Error decomposition, bias-variance tradeoff

Fraida Fund

Contents

Math prerequisites for this lecture: You should know about these concepts from probability:

- Expectation of a random variable, including linearity of expectation, expectation of a constant
- Variance of a random variable
- Independence of two random variables

In this lecture

- Prediction error
- Error decomposition
- Bias variance tradeoff

When learning fails

Consider the following scenario: You were given a learning task and have approached it with a choice of model, a training algorithm, and data. You used some of the data to fit the parameters and tested the fitted model on a test set. The test results, unfortunately, turn out to be unsatisfactory.

What went wrong then, and what should you do next?

There are many elements that can be "fixed." The main approaches are listed as follows:

- Fix a data problem
- Get more data
- Change the model (i.e. the function that maps input data to target variable) by:
  - Making it more flexible
  - Making it less flexible
  - Completely changing its form
- Change the feature representation of the data
- Change the training algorithm used to fit the model

In order to find the best remedy, it is essential first to understand the cause of the bad performance.

Note: this scenario is closely paraphrased from *Section 11.3 What to Do If Learning Fails* in Understanding Machine Learning: From Theory to Algorithms (Shalev-Shwartz and Ben-David).

Prediction error

ML premise: fit a function

Given as *training* data a set of feature vector-label pairs

$$(\mathbf{x_i}, y_i), \quad i = 1, \ldots n$$

we want to fit a function f (parameterized by $\mathbf{w}$, which we'll estimate as $\hat{\mathbf{w}}$) such that

$$y_i \approx f(\mathbf{x_i}, \mathbf{w})$$

(i.e. $y_i \approx \hat{y}_i$.)

ML premise: true function

Suppose our data is sampled from some *unknown* "true" function t so that

$$y_i = t(\mathbf{x_i}) + \epsilon_i$$

where $\epsilon \sim N(0, \sigma_\epsilon^2)$ is some *unknowable* stochastic error.

ML premise: minimize error

Our goal is to minimize a squared error loss function on some *test* point, $(\mathbf{x_t}, y_t)$:

$$E[(y_t - \hat{y}_t)^2]$$

where the expectation is over the sampled data, and the noise.

ML premise: illustration

Figure 1: Imagine an infinite population of data, from which we sample training data and a test point.

Source of prediction error: noise

In the best case scenario, even if the model is exactly the true function

$$f(\mathbf{x}, \mathbf{w}) = t(\mathbf{x}) \quad \forall x$$

we still have some prediction error due to the stochastic noise!

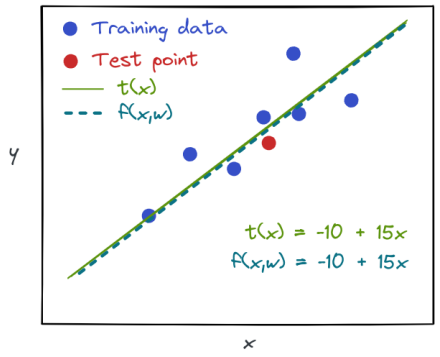

Figure 2: In the best case scenario, we still expect some error due to noise.

Source of prediction error: parameter estimate

Perhaps the true function is

$$t(\mathbf{x}) = f(\mathbf{x}, \mathbf{w_t}) \quad \forall x$$

but because of the random sample of training data + noise in the data, our parameter estimate is not exactly correct: $\hat{\mathbf{w}} \neq \mathbf{w_t}$.

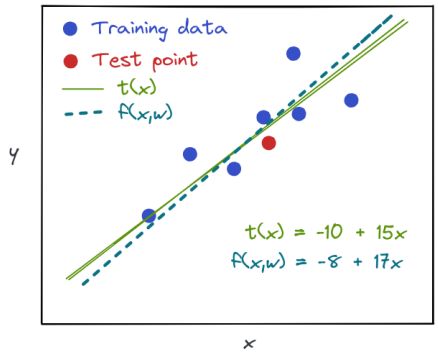

Figure 3: We may have an error in our parameter estimate (due to sample of training data + noise).

Source of prediction error: assumed model class (1)

Maybe

$$t(\mathbf{x}) \neq f(\mathbf{x}, \mathbf{w})$$

for any $\mathbf{w}$!

Our assumed *model class* (or *hypothesis class*) may not be complex enough to model the true function.

Note: the *model class* is the set of possible models we could fit, parameterized by the parameter vector.

Note: the set of assumptions we make - such as selecting a model class f - introduce what's known as *inductive bias* into our model.

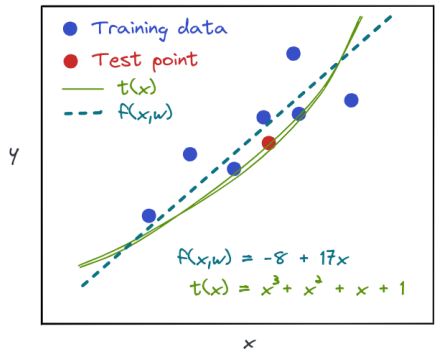

Figure 4: Our model class is not flexible enough.

Source of prediction error: assumed model class (2)

What if we use a model class that is *too* complex?

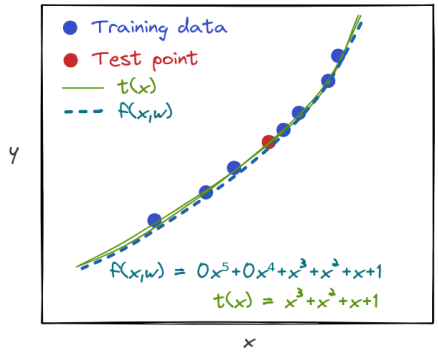

Figure 5: If there was no noise, a too-complex model class wouldn't necessarily be a problem.

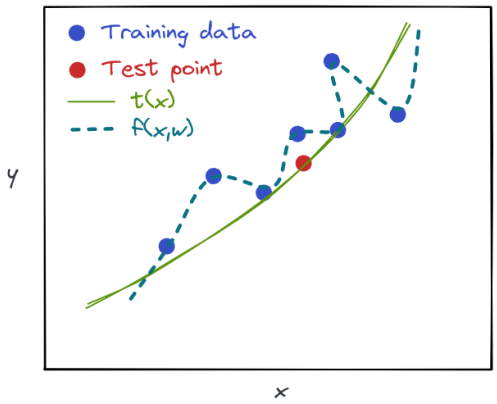

Figure 6: But the combination of too-complex model + noise in training data *is* a problem! The too-complex model "overfits" to the unknowable stochastic noise in the *training* data - which will increase expected error on the *test* data.

This is not specific to polynomial models - there are many situations where a training algorithm will "learn" a parameter value that should be zero, because it fits the noise. For example, if you have irrelevant features used as input to the model.

Sources of prediction error: summary

- Stochastic noise which is fundamentally unpredictable
- Parameter estimate has some error due to noise in training data
- Assumed model class is not complex enough (**under-modeling**)

Note: the "parameter estimate" error also includes overfitting!

Error decomposition

A note on this decomposition

We will derive the expected error on the test *point*:

- first, assuming the training sample is fixed, so the expectation is only over ϵ_t
- then, relaxing this assumption, so the expectation is over the training set sample $\mathcal{D}$

This is allowed because of independence of ϵ_t and $\mathcal{D}$; so

$$E_{\mathcal{D},\epsilon}[...] = E_{\mathcal{D}}[E_{\epsilon}[...]]$$

Finally, we'll take that expectation over all the test points.

First: assuming fixed training sample

For convenience, denote $f(\mathbf{x_t}, \hat{\mathbf{w}})$ as f and $t(\mathbf{x_t})$ as t.

$$\begin{aligned} E_{\epsilon}[(y_t - \hat{y_t})^2] &= E_{\epsilon}[(t + \epsilon_t - f)^2] \\ &= E_{\epsilon}[(t-f)^2 + \epsilon_t^2 + 2\epsilon_t(t-f)] \\ &= (t-f)^2 + E_{\epsilon}[\epsilon_t^2] + 0 \\ &= (t-f)^2 + \sigma_{\epsilon}^2 \end{aligned}$$

The expected value (over the ϵ) of squared error is because:

- under the assumption that training sample is fixed, f and t are constant
- $E[\epsilon_t] = 0$

The last term is not affected when we then take the expectation over $\mathcal{D}$, either. This term is called the *irreducible error*, and it not under our control.

The first term ($(t-f)^2$) is the model estimation error, and this *is* under our control - it is *reducible* error - so next we will turn to $E_{\mathcal{D}}[(t-f)^2]$.

Second: expectation over $\mathcal{D}$

We again denote $f(\mathbf{x_t}, \hat{\mathbf{w}})$ as f and $t(\mathbf{x_t})$ as t.

$$\begin{aligned} E_{\mathcal{D}}[(t-f)^2 + \sigma_{\epsilon}^2] &= E_{\mathcal{D}}[(t-f)^2] + \sigma_{\epsilon}^2 \\ &= E_{\mathcal{D}}[t^2 + f^2 - 2tf] + \sigma_{\epsilon}^2 \\ &= t^2 + E_{\mathcal{D}}[f^2] - 2tE_{\mathcal{D}}[f] + \sigma_{\epsilon}^2 \\ &= (t - E_{\mathcal{D}}[f])^2 + (E_{\mathcal{D}}[f^2] - E_{\mathcal{D}}[f]^2) + \sigma_{\epsilon}^2 \end{aligned}$$

because:

- the true value $t(\mathbf{x_t})$ is independent of the training sample drawn from $\mathcal{D}$.

A hypothetical (impossible) experiment

To understand this decomposition, it helps to think about this experiment.

Suppose we would get many independent training sets (from same process).

For each training set,

- train our model (estimate parameters), and
- use this model to estimate value of test point(s)

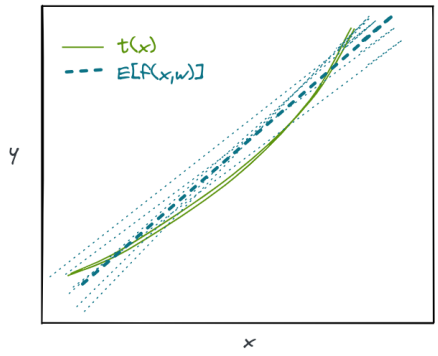

Figure 7: Hypothetical experiment, showing many trained models, and the mean of all those trained models.

Error decomposition: bias

In the first term in

$$(t - E_{\mathcal{D}}[f])^2 + (E_{\mathcal{D}}[f^2] - E_{\mathcal{D}}[f]^2) + \sigma_\epsilon^2$$

$t - E_{\mathcal{D}}[f]$ is called the **bias**.

The bias is the difference between the *true value* and the *mean prediction of the model* (over many different random samples of training data.)

Informally: it tells us to what extent the model is *systematically* wrong!

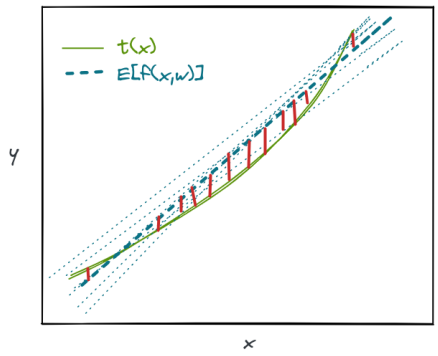

Figure 8: The bias term is the difference between the mean model and true function.

Error decomposition: variance

The second term in

$$(t - E_{\mathcal{D}}[f])^2 + (E_{\mathcal{D}}[f^2] - E_{\mathcal{D}}[f]^2) + \sigma_\epsilon^2$$

$E_{\mathcal{D}}[f^2] - E_{\mathcal{D}}[f]^2$ is the **variance** of the model prediction over $\mathcal{D}$.

Informally: it tells us: if you train many of these models, with a new sample of training data each time, how much variation is there in the model output?

Or: how much does the model output depend on the training data?

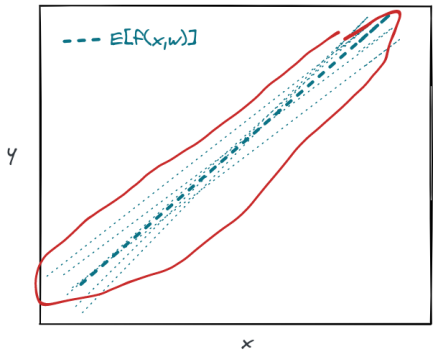

Figure 9: The variance term is the difference between the mean model and the individual models.

Error decomposition: irreducible error

We already said that the third term in

$$(t - E_{\mathcal{D}}[f])^2 + (E_{\mathcal{D}}[f^2] - E_{\mathcal{D}}[f]^2) + \sigma_\epsilon^2$$

is called the **irreducible errror**.

This term is a *lower bound* on the MSE.

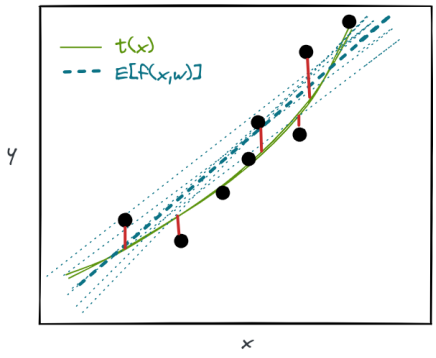

Figure 10: The irreducible error is the difference between data points and the output of the true function.

Error decomposition: summary

Putting it together, the expected test point error

$$(t - E_{\mathcal{D}}[f])^2 + (E_{\mathcal{D}}[f^2] - E_{\mathcal{D}}[f]^2) + \sigma_\epsilon^2$$

is

$$(\text{Bias})^2 + \text{Variance over } \mathcal{D} + \text{Irreducible Error}$$

Bias-variance tradeoff

Intuition behind bias-variance and model complexity

It's often the case that changing the model to reduce bias, increases variance (and vice versa). Why?

Bias variance tradeoff

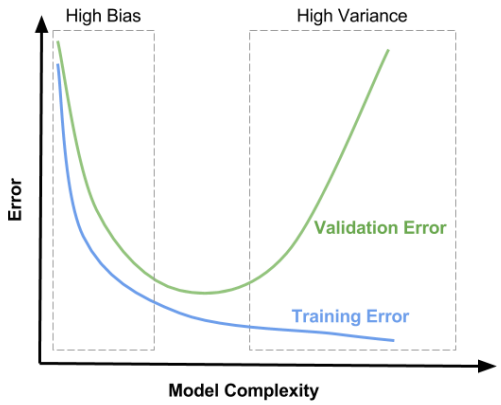

Figure 11: Bias variance tradeoff

Note: this is a "classic" view of the bias-variance tradeoff. Recent results suggest that this is only part of the picture.

Updated view: double descent

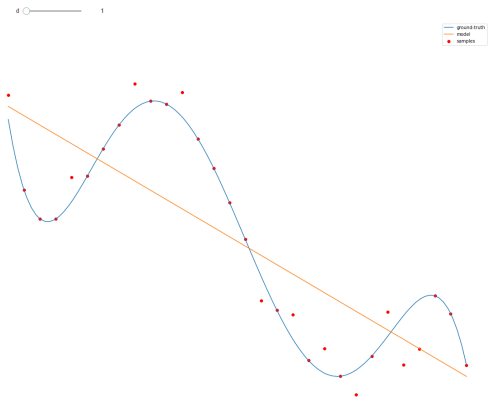

Figure 12: Polynomial model before and after the interpolation threshold. Image source: Boaz Barak, click link to see animation.

Explanation (via Boaz Barak):

> When d of the model is less than d_t of the polynomial, we are "under-fitting" and will not get good performance. As d increases between d_t and n, we fit more and more of the noise, until for $d = n$ we have a perfect interpolating polynomial that will have perfect training but very poor test performance. When d grows beyond n, more than one polynomial can fit the data, and (under certain conditions) SGD will select the minimal norm one, which will make the interpolation smoother and smoother and actually result in better performance.

For an intuitive explanation of "double descent", see:

- Double Descent (Jared Wilber, Brent Werness)
- Double Descent 2 (Brent Werness, Jared Wilber)

Recap

- Decomposition of model error into different parts
- Intuition about model complexity vs model error
- Next: how to choose "good" models that balance the tradeoff?