20 Bias and Variance: The two big sources of error

Suppose your training, dev and test sets all come from the same distribution. Then you should always try to get more training data, since that can only improve performance, right?

Even though having more data can't hurt, unfortunately it doesn't always help as much as you might hope. It could be a waste of time to work on getting more data. So, how do you decide when to add data, and when not to bother?

There are two major sources of error in machine learning: bias and variance. Understanding them will help you decide whether adding data, as well as other tactics to improve performance, are a good use of time.

Suppose you hope to build a cat recognizer that has 5% error. Right now, your training set has an error rate of 15%, and your dev set has an error rate of 16%. In this case, adding training data probably won't help much. You should focus on other changes. Indeed, adding more examples to your training set only makes it harder for your algorithm to do well on the training set. (We explain why in a later chapter.)

If your error rate on the training set is 15% (or 85% accuracy), but your target is 5% error (95% accuracy), then the first problem to solve is to improve your algorithm's performance on your training set. Your dev/test set performance is usually worse than your training set performance. So if you are getting 85% accuracy on the examples your algorithm has seen, there's no way you're getting 95% accuracy on examples your algorithm hasn't even seen.

Suppose as above that your algorithm has 16% error (84% accuracy) on the dev set. We break the 16% error into two components:

- First, the algorithm's error rate on the training set. In this example, it is 15%. We think of this informally as the algorithm's **bias**.
- Second, how much worse the algorithm does on the dev (or test) set than the training set. In this example, it does 1% worse on the dev set than the training set. We think of this informally as the algorithm's **variance**.[6]

[6] The field of statistics has more formal definitions of bias and variance that we won't worry about. Roughly, the bias is the error rate of your algorithm on your training set when you have a very large training set. The variance is how much worse you do on the test set compared to the training set in

Some changes to a learning algorithm can address the first component of error—**bias**—and improve its performance on the training set. Some changes address the second component—**variance**—and help it generalize better from the training set to the dev/test sets.[7] To select the most promising changes, it is incredibly useful to understand which of these two components of error is more pressing to address.

Developing good intuition about Bias and Variance will help you choose effective changes for your algorithm.

this setting. When your error metric is mean squared error, you can write down formulas specifying these two quantities, and prove that Total Error = Bias + Variance. But for our purposes of deciding how to make progress on an ML problem, the more informal definition of bias and variance given here will suffice.

[7] There are also some methods that can simultaneously reduce bias and variance, by making major changes to the system architecture. But these tend to be harder to identify and implement.

21 Examples of Bias and Variance

Consider our cat classification task. An "ideal" classifier (such as a human) might achieve nearly perfect performance in this task.

Suppose your algorithm performs as follows:

- Training error = 1%
- Dev error = 11%

What problem does it have? Applying the definitions from the previous chapter, we estimate the bias as 1%, and the variance as 10% (=11%-1%). Thus, it has **high variance**. The classifier has very low training error, but it is failing to generalize to the dev set. This is also called **overfitting**.

Now consider this:

- Training error = 15%
- Dev error = 16%

We estimate the bias as 15%, and variance as 1%. This classifier is fitting the training set poorly with 15% error, but its error on the dev set is barely higher than the training error. This classifier therefore has **high bias**, but low variance. We say that this algorithm is **underfitting**.

Now, consider this:

- Training error = 15%
- Dev error = 30%

We estimate the bias as 15%, and variance as 15%. This classifier has **high bias and high variance**: It is doing poorly on the training set, and therefore has high bias, and its performance on the dev set is even worse, so it also has high variance. The overfitting/underfitting terminology is hard to apply here since the classifier is simultaneously overfitting and underfitting.

Finally, consider this:

- Training error = 0.5%
- Dev error = 1%

This classifier is doing well, as it has low bias and low variance. Congratulations on achieving this great performance!

22 Comparing to the optimal error rate

In our cat recognition example, the "ideal" error rate—that is, one achievable by an "optimal" classifier—is nearly 0%. A human looking at a picture would be able to recognize if it contains a cat almost all the time; thus, we can hope for a machine that would do just as well.

Other problems are harder. For example, suppose that you are building a speech recognition system, and find that 14% of the audio clips have so much background noise or are so unintelligible that even a human cannot recognize what was said. In this case, even the most "optimal" speech recognition system might have error around 14%.

Suppose that on this speech recognition problem, your algorithm achieves:

- Training error = 15%
- Dev error = 30%

The training set performance is already close to the optimal error rate of 14%. Thus, there is not much room for improvement in terms of bias or in terms of training set performance. However, this algorithm is not generalizing well to the dev set; thus there is ample room for improvement in the errors due to variance.

This example is similar to the third example from the previous chapter, which also had a training error of 15% and dev error of 30%. If the optimal error rate is ~0%, then a training error of 15% leaves much room for improvement. This suggests bias-reducing changes might be fruitful. But if the optimal error rate is 14%, then the same training set performance tells us that there's little room for improvement in the classifier's bias.

For problems where the optimal error rate is far from zero, here's a more detailed breakdown of an algorithm's error. Continuing with our speech recognition example above, the total dev set error of 30% can be broken down as follows (a similar analysis can be applied to the test set error):

- **Optimal error rate ("unavoidable bias")**: 14%. Suppose we decide that, even with the best possible speech system in the world, we would still suffer 14% error. We can think of this as the "unavoidable" part of a learning algorithm's bias.

- **Avoidable bias**: 1%. This is calculated as the difference between the training error and the optimal error rate.[8]
- **Variance**: 15%. The difference between the dev error and the training error.

To relate this to our earlier definitions, Bias and Avoidable Bias are related as follows:[9]

Bias = Optimal error rate ("unavoidable bias") + Avoidable bias

The "avoidable bias" reflects how much worse your algorithm performs on the training set than the "optimal classifier."

The concept of variance remains the same as before. In theory, we can always reduce variance to nearly zero by training on a massive training set. Thus, all variance is "avoidable" with a sufficiently large dataset, so there is no such thing as "unavoidable variance."

Consider one more example, where the optimal error rate is 14%, and we have:

- Training error = 15%
- Dev error = 16%

Whereas in the previous chapter we called this a high bias classifier, now we would say that error from avoidable bias is 1%, and the error from variance is about 1%. Thus, the algorithm is already doing well, with little room for improvement. It is only 2% worse than the optimal error rate.

We see from these examples that knowing the optimal error rate is helpful for guiding our next steps. In statistics, the optimal error rate is also called **Bayes error rate**, or Bayes rate.

How do we know what the optimal error rate is? For tasks that humans are reasonably good at, such as recognizing pictures or transcribing audio clips, you can ask a human to provide labels then measure the accuracy of the human labels relative to your training set. This would give an estimate of the optimal error rate. If you are working on a problem that even

[8] If this number is negative, you are doing better on the training set than the optimal error rate. This means you are overfitting on the training set, and the algorithm has over-memorized the training set. You should focus on variance reduction methods rather than on further bias reduction methods.

[9] These definitions are chosen to convey insight on how to improve your learning algorithm. These definitions are different than how statisticians define Bias and Variance. Technically, what I define here as "Bias" should be called "Error we attribute to bias"; and "Avoidable bias" should be "error we attribute to the learning algorithm's bias that is over the optimal error rate."

humans have a hard time solving (e.g., predicting what movie to recommend, or what ad to show to a user) it can be hard to estimate the optimal error rate.

In the section "Comparing to Human-Level Performance (Chapters 33 to 35), I will discuss in more detail the process of comparing a learning algorithm's performance to human-level performance.

In the last few chapters, you learned how to estimate avoidable/unavoidable bias and variance by looking at training and dev set error rates. The next chapter will discuss how you can use insights from such an analysis to prioritize techniques that reduce bias vs. techniques that reduce variance. There are very different techniques that you should apply depending on whether your project's current problem is high (avoidable) bias or high variance. Read on!

23 Addressing Bias and Variance

Here is the simplest formula for addressing bias and variance issues:

- If you have high avoidable bias, increase the size of your model (for example, increase the size of your neural network by adding layers/neurons).
- If you have high variance, add data to your training set.

If you are able to increase the neural network size and increase training data without limit, it is possible to do very well on many learning problems.

In practice, increasing the size of your model will eventually cause you to run into computational problems because training very large models is slow. You might also exhaust your ability to acquire more training data. (Even on the internet, there is only a finite number of cat pictures!)

Different model architectures—for example, different neural network architectures—will have different amounts of bias/variance for your problem. A lot of recent deep learning research has developed many innovative model architectures. So if you are using neural networks, the academic literature can be a great source of inspiration. There are also many great open-source implementations on github. But the results of trying new architectures are less predictable than the simple formula of increasing the model size and adding data.

Increasing the model size generally reduces bias, but it might also increase variance and the risk of overfitting. However, this overfitting problem usually arises only when you are not using regularization. If you include a well-designed regularization method, then you can usually safely increase the size of the model without increasing overfitting.

Suppose you are applying deep learning, with L2 regularization or dropout, with the regularization parameter that performs best on the dev set. If you increase the model size, usually your performance will stay the same or improve; it is unlikely to worsen significantly. The only reason to avoid using a bigger model is the increased computational cost.

24 Bias vs. Variance tradeoff

You might have heard of the "Bias vs. Variance tradeoff." Of the changes you could make to most learning algorithms, there are some that reduce bias errors but at the cost of increasing variance, and vice versa. This creates a "trade off" between bias and variance.

For example, increasing the size of your model—adding neurons/layers in a neural network, or adding input features—generally reduces bias but could increase variance. Alternatively, adding regularization generally increases bias but reduces variance.

In the modern era, we often have access to plentiful data and can use very large neural networks (deep learning). Therefore, there is less of a tradeoff, and there are now more options for reducing bias without hurting variance, and vice versa.

For example, you can usually increase a neural network size and tune the regularization method to reduce bias without noticeably increasing variance. By adding training data, you can also usually reduce variance without affecting bias.

If you select a model architecture that is well suited for your task, you might also reduce bias and variance simultaneously. Selecting such an architecture can be difficult.

In the next few chapters, we discuss additional specific techniques for addressing bias and variance.

25 Techniques for reducing avoidable bias

If your learning algorithm suffers from high avoidable bias, you might try the following techniques:

- **Increase the model size** (such as number of neurons/layers): This technique reduces bias, since it should allow you to fit the training set better. If you find that this increases variance, then use regularization, which will usually eliminate the increase in variance.
- **Modify input features based on insights from error analysis**: Say your error analysis inspires you to create additional features that help the algorithm eliminate a particular category of errors. (We discuss this further in the next chapter.) These new features could help with both bias and variance. In theory, adding more features could increase the variance; but if you find this to be the case, then use regularization, which will usually eliminate the increase in variance.
- **Reduce or eliminate regularization** (L2 regularization, L1 regularization, dropout): This will reduce avoidable bias, but increase variance.
- **Modify model architecture** (such as neural network architecture) so that it is more suitable for your problem: This technique can affect both bias and variance.

One method that is not helpful:

- **Add more training data**: This technique helps with variance problems, but it usually has no significant effect on bias.

26 Error analysis on the training set

Your algorithm must perform well on the training set before you can expect it to perform well on the dev/test sets.

In addition to the techniques described earlier to address high bias, I sometimes also carry out an error analysis on the *training data*, following a protocol similar to error analysis on the Eyeball dev set. This can be useful if your algorithm has high bias—i.e., if it is not fitting the training set well.

For example, suppose you are building a speech recognition system for an app and have collected a training set of audio clips from volunteers. If your system is not doing well on the training set, you might consider listening to a set of ~100 examples that the algorithm is doing poorly on to understand the major categories of training set errors. Similar to the dev set error analysis, you can count the errors in different categories:

Audio clip	Loud background noise	User spoke quickly	Far from microphone	Comments
1	✔			Car noise
2	✔		✔	Restaurant noise
3		✔	✔	User shouting across living room?
4	✔			Coffeeshop
% of total	75%	25%	50%	

In this example, you might realize that your algorithm is having a particularly hard time with training examples that have a lot of background noise. Thus, you might focus on techniques that allow it to better fit training examples with background noise.

You might also double-check whether it is possible for a person to transcribe these audio clips, given the same input audio as your learning algorithm. If there is so much background noise that it is simply impossible for anyone to make out what was said, then it might be unreasonable to expect any algorithm to correctly recognize such utterances. We will discuss the benefits of comparing your algorithm to human-level performance in a later section.

27 Techniques for reducing variance

If your learning algorithm suffers from high variance, you might try the following techniques:

- **Add more training data**: This is the simplest and most reliable way to address variance, so long as you have access to significantly more data and enough computational power to process the data.
- **Add regularization** (L2 regularization, L1 regularization, dropout): This technique reduces variance but increases bias.
- **Add early stopping** (i.e., stop gradient descent early, based on dev set error): This technique reduces variance but increases bias. Early stopping behaves a lot like regularization methods, and some authors call it a regularization technique.
- **Feature selection to decrease number/type of input features:** This technique might help with variance problems, but it might also increase bias. Reducing the number of features slightly (say going from 1,000 features to 900) is unlikely to have a huge effect on bias. Reducing it significantly (say going from 1,000 features to 100—a 10x reduction) is more likely to have a significant effect, so long as you are not excluding too many useful features. In modern deep learning, when data is plentiful, there has been a shift away from feature selection, and we are now more likely to give all the features we have to the algorithm and let the algorithm sort out which ones to use based on the data. But when your training set is small, feature selection can be very useful.
- **Decrease the model size** (such as number of neurons/layers): *Use with caution*. This technique could decrease variance, while possibly increasing bias. However, I don't recommend this technique for addressing variance. Adding regularization usually gives better classification performance. The advantage of reducing the model size is reducing your computational cost and thus speeding up how quickly you can train models. If speeding up model training is useful, then by all means consider decreasing the model size. But if your goal is to reduce variance, and you are not concerned about the computational cost, consider adding regularization instead.

Here are two additional tactics, repeated from the previous chapter on addressing bias:

- **Modify input features based on insights from error analysis**: Say your error analysis inspires you to create additional features that help the algorithm to eliminate a particular category of errors. These new features could help with both bias and variance. In

theory, adding more features could increase the variance; but if you find this to be the case, then use regularization, which will usually eliminate the increase in variance.

- **Modify model architecture** (such as neural network architecture) so that it is more suitable for your problem: This technique can affect both bias and variance.