different models (e.g. different degrees of polynomial), we need further partition of the examples. All examples should now be divided into 3 groups: **training set**, **cross-validation set** and **test set**. Typically the percentages of the 3 groups are 60%, 20% and 20%.

For each training model candidate $h_\theta^{(i)}(x)$, we train a series of parameters $\theta^{(i)}$ by minimizing $J(\theta)$. Then we calculate the cross-validation set error $J_{cv}(\theta^{(i)})$ for them, in which

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

for linear regression, and similar as (1) for logistic regression. Finally, the model that minimizes $J_{cv}(\theta)$ stands out, and its performance can be measured with the test set error.

1.3 Bias & variance

When the training set is underfitted, the algorithm is said to have high bias, while when the training set is overfitted, it is said to have high variance. We will introduce methods to diagnose such problems.

In a specific linear regression problem, when higher order items are introduced into the hypothesis, the training set error (regularization not considered) will be reduced. However, too many high order items may cause overfit problem, which will increase the cross-validation set error. Figure 1 depicts such trend.

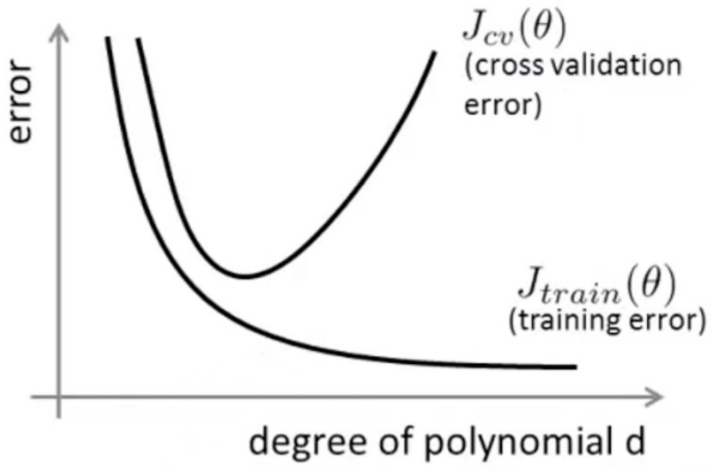

Figure 1: Diagnosing bias and variance with errors: different polynomial degrees

It is obvious that in a high bias (underfit) case, both $J_{cv}(\theta)$ and $J_{train}(\theta)$ are high, while in a high variance (overfit) case, $J_{cv}(\theta)$ is high but $J_{train}(\theta)$ is low.

We introduced regularization as the method to combat overfit, but the choice of the regularization parameter λ can be subtle. It turns out that an appropriate λ can be chosen following a criteria similar to that of model selection. Also, we can diagose bias and variance following the same rule stated above, as shown in Figure 2. Here the $J_{train}(\theta)$ does not include the regularization item.

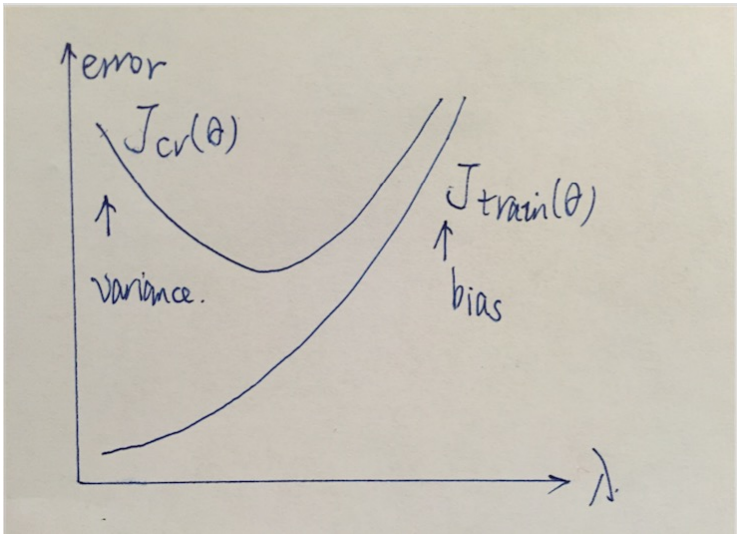

Figure 2: Diagnosing bias and variance with errors: different regularization parameter

1.4 Learning curve

Learning curve is the curve that depicts the variance of the training error and the cross-validation error with the training set size m. It can help estimate whether or not the performance of the algorithm will be improved by adding more examples into the training set.

When more examples are added into the training set, the hypothesis can no longer fit all examples well, thus the training error $J_{train}(\theta)$ will increase, whereas the hypothesis extends better towards new examples, thus the cross-validation error $J_{cv}(\theta)$ decreases. A typical learning curve is shown in Figure 3.

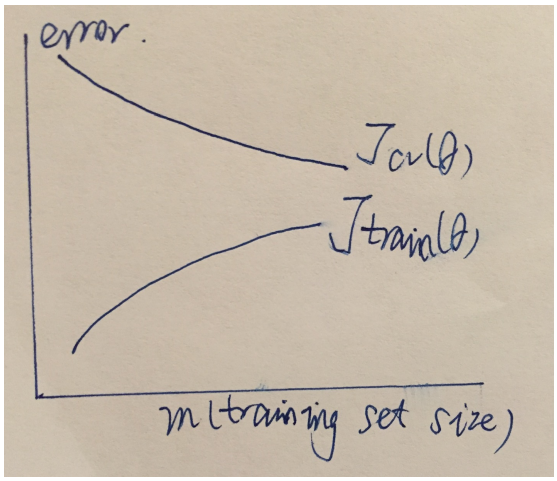

Figure 3: A typical learning curve

In a high-bias case, the examples are underfitted (e.g. trying to fit examples that follow a 5-degree polynomial with a straight line). Even if the size of the training set is increased, the hypothesis will still fail to correctly describe the relationship between the input and the output. Thus, both $J_{cv}(\theta)$ and $J_{train}(\theta)$ will be high, and they will be quite close to each other, as shown in Figure 4. In this case, **collecting more training examples is not likely to help.** Essential adjustment to the hypothesis should be made, i.e. adding more features.

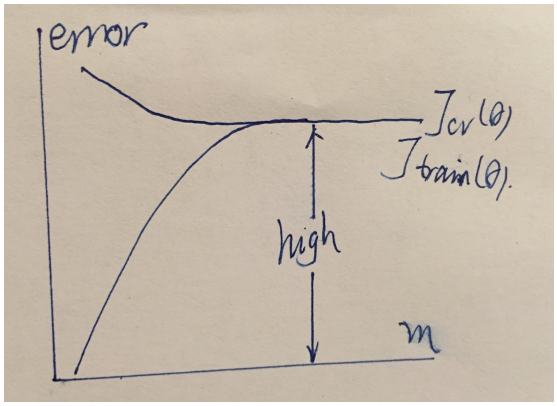

Figure 4: Learning curve with high bias

In a high variance case, the examples are overfitted. $J_{train}(\theta)$ is typically lower than $J_{cv}(\theta)$, as shown in Figure 5. If more examples are added to the training set, the training set becomes "less overfitted", thus $J_{train}(\theta)$ is likely to increase, while $J_{cv}(\theta)$ will decrease. In such case, **collecting more training**

examples will help to improve the performance.

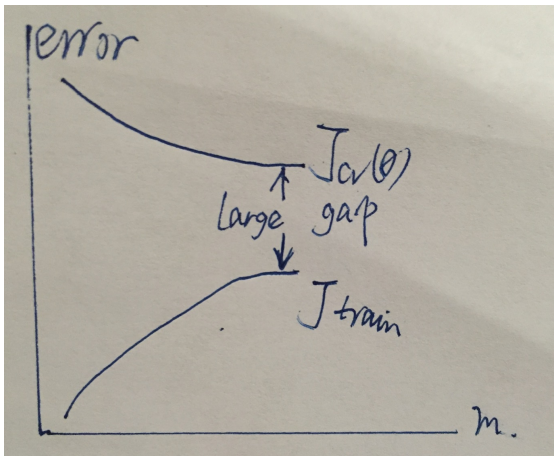

Figure 5: Learning curve with high variance

1.5 Conclusion

We will now conclude this section by providing the effects of the solutions to performance problems listed at the begining.

Get more training examples fixes high variance

Try smaller sets of features fixes high variance

Try additional features fixes high bias

Try polynomial features fixes high bias

Increase regularization parameter λ fixes high variance

Decrease regularization parameter λ fixes high bias

2 Machine learning system design

When designing a machine learnig system, a bunch of different options are available for choice to achieve better performance. Take the example of an email spam classifier, we can:

- Collect large amount of data.
- Develop sophisticated features based on email routing information, e.g. email header.

- Develop sophisticated features for message body, e.g. treating 'discount' and 'discounts', 'dealer' and 'Dealer' as the same word, ignoring punctuations, etc.
- Develop sophisticated algorithm to detect misspellings, e.g. 'magaz1ne', 'm0rtgage', etc.

Rather than randomly making a choice based on gut feeling, systematic analysis methods are available to help choose the best option.

2.1 Error analysis

A recommended approach to a machine learning problem is as follows. Start with a "quick and dirty" version of the algorithm easy to implement. Implement it and test it on the cross-validation set. Plot learning curves to decide if more data or more features are needed. Then conduct what we call **error analysis**: manually examine examples in c-v set on which the algorithm made errors, and try to figure out any systematic trend of such examples. Based on the trends, add new features to the model and see if it helps.

2.2 Handling skewed data

Take the cancer classification as an example. If we have a learning algorithm that diagnoses the test set with 1% error, i.e. 99% of the diagnostics it make are correct, it seems to be a good algorithm. However, provided that only 0.5% of the patients in the training set have cancer, this algorithm is far from satisfactory. Actually, a dumb classifier that always outputs $y = 0$ (no cancer) appears to have better performance. Obviously, in such case, classification error, or classification accuracy, is no longer an appropriate metric of the performance of the algorithm.

Table 1: Definitions for skewed data error metrics

Predicted \ Actual	1	0
1	True Positive	False Positive
0	False Negative	True Negative

Table 1 provides some useful definitions for skewed data error metrics. If the actual class is 1 and the algorithm predicts 1, we say that the algorithm has made a true positive prediction, etc. Now we can define the **precision** and the **recall** of the algorithm.

$$\begin{aligned} \text{Precision} &= \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \\ \text{Recall} &= \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \end{aligned} \tag{2}$$

Clearly, precision is the percentage of patients who actually have cancer among patients diagnosed to have cancer, while recall is the percentage of patients who are diagnosed to have cancer among patients who actually have cancer.

With precision and recall defined, we can introduce the tradeoff between them. Normally we predict $y = 1$ when $h_\theta(x) \geq 0.5$ and $y = 0$ when $h_\theta(x) < 0.5$. If we want to predict $y = 1$ only when very confident, we can increase the threshold for prediction, e.g. to 0.9. Obviously this will increase the precision but decrease the recall. On the contrary, if we want to alert more patients possible to have cancer, we can decrease the threshold, say to 0.3, and we will end up with lower precision but higher recall.

Taking both precision and recall into account, we define the F_1 score of the algorithm

$$F_1 = \frac{2PR}{P + R} \tag{3}$$

and use it as a measurement of the performance of the algorithm. Generally, an algorithm with a better F_1 score has better overall performance.

2.3 Data

Sufficient and appropriate data is essential to the performance of the algorithm. We should always ensure that the features we use include **enough** information to predict the output correctly. A useful test to determine whether the information is enough is to ask ourselves: is a human expert in this field capable of making a confident predication of the output based on the information we provide?

If we are using low-bias algorithms, e.g. neural network with a lot of hidden layers or a linear regression with a lot of feature, large amount of data will help avoid overfitting and is thus usually preferable.