Machine Learning Course - CS-433

Bias-Variance Decomposition

Oct 13, 2016

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Motivation

In the last lecture we have discussed how we can choose good hyper-parameters, i.e., how we can decide which of various models we are considering is best. The basic idea was to split the data and to use one part to *train* and the other to *validate*.
Today we will talk about an inherent *bias* - *variance* trade-off that we are facing when we perform the model selection and are trying to decide how "complex" or 'rich" we should make our model.
The idea is very simple. To be concrete, consider a regression problem with a one-dimensional input and a polynomial basis as features, but the same principle applies whenever we have a parameter that regulates the complexity of the model (e.g., like in the ridge regression problem). Assume that the maximum degree of the polynomial we allow is d. We are trying to decide how large to pick d.
As we have discussed, when d is very large (think $d \to \infty$) we are considering a very rich model. But if d is very small (think $d = 0, 1$) then the model is very constrained. In both these extreme cases we are likely going to see a poor performance of our prediction, but for very different reasons. If d is very small (think $d = 0, 1$) then it is likely that we cannot find a good prediction function within our model to fit the data. We are saying that we are having a large *bias* (bad fit). But since we are only considering very simple models (e.g., a constant, or affine function) even with relatively little data S, we are likely to always *learn* a very similar prediction function f_S and so the associated loss will not vary widely.

Hence the variance of $L_{\mathcal{D}}(f_S)$ (the variations due to the finite data S) is small. In summary: for very simple models we expect a large bias but a small variance.
Consider now the other extreme of a very large d (very complex model). In this case there are likely functions contained in the model that can model the data very well and so we expect a smaller bias. But we are in danger of overfitting the data and even changing a single sample from S_t might change the resulting function considerably. We therefore expect to see a high variance of $L_{\mathcal{D}}(f_S)$ as a function of S_t. In summary: for complex models we expect small bias but high variance.

Data Generation Model

Let us now make the above intuition and observation more precise. Assume that the data is generated in the following way. Let

$$y = f(\mathbf{x}) + \epsilon, \tag{1}$$

where f is some (arbitrarily complex and unknown) function and ϵ is additive *noise* which is independent from sample to sample and independent from the data. We also assume that the noise has zero mean (otherwise this constant can be included into f). Note that the function f is in general not in our model class.
We further assume that $\mathbf{x}$ is generated according to some fixed but unknown distribution $\mathcal{D}_{\mathbf{X}}$ and we let $\mathcal{D}$ be the *induced* distribution on pairs $(y, \mathbf{x})$. Finally, we assume that the loss function $\ell(\cdot, \cdot)$ is the square loss.

Error Decomposition

As always, we have given some training data S_t, consisting of iid samples according to $\mathcal{D}$. Given our learning algorithm $\mathcal{A}$ we compute the prediction function $f_{S_t} = \mathcal{A}(S_t)$. We look at the square loss of this prediction function for a fixed element $\mathbf{x}_0$, i.e., we compute

$$(f(\mathbf{x}_0) + \epsilon - f_{S_t}(\mathbf{x}_0))^2,$$

where we used our specific data generation model.
We imagine that we are running the experiment many times: create S_t, learn the model f_{S_t}, and evaluate the performance by computing the square loss for this fixed element $\mathbf{x}_0$.
So let us look at the expected value of this quantity:

$$\mathbb{E}_{S_t \sim \mathcal{D}, \text{noise}}[(f(\mathbf{x}_0) + \epsilon - f_{S_t}(\mathbf{x}_0))^2].$$

It turns out that we can rewrite the above quantity as a sum of three non-negative terms and this decomposition has a nice interpretation. Let us frist do the rewriting. We will look at the interpretation shortly.

Even with the best ML algo imaginable, there is some bounds to the prediction we can make.

$$\begin{aligned}
&\mathbb{E}_{S_t\sim\mathcal{D},\text{noise}}[(f(\mathbf{x}_0)+\epsilon-f_{S_t}(\mathbf{x}_0))^2] \\
\overset{(a)}{=}\ &\mathbb{E}[\epsilon^2]+\mathbb{E}_{S_t\sim\mathcal{D},\text{noise}}[(f(\mathbf{x})-f_{S_t}(\mathbf{x}))^2] \\
\overset{(b)}{=}\ &\text{Var}[\epsilon]+\mathbb{E}_{S_t\sim\mathcal{D}}[(f(\mathbf{x}_0)-f_{S_t}(\mathbf{x}_0))^2] \\
\overset{(c)}{=}\ &\underbrace{\text{Var}[\epsilon]}_{\text{noise}}+ \\
&\underbrace{(f(\mathbf{x}_0)-\mathbb{E}_{S_t\sim\mathcal{D}}[f_{S_t}(\mathbf{x}_0)])^2}_{\text{bias}}+ \\
&\mathbb{E}_{S_t\sim\mathcal{D}}[\underbrace{(\mathbb{E}_{S_t\sim\mathcal{D}}[f_{S_t}(\mathbf{x}_0)]-f_{S_t}(\mathbf{x}_0))^2}_{\text{variance}}].
\end{aligned}$$

Go down with more complexe model (but wrong)

In step (a), besides the two terms that we have written down we also have the term

$$\mathbb{E}_{S_t\sim\mathcal{D},\text{noise}}[2\epsilon(f(\mathbf{x}_0)-f_{S_t}(\mathbf{x}_0))^2].$$

But since the noise is independent from S_t by assumption we can first average over the noise, and by observing that the noise has mean zero, we see that this term is in fact zero. Further, since the noise has zero mean, the second moment is equal to the variance. This explains step (b).
In step (c) we have added and substracted the constant term $\mathbb{E}_{S_t\sim\mathcal{D}}[f_{S_t}(\mathbf{x}_0)]$ to the expression and then expanded the square. This yields the two expression which are stated (termed "bias" and "variance") as well as the cross term

$$\begin{aligned}
&\mathbb{E}_{S_t\sim\mathcal{D}}[(f(\mathbf{x}_0)-\mathbb{E}_{S_t\sim\mathcal{D}}[f_{S_t}(\mathbf{x}_0)])(\mathbb{E}_{S_t\sim\mathcal{D}}[f_{S_t}(\mathbf{x}_0)]-f_{S_t}(\mathbf{x}_0))] \\
=&(f(\mathbf{x}_0)-\mathbb{E}_{S_t\sim\mathcal{D}}[f_{S_t}(\mathbf{x}_0)])\mathbb{E}_{S_t\sim\mathcal{D}}[(\mathbb{E}_{S_t\sim\mathcal{D}}[f_{S_t}(\mathbf{x}_0)]-f_{S_t}(\mathbf{x}_0))] \\
=&0.
\end{aligned}$$

Interpretation of Decomposition

Let us now interpret this decomposition. We first note that each of the three terms is non-negative. Hence each of them is a lower bound on the expected loss when we predict the value for the input $\mathbf{x}_0$.

In particular, if the data contains noise, as we have assumed, then this noise imposes a strict lower bound on what error we can achieve. This is given by the term $\mathrm{Var}[\epsilon]$.

Var(e)
(noise)

Next, consider the "bias term." As we discussed, if we only allow simple models then we will not be able to find a good fit for the value $f(\mathbf{x}_0)$ in average. Hence, the bias will be large. This bias adds to the error that we observe.

Bias
Bias
Bias

Finally we have the "variance" term. If we consider very complicated models then small variations in the data set can produce vastly different models and our prediction of the value associated to some input $\mathbf{x}_0$ will vary widely. Also this variance term adds to error we will have.

Examples

The following four figures are take from the book by James, Witten, Hastie, and Tibshiranie (Introduction to Statistical Learning).

The first three pictures show three different functions each (the true function is the black curve). The first function has medium "complexity", the second is very simple, and the third is the most complicated. In each case, three different predictions are done based on models of increasing complexity.

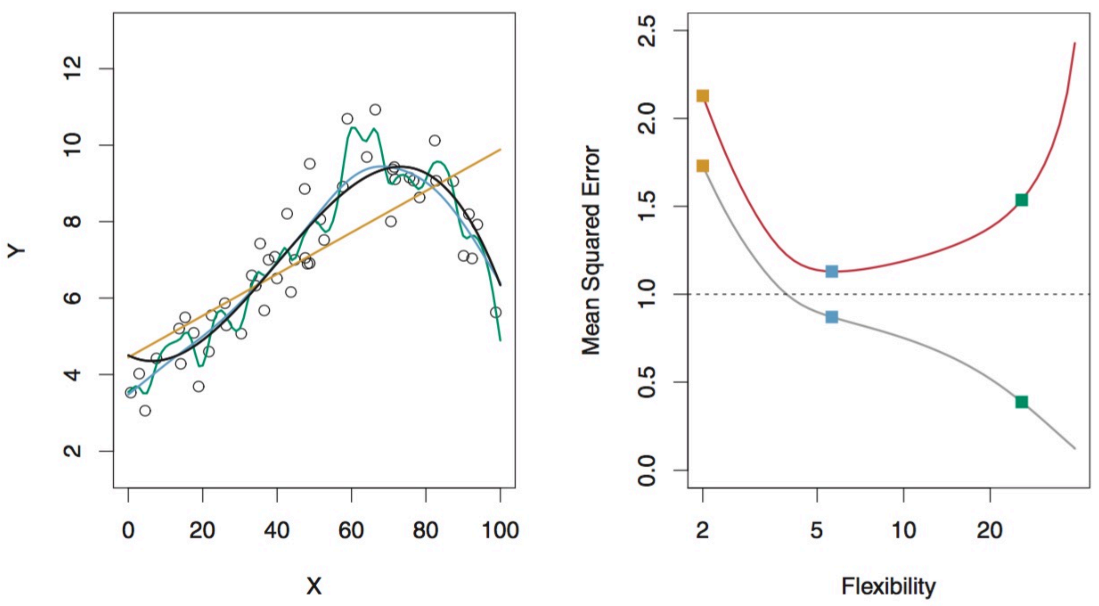

FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

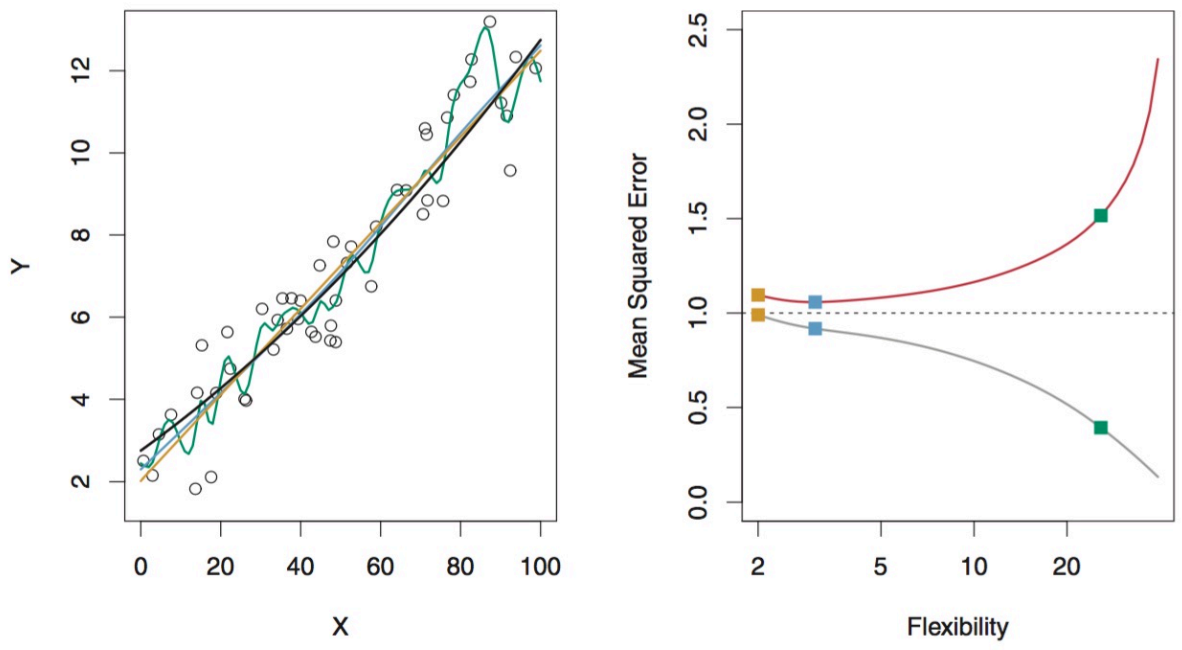

FIGURE 2.10. *Details are as in Figure 2.9, using a different true f that is much closer to linear. In this setting, linear regression provides a very good fit to the data.*

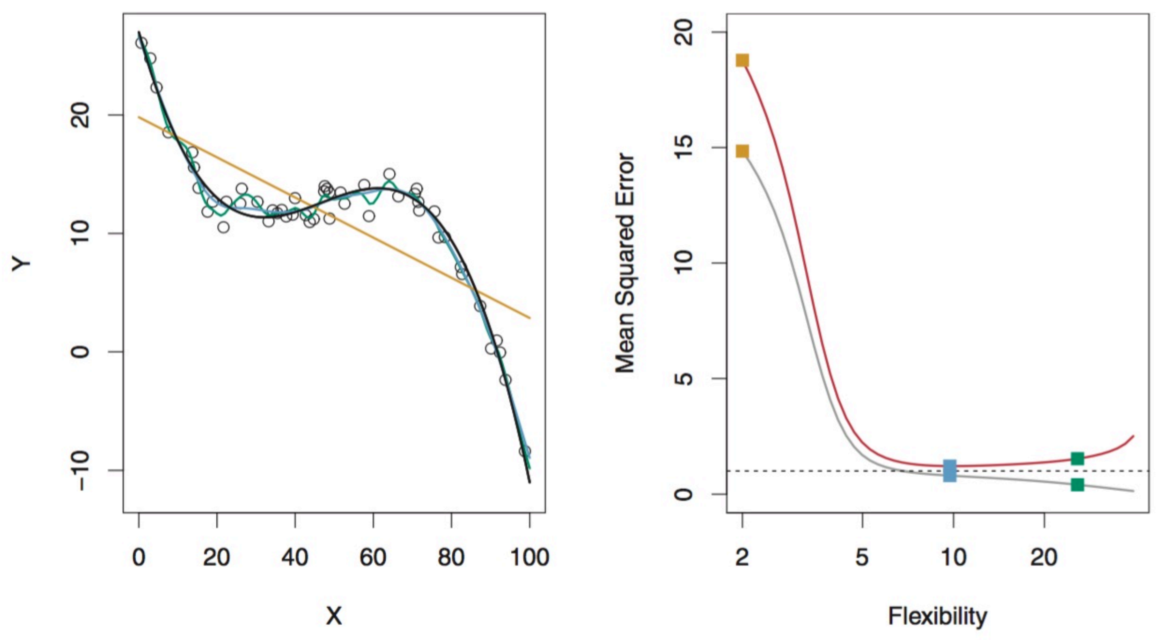

FIGURE 2.11. *Details are as in Figure 2.9, using a different f that is far from linear. In this setting, linear regression provides a very poor fit to the data.*

The final figure shows the bias-variance decomposition for each of these three models as a function of increasing complexity.

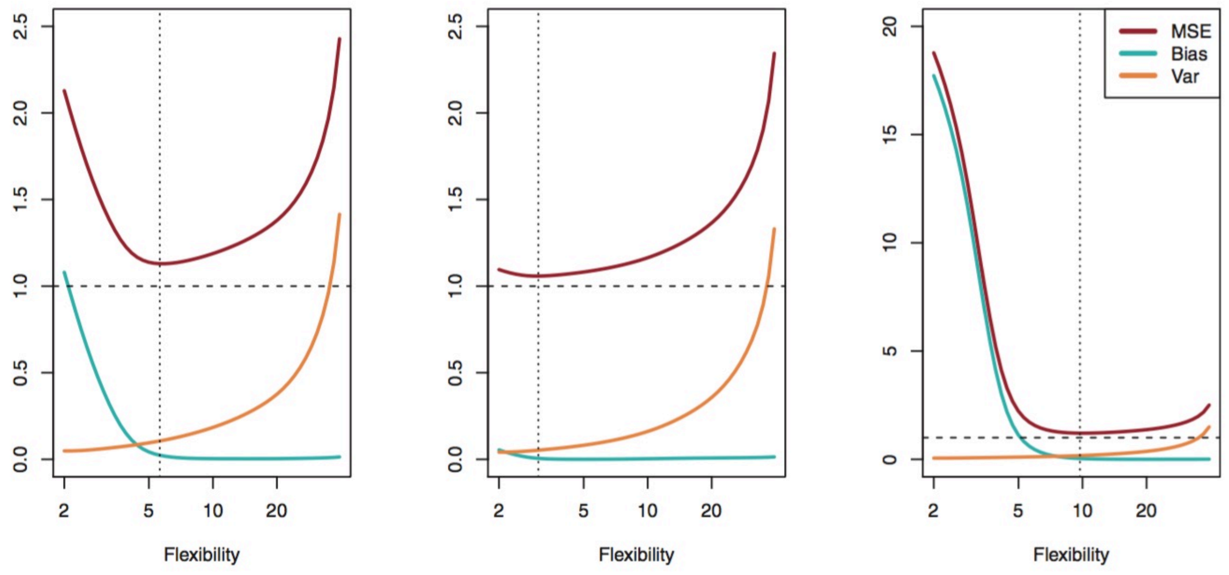

FIGURE 2.12. *Squared bias (blue curve), variance (orange curve), Var(ϵ) (dashed line), and test MSE (red curve) for the three data sets in Figures 2.9–2.11. The vertical dotted line indicates the flexibility level corresponding to the smallest test MSE.*

Additional Notes

Cross validation and generalization error

Cross validation estimates the expected train and test error. If the learning curve is steep, then cross validation overestimates the true objective function. Please read HTF Section 7.10.

Testing regression methods

You can learn about the following from HTF Page 47-49 and Chapter 3 of JWHT (see course infromation for book details).

- R^2 and RMSE goodness of fit.
- Significance and hypothesis testing.
- Confidence interval, standard error, p-value, t-statistics etc.
- Feature engineering: transformations of input variables, adding interactions, dummy encoding of binary and categorical variables, missing values.

ToDo

- Clearly understand the definition of expected test and train errors (do the exercise).
- Read HTF Section 7.2 and 7.3. This may not be an easy read.
- Revise the derivation of bias-variance decomposition.
- Visualize bias-variance decomposition during labs.
- For testing regression methods, read Page 47-49 from HTF and Chapter 3 of JWHT (see course infromation for book details).