Stanford Machine Learning - Week VI

Eric N Johnson

August 21, 2016

1 Evaluating a Learning Algorithm

If we are going to employ machine learning algorithms we should know how to choose the most effective avenue to follow.

1.1 Deciding What to do Next

Suppose we have implemented a regularized linear regression to predict housing prices. However, when we test the hypothesis on a new set of houses we get unacceptably large errors in its predictions. What should we try next?

- Get more training examples
- Try smaller sets of features - are some redundant?
- Try additional features - are we capturing the ones that matter?
- Try adding polynomial features
- Try altering λ...

We should not be picking one of these at random! Fortunately there is a simple technique that will help us make intelligent decisions called a *Machine Learning Diagnostic*.

1.2 Evaluating a Hypothesis

When we apply an algorithm, we minimize some quantity like the residual... But we also know we can over fit data. We call this the *bias / variance trade-off*. How do we evaluate our hypothesis to have a good balance?

- We can split our original training set into two smaller sets: A training set (70%) and a test set (30%).
- Let m_{test} to denote the number of test examples $\left(x^{(i)}_{\text{test}}, y^{(i)}_{\text{test}}\right)$.
- Obviously if we are overfitting we should expect the training error to be small and the test error to be high.
- Here's a procedure for testing linear regression:
  1. Learn the parameter θ from the training data.
  2. Compute the test set error:

$$J_{\text{test}}(\theta) = \frac{1}{2m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} \left(h_\theta(x^{(i)}_{\text{test}}) - y^{(i)}_{\text{test}}\right)^2$$

- Here's a similar procedure for testing logistic regression:
  1. Learn the parameter θ from the training data.
  2. Compute the test set error:

$$J_{\text{test}}(\theta) = -\frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} y^{(i)}_{\text{test}} \log(h_\theta(x^{(i)}_{\text{test}}) + (1 - y^{(i)}_{\text{test}}) \log(h_\theta(x^{(i)}_{\text{test}}))$$

  3. Find misclassification errors (also known as 0/1 misclassification error):

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

4. Then find the test error:

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

1.3 Model Selection and Training, Validation and Test Sets

We already know that the training set error is not a good predictor for how well an algorithm will perform on real data. Lets consider the model selection problem. Suppose we need to chose a model...

$$\begin{aligned}
(1)\ \ & h_\theta(x) = \theta_0 + \theta_1 x \\
(2)\ \ & h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \\
(3)\ \ & h_\theta(x) = \theta_0 + ... + \theta_3 x^3 \\
& \vdots \\
(10)\ & h_\theta(x) = \theta_0 + ... + \theta_{10} x^{10}
\end{aligned}$$

We can think of this as another parameter d: what order polynomial do we want to use? Denote $\theta^{(i)}$ as the value of the theta parameters from these ten models. We could find...

$$\begin{aligned}
(1)\ \ & h_\theta(x) \to \theta^{(1)} \to J_{\text{test}}(\theta^{(1)}) \\
(2)\ \ & h_\theta(x) \to \theta^{(2)} \to J_{\text{test}}(\theta^{(2)}) \\
(3)\ \ & h_\theta(x) \to \theta^{(3)} \to J_{\text{test}}(\theta^{(3)}) \\
& \vdots \\
(10)\ & h_\theta(x) \to \theta^{(10)} \to J_{\text{test}}(\theta^{(10)})
\end{aligned}$$

then chose (5)... But does this model generalize well? The problem here is that (5) is likely to be an overly optimistic estimate of the generalized error.

Instead, we split the training set into three parts, and define errors.

- The training set (60%) with Training Set Error:

$$J_{\text{train}} = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- The test set (20%), with Test Error:

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

- and the cross-validation set (20%), denoted $(x_{cv}^{(i)}, y^{(i)})$, with Cross-Validation Error:

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

Instead of arbitrarily choosing a model, we test each of the models to find the lowest cross-validation error:

$$\begin{aligned}
(1)\ \ & h_\theta(x) \to \theta^{(1)} \to J_{cv}(\theta^{(1)}) \\
(2)\ \ & h_\theta(x) \to \theta^{(2)} \to J_{cv}(\theta^{(2)}) \\
(3)\ \ & h_\theta(x) \to \theta^{(3)} \to J_{cv}(\theta^{(3)}) \\
& \vdots \\
(10)\ & h_\theta(x) \to \theta^{(10)} \to J_{cv}(\theta^{(10)})
\end{aligned}$$

If we find that (4) has the lowest cross-validation error, for example, we still have the test set to use for validation.

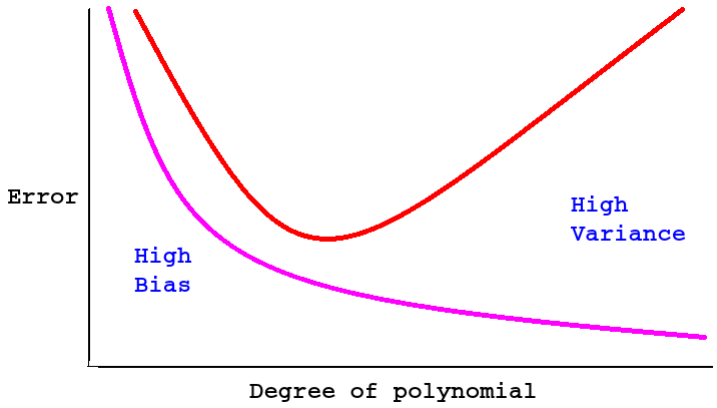
2 Bias Verses Variance

Most of the time when a model isn't working as well as hoped there is either a *bias problem* or a *variance problem*. Which of these is present gives useful clues about what is going wrong. Recall that an underfit model has *high bias*, and an overfit model has *high variance*.

2.1 Diagnosing Bias vs. Variance

Suppose we plot error with respect to d using the training error and cross-validation error as defined previously.

- Training error $J_{\text{train}}(\theta)$
- Cross-validation error $J_{cv}(\theta)$



This plot is helpful because we can use it to determine whether a model has a bias problem or variance problem.

- High Bias (Underfit model).
  - $J_{\text{train}}(\theta)$ will be high
  - $J_{\text{train}}(\theta) \approx J_{cv}(\theta)$
- High Variance (Overfit model).
  - $J_{\text{train}}(\theta)$ will be low
  - $J_{\text{train}}(\theta) << J_{cv}(\theta)$

2.2 Regularization and Bias / Variance

What effect does regularization have on Bias and Variance?

Suppose we are fitting a linear model with regularization:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{m} \theta_j^2$$

And consider three cases:

- Large λ, high bias (underfit), $\lambda = 10000$, $\theta_0 \approx 0$, $\theta_1 \approx 0$, ..., $\theta_4 \approx 0$, $h_\theta(x) \approx 0$.

- Intermediate λ "just right"

- Small λ, high variance (overfit)

If we use regularization, define the training set, cross-validation set, and test set error as the sum of square errors *without* the regularization terms.

$$J_{\text{train}}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)} - y^{(i)} \right)^2$$

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

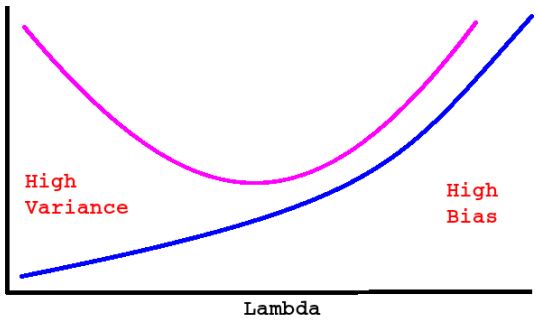
$$J_{\text{test}}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x_{\text{test}}^{(i)} - y_{\text{test}}^{(i)} \right)^2$$

Then deciding which model to pick, try different values of λ...

1. Try $\lambda = 0$
2. Try $\lambda = 0.01$
3. Try $\lambda = 0.02$

   ⋮

then pick the model with the lowest cross-validation error.
If we plot $J_{\text{train}}(\theta)$ and $J_{\text{cv}}(\theta)$ against λ, we will likely get a plot like this:



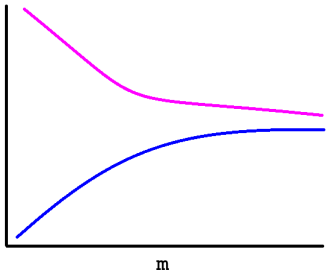
where

- Cross-Validation Error $J_{\text{cv}}(\theta)$
- Training Error $J_{\text{train}}(\theta)$

2.3 Learning Curves

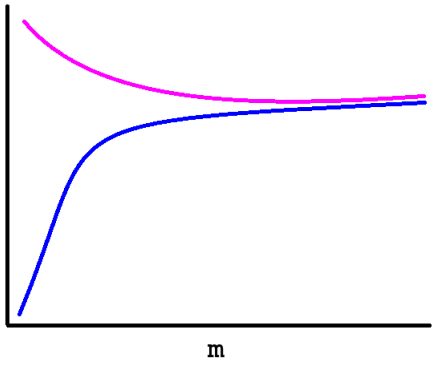
Learning curves are a useful way of visualizing how well a machine learning algorithm is working. We plot either $J_{\text{train}}(\theta)$ or $J_{\text{cv}}(\theta)$ with respect to artificially varied training set sizes, m. When m is very small, the error for a quadratic hypothesis, for example, will be very small. But as the training set gets larger, the error will grow. As m increases, however, the cross-validation error should get smaller.

- Cross-Validation Error $J_{\text{cv}}(\theta)$
- Training Error $J_{\text{train}}(\theta)$

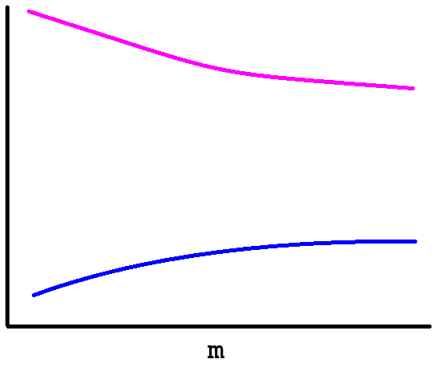


What will this start to look like if there is a problem?

- In the High Bias case, $J_{\text{cv}}(\theta)$ and $J_{\text{train}}(\theta)$ will get very close as m increases in size. Also, getting more training data will not help.



- In the High Variance case, the two curves are very far apart. A very large number of training examples will be required for $J_{\text{cv}}(\theta)$ and $J_{\text{train}}(\theta)$ to be close.



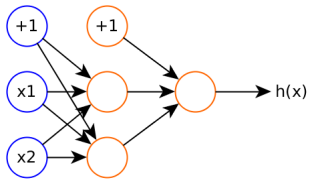
2.4 Deciding what to do next (revisited)

How do these things help us determine what we can do to improve our algorithm?

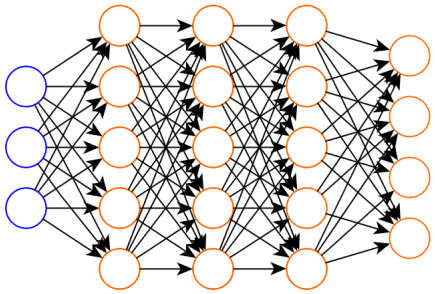
- Get more training examples: → Fixes high variance
- Try smaller sets of features - are some redundant? → Fixes high variance
- Try additional features - are we capturing the ones that matter? → Fixes high bias
- Try adding polynomial features → Fixes high bias
- Try altering λ...
  - Decreasing λ → Fixes high bias
  - Increasing λ → Fixes high variance

2.5 Neural Networks and Overfitting

- Small neural networks with fewer parameters are more prone to underfitting. They are computationally cheaper.



- Large neural networks with more parameters are prone to overfitting. They are computationally more expensive.



  Regularization can be used to address overfitting.

- Often using a larger neural network with regularization performs best. The computational cost is often not an issue. The number of hidden layers can be determined by using a cross-validation set and testing their performance.

3 Machine Learning System Design

We will discuss what things to do when designing a learning system.

3.1 Prioritizing What to Work On: Building a Spam Classifier

- Building a spam classifier:

$$x = \text{features of email}$$
$$y = \text{spam (1) or not spam (0)}$$

  we could, for instance, choose 100 words that are indicative of spam / not spam.

Pick some words, and build a binary vector for whether the email contains the word or not.

$$x = \begin{bmatrix} Eric \\ Buy \\ Viagra \\ deal \\ discount \\ now \\ \vdots \\ Cialis \\ cheap \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

In practice we may take the most frequently occurring n words (10,000 - 50,000) and use them.

How can we spend our design time to make it have low error?

- Collect lots of data.
- Create a honeypot email to collect Spam
- Develop something that looks at features contained in email header like routing information.
- Look at the email message body. Should "discount" and "discounts" be treated as the same word?
- Develop something to detect intentional misspellings like "W4tches" or "Ca$h".

  Which feature will be most illuminating may not be at all obvious! Don't pick some approach arbitrarily.

3.2 Error Analysis

Here is a recommended approach to designing a machine learning algorithm.

- Start with a simple algorithm that you can implement quickly. Implement it to test your cross validation data.
- Plot learning curves to decide if more data, more features, etc. will likely help.
- Error analysis: Manually examine the examples in the cross validation set that your algorithm made errors on. See if you can spot any systematic trend in what type of examples is making errors on.
- Avoid premature optimization!

Here is an example. m_{cv} = 500 examples in cross validation set. The algorithm misclassifies 100 emails. Manually examine these 100 errors and categorize them based on:

- What type of email it is...
  - emails trying to sell drugs
  - emails trying to steal passwords
  - emails trying to verify that the email is valid (and used)
- What cues (features) you think would have helped the algorithm classify them correctly...
  - Weird email return addresses
  - Unusual punctuation
  - Strange spellings...

3.3 Numerical Evaluation

It is very helpful to have a way to numerically evaluate our algorithm.

- For example, should we use "stemming" software (like Porter Stemmer) to treat similar words equally? (This is a natural language process package.) We need to evaluate our model with and without stemming.
- Should we ignore case? It would be helpful to numerically evaluate whether ignoring case will help or not.

4 Error Metrics for Skewed Classes

Skewed classes: Skewed distributions that are asymmetrical and have data that clusters toward one end.

For example: We have far more examples from one class than another because one class has a low probability of occurring. In this case, we can have very high classification accuracy - but we still don't capture the low probability classes correctly.

4.1 Precision and Recall

Precision / Recall:
We can use ratios of correct vs. incorrect responses to get some useful metrics.

Let $y = 1$ in presence of rare classes we want to detect (by convention). Create a truth table...

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

- **Precision**:

$$\text{Precision} = \frac{\text{True Positives}}{\text{Number of Predicted Positives}} = \frac{N(\text{True Positives})}{N(\text{True Positives}) + N(\text{False Positives})}$$

- **Recall**:

$$\text{Recall} = \frac{\text{True Positives}}{\text{Number of Actual Positives}} = \frac{N(\text{True Positives})}{N(\text{True Positives}) + N(\text{False Negatives})}$$

4.2 Trading Off Precision and Recall

There will always be a trade-off between precision and recall.

We can change the threshold values for classification. Say,

$$h_\theta(x) \geq 0.9 \rightarrow \text{predict } y = 1$$
$$h_\theta(x) < 0.9 \rightarrow \text{predict } y = 0$$

So that $y = 1$ only if we are very confident. We have then a classifier with higher precision but lower recall. However, we could do the opposite and write our threshold to have less false negatives.

$$h_\theta(x) \geq 0.3 \rightarrow \text{predict } y = 1$$
$$h_\theta(x) < 0.3 \rightarrow \text{predict } y = 0$$

In this case we have higher recall, but lower precision.

It may be helpful to plot recall (x) against precision (y).

How do we judge a models precision and recall? We use an F_1-Score:

$$F_1 \text{ score } = 2\frac{PR}{P + R}$$

A perfect F_1-Score = 1. This value may be helpful in determining what threshold to use. We can maximize this value to pick our threshold.