Evaluating a Hypothesis

We can move on to evaluate our new hypothesis.

A hypothesis may have a low error for the training examples but still be inaccurate (because of overfitting). Thus, to evaluate a hypothesis, given a dataset of training examples, we can split up the data into two sets: a **training set** and a **test set**. Typically, the training set consists of 70 % of your data and the test set is the remaining 30 %.

## **The test set error**

A screenshot of a social media post

Description automatically generated

Model Selection and Train/Validation/Test Sets

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:

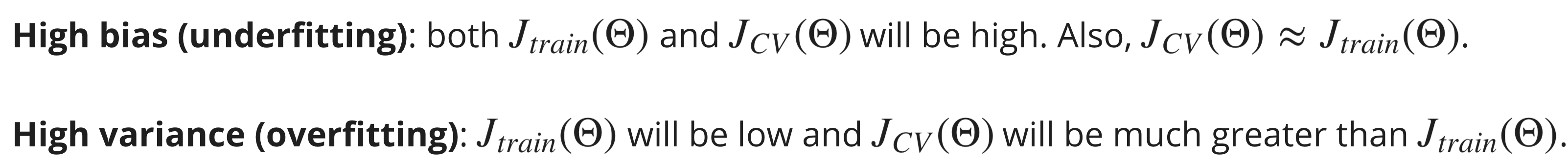
1. Optimize the parameters in Θ using the training set for each polynomial degree.
2. Find the polynomial degree d with the least error using the cross validation set.
3. Estimate the generalization error using the test set with Jtest(Θ(d)), (d = theta from polynomial with lower error);

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

Diagnosing Bias vs. Variance

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.



A close up of text on a white background

Description automatically generated

Regularization and Bias/Variance

**A close up of a map

Description automatically generated**

 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 *λ*s and for each *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.

Learning Curves

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

A screenshot of a cell phone

Description automatically generated

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

A screenshot of text

Description automatically generated

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.

**Model Complexity Effects:**

* Lower-order polynomials (low model complexity) have high bias and low variance. In this case, the model fits poorly consistently.
* Higher-order polynomials (high model complexity) fit the training data extremely well and the test data extremely poorly. These have low bias on the training data, but very high variance.
* In reality, we would want to choose a model somewhere in between, that can generalize well but also fits the data reasonably well.

Spam Classifier

So how could you spend your time to improve the accuracy of this classifier?

* Collect lots of data (for example "honeypot" project but doesn't always work)
* Develop sophisticated features (for example: using email header data in spam emails)
* Develop algorithms to process your input in different ways (recognizing misspellings in spam).

# Error Analysis

The recommended approach to solving machine learning problems is to:

* **Start with a simple algorithm, implement it quickly, and test it early on your cross validation data.**
* **Plot learning curves to decide if more data, more features, etc. are likely to help.**
* **Manually examine the errors on examples in the cross validation set and try to spot a trend where most of the errors were made.**

**A screenshot of a cell phone

Description automatically generated**

A screenshot of a cell phone

Description automatically generated