Machine Learning I

Lecture 5: The Bias-Complexity Tradeoff

Nathaniel Bade

Northeastern University Department of Mathematics

Table of contents

The Bias-Complexity Tradeoff

Avoiding Underfitting

The no free lunch theorem isn't the end of machine learning, it simply asserts that there is no universally best learner for every task. If fact, it implies that we should use any prior knowledge to avoid learners that perform poorly on a distribution. Such prior knowledge can be expressed by restricting the hypothesis class.

But how do we choose this class? On one hand, we want a class that contains a classifier that will return the minimum error. On the other hand, the class of all functions is clearly not learnable so we cant just choose the richest class.

Bias and Complexity

To begin to address the question, we decompose the total error

$$L_{\mathcal{D}}(h_{\mathcal{T}}) = \underbrace{\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)}_{\epsilon_{app}} + \underbrace{\left(L_{\mathcal{D}}(h_{\mathcal{T}}) - \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)\right)}_{\epsilon_{est}}$$

into

Approximation Error: The minimum risk achievable within a class. This term is also known as the **inductive bias**.

Estimation Error: The difference between the minimum achievable risk and the ERM error. This is the term to be minimized in APAC.

Again: **APAC learning bounds estimation error.** It fundamentally says nothing about approximation error.

Estimation Error

Last lecture we derived an expression for the estimation error of a finite hypothesis class $\mathcal{H}$. We found,

$$\epsilon_{est} = L_{\mathcal{D}}(h_{\mathcal{T}}) - \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \geq \left(\frac{\log(2|\mathcal{H}|/\delta)}{2N} \right)^{\frac{1}{2}} .$$

For fixed δ, if the training size N decreases or $|\mathcal{H}|$ increases (exponentially), the **estimation error** increases. This justifies thinking of ϵ_{est} as a measure of relative complexity.

Bias and Complexity

The decomposition

$$L_{\mathcal{D}}(h_{\mathcal{T}}) = \underbrace{\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)}_{\epsilon_{app}} + \underbrace{\left(L_{\mathcal{D}}(h_{\mathcal{T}}) - \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)\right)}_{\epsilon_{est}}$$

is known as the **bias-complexity trade off**. It states that within a hypothesis class we can trade bias for complexity, but we cannot minimize both.

Note that this decomposition is quite general: it holds regardless of the loss function, the hypothesis class or the learning algorithm. In the rest of this lecture, we will derive more specific results by analyzing the decomposition for our test hypothesis classes, the linear predictors and k nearest neighbors.

The Bias-Variance Tradeoff: Bias-Complexity for RSS

Estimation Error

There is more we can say if we restrict ourselves to a specific loss function and distribution. Assume for the moment that there is a true labeling function $f : \mathcal{X} \to \mathcal{Y}$. Consider the pointwise **mean squared error** of a predictor $h_{\mathcal{T}}$ on some $x_0 \in \mathcal{X}$.

$$MSE(x_0) = E_{\mathcal{T}}[f(x_0) - h_{\mathcal{T}}(x_0)]^2$$

Define $y_0^{\mathcal{H}} := E_{\mathcal{T}}[h_{\mathcal{T}}(x_0)]$ to be the expected label of x_0 over the training sets. Then

$$\begin{aligned} MSE(x_0) &= E_{\mathcal{T}}\Big[\,\big(f(x_0) - y_0^{\mathcal{H}}\,\big) + \big(\,y_0^{\mathcal{H}} - h_{\mathcal{T}}(x_0)\,\big)\,\Big]^2 \\ &= E_{\mathcal{T}}\big[\,f(x_0) - y_0^{\mathcal{H}}\,\big]^2 + E_{\mathcal{T}}\big[\,y_0^{\mathcal{H}} - h_{\mathcal{T}}(x_0)\,\big]^2 \\ &\qquad + 2E_{\mathcal{T}}\big[\,\big(f(x_0) - y_0^{\mathcal{H}}\,\big)\big(y_0^{\mathcal{H}} - h_{\mathcal{T}}(x_0)\big)\big] \end{aligned}$$

The final term cancels since $E_{\mathcal{T}}[h_{\mathcal{T}}(x_0)] = y_0^{\mathcal{H}}$.

Estimation Error

There is more we can say if we restrict ourselves to a specific loss functions and distributions. Assume for the moment that there is a true labeling function $f : \mathcal{X} \to \mathcal{Y}$. Consider the pointwise **mean squared error** of a predictor $h_{\mathcal{T}}$ on some $x_0 \in \mathcal{X}$.

$$MSE(x_0) = E_{\mathcal{T}}[f(x_0) - h_{\mathcal{T}}(x_0)]^2$$

Define $y_0^{\mathcal{H}} := E_{\mathcal{T}}[h_{\mathcal{T}}(x_0)]$ to be the expected label of x_0 over the training sets. Then

$$\begin{aligned} MSE(x_0) &= E_{\mathcal{T}}\Big[\,\big(f(x_0) - y_0^{\mathcal{H}}\,\big) + \big(\,y_0^{\mathcal{H}} - h_{\mathcal{T}}(x_0)\,\big)\,\Big]^2 \\ &= E_{\mathcal{T}}\big[\,f(x_0) - y_0^{\mathcal{H}}\,\big]^2 + E_{\mathcal{T}}\big[\,y_0^{\mathcal{H}} - h_{\mathcal{T}}(x_0)\,\big]^2 \\ &\quad + \cancel{2E_{\mathcal{T}}\big[\big(f(x_0) - y_0^{\mathcal{H}}\big)\big(y_0^{\mathcal{H}} - h_{\mathcal{T}}(x_0)\big)\big]} \end{aligned}$$

The final term cancels since $E_{\mathcal{T}}[h_{\mathcal{T}}(x_0)] = y_0^{\mathcal{H}}$.

Estimation Error

So the mean squared error decomposes into a sum of squared terms

$$MSE(x_0) = E_{\mathcal{T}}\left[\, f(x_0) - y_0^{\mathcal{H}} \,\right]^2 + E_{\mathcal{T}}\left[\, y_0^{\mathcal{H}} - h_{\mathcal{T}}(x_0) \,\right]^2,$$

or, making the dependence on $\mathcal{T}$ explicit, we can write

$$MSE(x_0) = E_{\mathcal{T}}\left[\, f(x_0) - E_{\mathcal{T}}[h_{\mathcal{T}}(x_0)] \,\right]^2 + E_{\mathcal{T}}\left[\, E_{\mathcal{T}}[h_{\mathcal{T}}(x_0)] - h_{\mathcal{T}}(x_0) \,\right]^2.$$

The total error is often computed as the exception value of the mean squared error over all of $\mathcal{X}$:

$$E_{\mathcal{T}}\left[L_D(RSS(h_{\mathcal{T}}))\right] = \mathrm{Err} = E_{x_0}\left[\, MSE(x_0) \,\right].$$

Estimation Error

The decomposition of the error at a point

$$MSE(x_0) = E_{\mathcal{T}}\left[\, f(x_0) - E_{\mathcal{T}}[h_{\mathcal{T}}(x_0)]\,\right]^2 + E_{\mathcal{T}}\left[\, E_{\mathcal{T}}[h_{\mathcal{T}}(x_0)] - h_{\mathcal{T}}(x_0)\,\right]^2$$

is known as the **Bias-Variance decomposition**.Pointwise,

> **Variance**: $\mathrm{Var}_{\mathcal{T}}(h_{\mathcal{T}}(x_0)) = E_{\mathcal{T}}\left[\, E_{\mathcal{T}}[h_{\mathcal{T}}(x_0)] - h_{\mathcal{T}}(x_0)\,\right]^2$ is the variance of the predicted label $\hat{y}_0$ as we fit to different samples $\mathcal{T}$.
>
> **Bias**: $\mathrm{Bias}^2(h_{\mathcal{T}}(x_0)) = \left(\, f(x_0) - y_0^{\mathcal{H}}\,\right)^2$ here is the difference between the average labeling and the true labeling.

Notice that this definition of bias uses the average classifier instead of the error minimizing classifier. As usual these definitions are both widely used, but the average error definition seems to be used more in less theoretical applications.

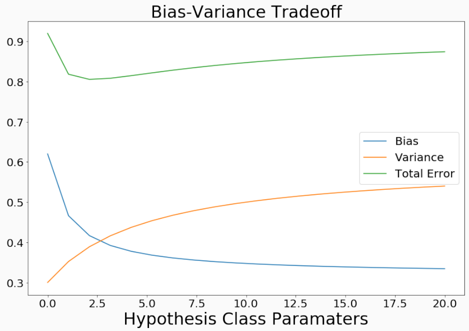
Families of Hypothesis Classes

Most learning algorithms depend on **hyperparameters** which are not fit, but allow us to shift between hypothesis classes. We can attempt to control the trade off between bias and variance by tuning these parameters, either by hand or algorithmically.

For example, k nearest neighbors is actually a family of hypothesis classes, with each class depending on the parameter k.

The class of linear predictors on the other hand do not directly depend on extra parameters. But the class can be extended in several ways: adding stochastic parameters, extending to polynomial predictors, or reducing to the p most important features are all examples of imedding the linear classifiers in wider family of hypothesis classes.

Bias, Variance and Parameters



It is very common when analyzing a hypothesis class on an data set to form a chart off the hyperparameters vs bias and complexity for parameter tuning.

Bias and Variance With Noise

k nearest neighbors

For a classification task, assume that in fact $Y = f_*(X) + \epsilon$, where ϵ is a random variable with $E[\epsilon] = 0$ and $\mathrm{Var}[\epsilon] = \sigma_\epsilon^2$. We will denote the ERM predictor by $\hat{f} := h_{\mathcal{T}}$ to match the notation in HTF.

The expected prediction error for a regression fit

$$Err(x_0) = E_{\mathcal{T}}\left[y - \hat{f}(x_0)\right]^2 = E_{\mathcal{T}}\left[f_*(x_0) + \epsilon - \hat{f}(x_0)\right]^2$$

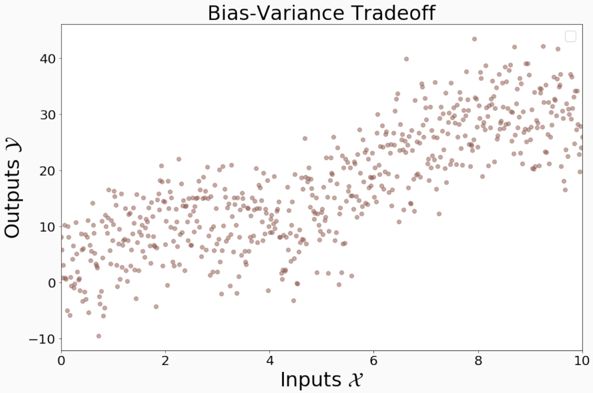
can be shown (**exercise**) to decompose as

$$Err(x_0) = \sigma_\epsilon^2 + [E_{\mathcal{T}}[\hat{f}(x_0)] - f_*(x_0)]^2 + E_{\mathcal{T}}\left[\hat{f}(x_0) - E_{\mathcal{T}}[\hat{f}(x_0)]\right]^2.$$

That is, into the bias, variance, irreducible error decomposition

$$Err(x_0) = \text{Irreducible Error} + \text{Bias}^2 + \text{Variance}.$$

Bias, Variance and Parameters

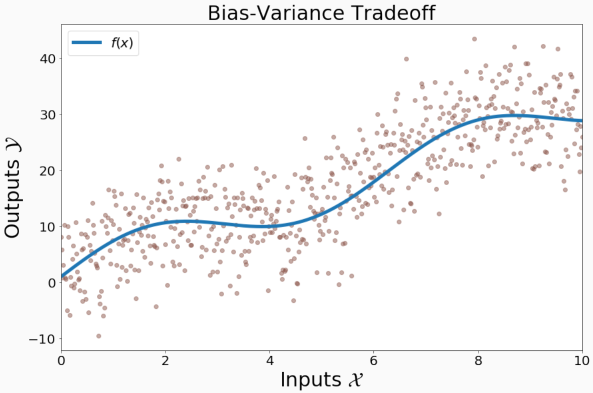


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2.$$

Consider a data set,

Bias, Variance and Parameters

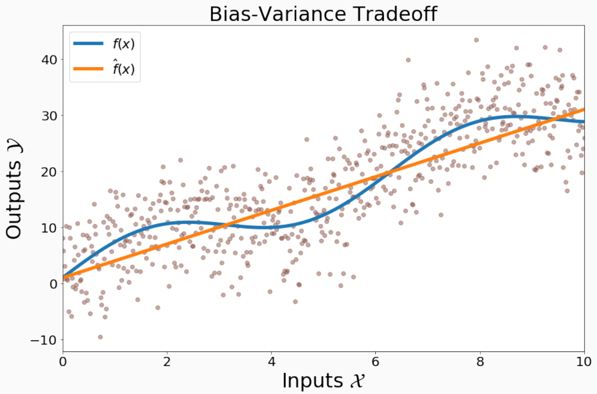


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2.$$

Consider a data set, generated by $f(x) + \epsilon$,

Bias, Variance and Parameters

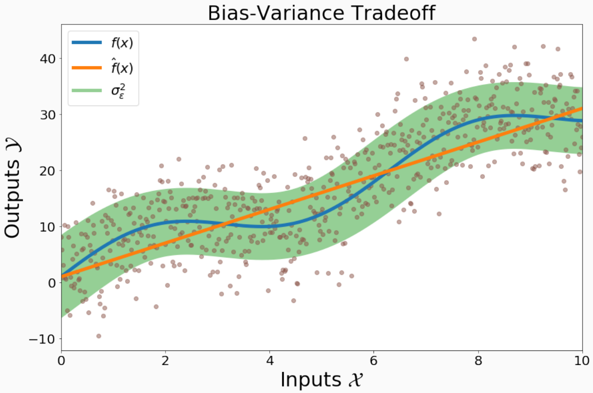


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_{\mathcal{T}}[\hat{f}(x_0)] - f(x_0)]^2 + E_{\mathcal{T}}\left[\hat{f}(x_0) - E_{\mathcal{T}}[\hat{f}(x_0)]\right]^2 .$$

Consider a data set, generated by $f(x) + \epsilon$, fit to $\hat{f}$.

Bias, Variance and Parameters

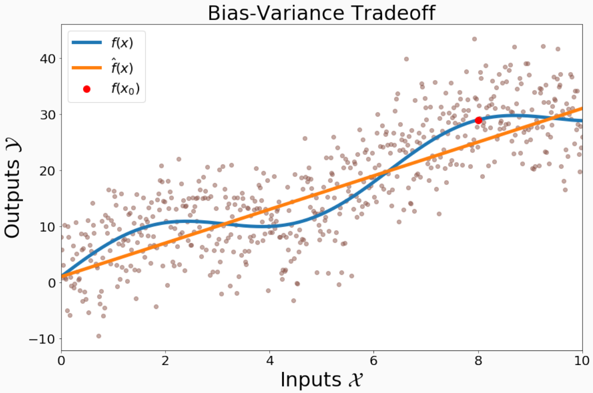


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + \left[E_{\mathcal{T}}[\hat{f}(x_0)] - f(x_0)\right]^2 + E_{\mathcal{T}}\left[\hat{f}(x_0) - E_{\mathcal{T}}[\hat{f}(x_0)]\right]^2.$$

The irreducible error comes from the variance σ_ϵ^2.

Bias, Variance and Parameters

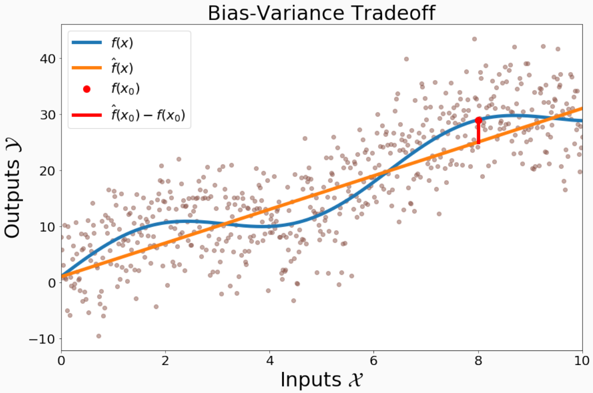


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2 .$$

The **bias** is the average distance between our estimate of $\hat{y}$ and the best estimate.

Bias, Variance and Parameters

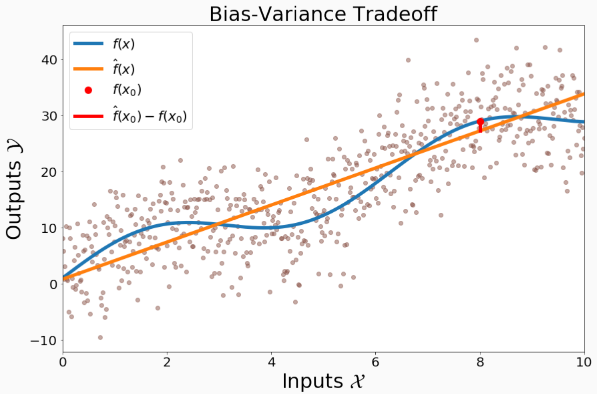


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2.$$

The **bias** is the average distance between our estimate of $\hat{y}$ and the best estimate.

Bias, Variance and Parameters

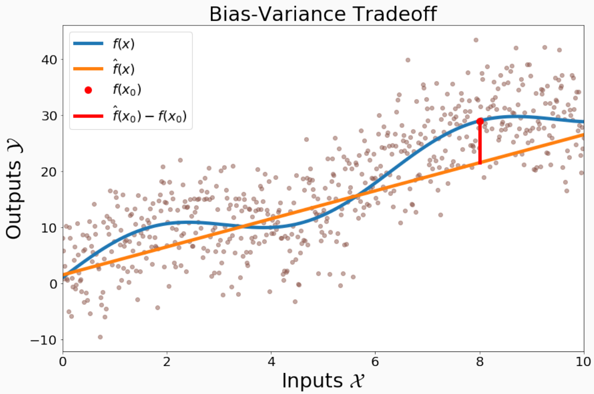


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2.$$

The **bias** is the average distance between our estimate of $\hat{y}$ and the best estimate.

Bias, Variance and Parameters

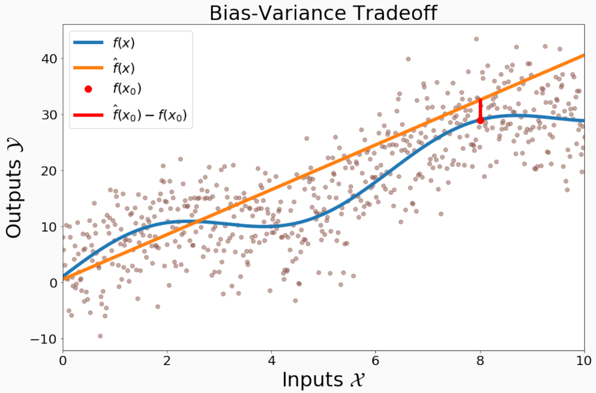


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2.$$

The **bias** is the average distance between our estimate of $\hat{y}$ and the best estimate.

Bias, Variance and Parameters

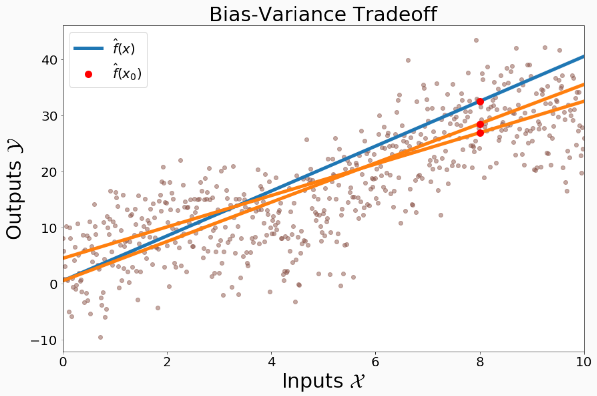


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2.$$

The **bias** is the average distance between our estimate of $\hat{y}$ and the best estimate.

Bias, Variance and Parameters

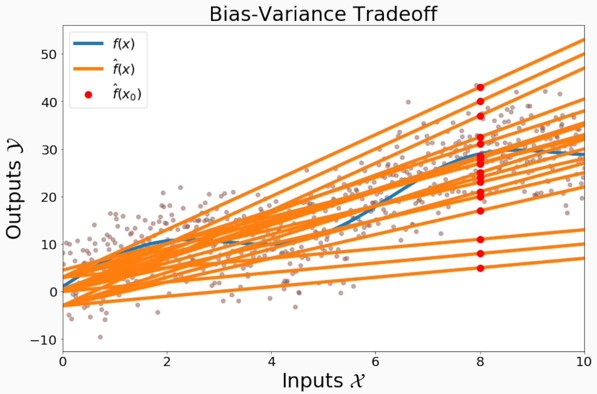


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2.$$

The **bias** is the average distance between our estimate of $\hat{y}$ and the best estimate.

Bias, Variance and Parameters

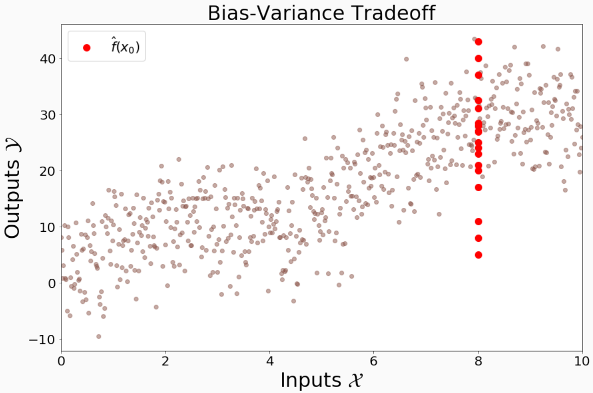


Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2.$$

The **variance** is the variance of the prediction for x_0 as we fit to different training data.

Bias, Variance and Parameters



Lets understand this visually.

$$Err(x_0) = \sigma_\epsilon^2 + [E_\mathcal{T}[\hat{f}(x_0)] - f(x_0)]^2 + E_\mathcal{T}\left[\hat{f}(x_0) - E_\mathcal{T}[\hat{f}(x_0)]\right]^2.$$

The **variance** is the variance of the prediction for x_0 as we fit to different training data.

Example: k nearest neighbors

k nearest neighbors

While we have been working at some level of abstraction, it's important to note that for specific algorithms quite a bit can be said, often up to deriving explicit formula for the bias and variance.

We will now derive the bias and complexity explicitly for the k nearest neighbors algorithm (for fixed input points), and show how the trade off depends both on the underlying data and the parameter k.

k nearest neighbors

For k nearest neighbor, the terms

$$Err(x_0) = \text{Irreducible Error} + \text{Bias}^2 + \text{Variance},$$

have a simple form. If we allow ourselves to fix the x_i and let the randomness come from the labels, the error can be decomposed as

$$Err(x_0) = \sigma_\epsilon^2 + \underbrace{\left[f_*(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})\right]^2}_{\textbf{Bias}^2} + \underbrace{\frac{\sigma_\epsilon^2}{k}}_{\textbf{Var.}} .$$

We will take a moment to show how this expression is derived.

k nearest neighbors: Variance

To compute the variance, we just need to use linearity.

$$
\begin{aligned}
E\left[\hat{f}(x_0) - E[\hat{f}(x_0)]\right]^2 &= \mathrm{Var}_{\mathcal{T}}\left(\frac{1}{k}\sum_{\ell} y_{(\ell)}\right) && \text{Def. of } k\text{NN} \\
&= \frac{1}{k^2}\sum_{\ell} \mathrm{Var}(f_*(x_{(\ell)}) + \epsilon_\ell) && \text{Linearity} \\
&= \frac{1}{k^2}\sum_{\ell} \mathrm{Var}(f_*(x_{(\ell)})) + \mathrm{Var}(\epsilon_\ell) && \text{Linearity}\,.
\end{aligned}
$$

By our (strong) assumption that the x_i are fixed, the variance in $f_*(x_{(\ell)})$ is 0, so all the variance comes from ϵ:

$$
E\left[\hat{f}(x_0) - E[\hat{f}(x_0)]\right]^2 = \frac{1}{k^2} k\sigma_\epsilon^2 = \frac{\sigma_\epsilon^2}{k}\,.
$$

k nearest neighbors: Bias

The bias term is also easily computed in the case of fixed neighbors:

$$\begin{aligned} E_{\mathcal{T}}[\hat{f}(x_0)] - f_*(x_0) &= E_{\mathcal{T}}\left[\frac{1}{k}\sum_{\ell} f_*(x_{(\ell)}) + \epsilon_i\right] - f_*(x_0) \\ &= \frac{1}{k}\sum_{\ell} f_*(x_{(\ell)}) - f_*(x_0)\,. \end{aligned}$$

The second line follow from the assumption that $E[\epsilon] = 0$ and the fact that for a fixed training inputs x_i, the average over training sets amounts to an averaging over all labels.

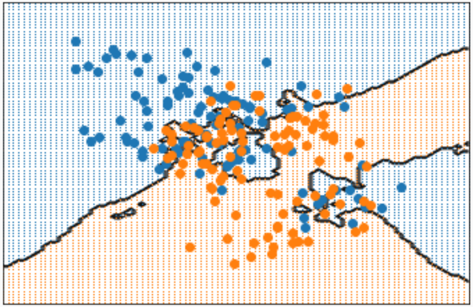
k nearest neighbors: Bias

Under the assumption that the training inputs are fixed, the point wise error can be represented as

$$Err(x_0) = \sigma_\epsilon^2 + \left[f_*(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})\right]^2 + \frac{\sigma_\epsilon^2}{k}\,.$$

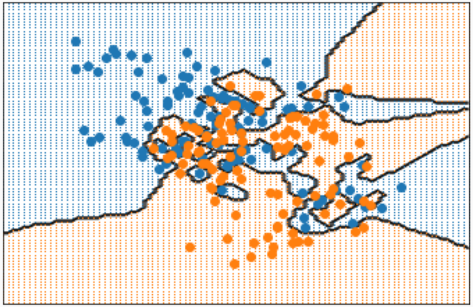
We see that the number of neighbors is inversely related to the model complexity. For small k, the estimate can potential adapt better to $f(x)$, leaving a smaller bias. For large k we would expect the bias to grow, while the variance dies off.

k nearest neighbors: Error



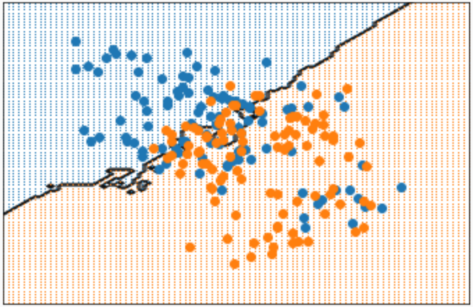
We see that the number of neighbors is inversely related to the model complexity. For small k, the estimate can potential adapt better to $f(x)$, leaving a smaller bias. For large k we would expect the bias to grow, while the variance dies off.

k nearest neighbors: Error



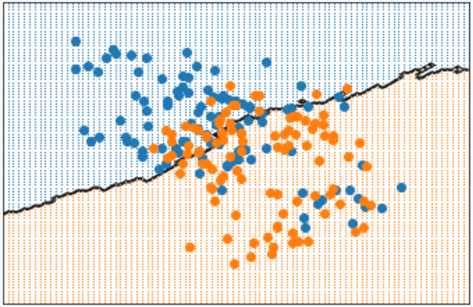
We see that the number of neighbors is inversely related to the model complexity. For small k, the estimate can potential adapt better to $f(x)$, leaving a smaller bias. For large k we would expect the bias to grow, while the variance dies off.

k nearest neighbors: Error



We see that the number of neighbors is inversely related to the model complexity. For small k, the estimate can potential adapt better to $f(x)$, leaving a smaller bias. For large k we would expect the bias to grow, while the variance dies off.

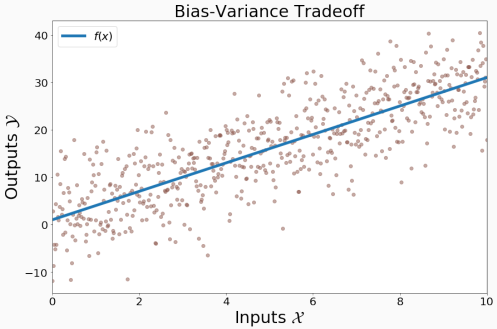
k nearest neighbors: Error



We see that the number of neighbors is inversely related to the model complexity. For small k, the estimate can potential adapt better to $f(x)$, leaving a smaller bias. For large k we would expect the bias to grow, while the variance dies off.

Example: Linear Predictor

Linear Classifier with Noise

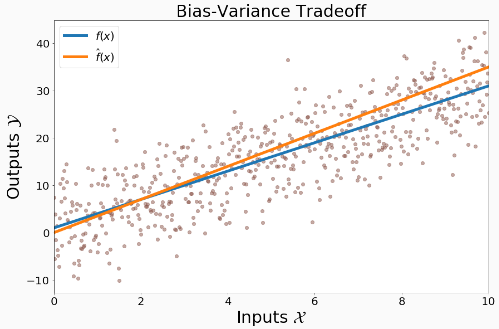


Suppose that we know that the relationship between X and Y is almost linear

$$Y = f_*(X) = X^T\beta_* + \epsilon\,,$$

where β_* denotes the actual weights and ϵ is a random variable drawn from a normal distribution with $E[\epsilon] = 0$ and variance σ^2.

Linear Predictor with Noise



Denote the fit linear predictor by $\hat{Y} = \hat{f}(X) = X^T\hat{\beta}$, the bias-variance trade off will again take a nice form:

$$Err(x_0) = \sigma_\epsilon^2 + \underbrace{E_{\mathcal{T}} x_0^T(\mathbf{X}^T\mathbf{X})^{-1}x_0\sigma_\epsilon^2}_{\textbf{Var}} + \underbrace{0^2}_{\textbf{Bias}^2}.$$

In such a case, the classifier is called **unbiased predictor** indicating that it attains the lowest possible squared bias.

Unbiased Predictor

We will first show that the linear predictor is unbiased. Recall that $\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$ and $\mathbf{Y} = \mathbf{X}\beta_* + \epsilon$,

$$\begin{aligned}
\textbf{Bias} &= f_*(x_0) - E_{\mathcal{T}}[\hat{f}(x_0)] && \\
&= x_0^T\beta_* - E_{\mathcal{T}}[x_0(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}] && \text{Def. of } \hat{f}. \\
&= x_0^T\beta_* - E_{\mathcal{T}}[x_0(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\beta_* + \epsilon)] && \text{Def. of } \mathbf{Y}. \\
&= x_0^T\beta_* - E_{\mathcal{T}}[x_0\beta_* + x_0(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon)] && AA^{-1} = I \\
&= x_0^T\beta_* - x_0\beta_* + x_0 E_{\mathcal{T}}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]\, E_\epsilon[\epsilon] && {}^*\text{See Note} \\
&= 0 && E_\epsilon[\epsilon] = 0.
\end{aligned}$$

So we see that when averaged over all training sets, $\hat{f}$ gives the actual labeling function f_*, up to a random variable of mean 0.

* Since the variance in the labels is independent of the choice of training samples, the expectation value splits.

Unbiased Predictor

The last slide relied on an important note that I want to say a bit more about here. By our assumptions, the distribution on $\mathcal{X} \times \mathcal{Y}$ is produced by a true labeling, plus an independent stochastic parameter:

$$Y = X^T \beta_* + \epsilon \,.$$

This means that any i.i.d. choice of training samples $(x_i, y_i = x_i^T \beta + \epsilon_i)$ is made up of N choices of X and N **independent** choices of ϵ. Fubini's theorem then allows us compute the expectation values of products separately:

$$E_{\mathcal{T}}[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon] = E_X[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T] \cdot E_\epsilon[\epsilon] \,,$$

where each expectation is taken of N i.i.d. choices of the variable.

Point Variance of Linear Predictor

We will now compute the variance. Since the bias is 0, averaging $\hat{f}$ over all training gives f_*. Then

$$\begin{aligned}
\mathbf{Var} &= E_{\mathcal{T}}\big[\,\hat{f}(x_0) - E_{\mathcal{T}}[\hat{f}(x_0)]\,\big]^2 \\
&= E_{\mathcal{T}}\big[\,x_0^T\hat{\beta} - x_0^T\beta_*\,\big]^2 && \text{Def. of } \hat{f},\ f_*. \\
&= E_{\mathcal{T}}\big[x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{X}\beta_* + \epsilon) - x_0^T\beta_*\big]^2 && \text{Def. of } \hat{\beta},\ \mathbf{Y}. \\
&= E_{\mathcal{T}}[x_0^T\beta_* + x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon - x_0^T\beta_*]^2 && AA^{-1} = I \\
&= E_{\mathcal{T}}\big[(\,x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon\,)^2\big]
\end{aligned}$$

We want to extract the dependence of the variance on the random variables.

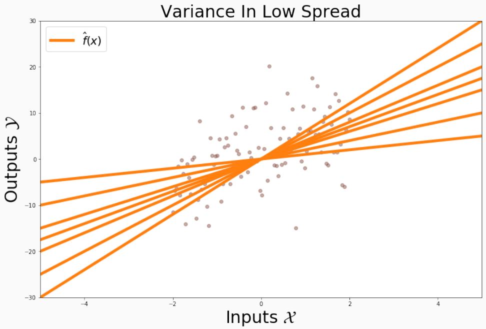
Point Variance of Linear Predictor

Since $x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon$ is a vector, squaring it is the same as multiply by its transpose. This allow us to write

$$\begin{aligned}
\mathbf{Var} &= E_{\mathcal{T}}\left[(\, x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon\,)^2\right] && \text{From before,}\\
&= E_{\mathcal{T}}\left[(\, x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon\,)(\, x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon\,)^T\right] && \\
&= E_{\mathcal{T}}\left[(\, x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\epsilon\,)(\epsilon^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\, x_0)\right] && \\
&= E_X\,[(\, x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)E_\epsilon[\epsilon\epsilon^T](\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}x_0)\,] && \mathbf{X},\ \epsilon\,\text{indep,}\\
&= E_X\,[(\, x_0^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T)\,(\sigma_\epsilon^2 I)\,(\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}x_0)\,] && \text{Def. of Var,}\\
&= \sigma_\epsilon^2\ x_0^T E_X[(\mathbf{X}^T\mathbf{X})^{-1}]x_0 && \text{Simplify.}
\end{aligned}$$

The variance is proportional to the variance in the random variable. But how do we understand the matrix $(\mathbf{X}^T\mathbf{X})^{-1}$?

Linear Predictor with Noise

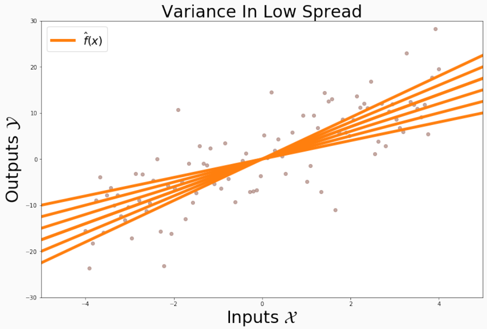


Assume the input has one feature, and so $\mathbf{X}$ is a $N \times 2$ matrix contains a column of 1's and a single variable worth of input. In this case, x_0 is a scalar. Then the lower right element of $\sigma_\epsilon^2(\mathbf{X}^T\mathbf{X})^{-1}$ is

$$\frac{\sigma_\epsilon^2}{\sum_{i=1}^N (x_i - \bar{x})^2}.$$

The denominator is the variance in the x_i coordinate of the data.

Linear Predictor with Noise



In particular, this is telling us that we lower the variance of the linear predictor by taking a wider spread of input data.

$$\frac{\sigma_\epsilon^2}{\sum_{i=1}^N (x_i - \bar{x})^2} \, .$$

Bias and Variance Compared

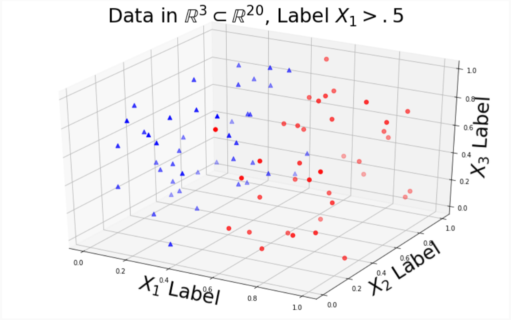
Best Subset Selection

The advantage of understanding the bias complexity trade off comes from our ability to tune hyperparameters, like the k in k-nearest neighbors. But the hypothesis class of linear predictors doesn't have any hyperparameters.

One way to introduce a parameter into the class of linear predictors is **best subset selection** of size p. The idea is to fit linear regressors to all subsets of the variables of size p and take the best performing result. This picks out the p most linearly predictive variables.

Best subset regression is often used when the feature space is so large that computing $(\mathbf{X}^T\mathbf{X})^{-1}$ becomes hard.

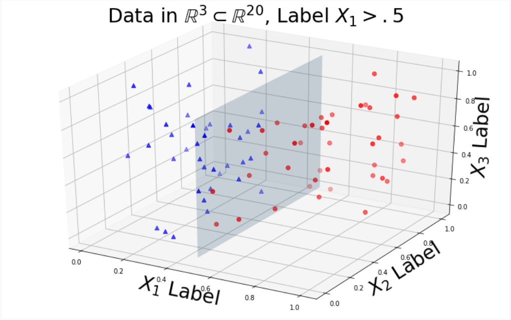
Test Data



We will test the bias variance trade off for two datasets: Both have input space $[0,1]^{20}$ and are labeled deterministically.

$$\textbf{Case 1:} \qquad f(X) = \begin{cases} 1 & \text{if } X_1 > .5\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

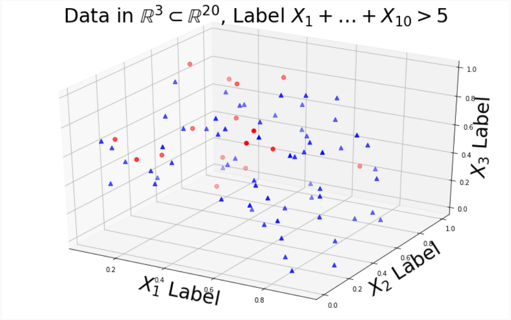
Test Data



We will test the bias variance trade off for two datasets: Both have input space $[0,1]^{20}$ and are labeled deterministically.

$$\textbf{Case 1:} \qquad f(X) = \begin{cases} 1 & \text{if } X_1 > .5\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

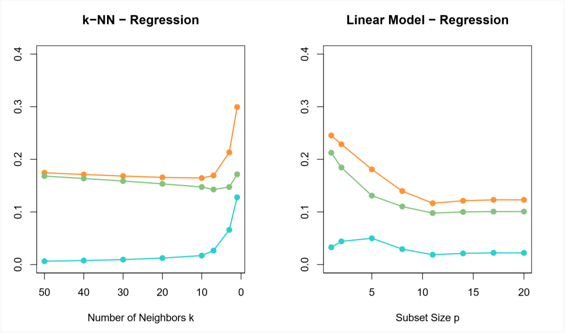
Test Data



We will test the bias variance trade off for two datasets: Both have input space $[0,1]^{20}$ and are labeled deterministically.

$$\textbf{Case 2:} \qquad f(X) = \begin{cases} 1 & \text{if } X_1 + \ldots + X_{10} > .5\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

Bias and Variance for Regression

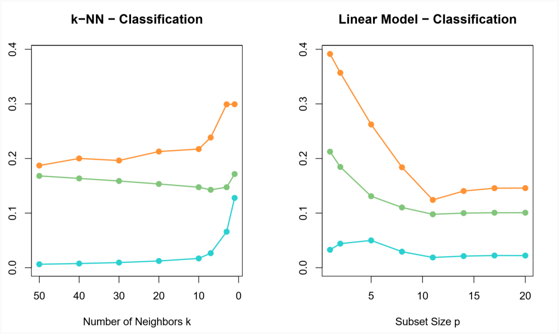


Here we see the expected prediction error, bias^2 and variance.

On the left we have a kNN regression fit of Case 1 and on the right we have a linear regression fit of Case 2.

Bias and variance add to total loss, with kNN having a minimum loss around $k = 5$ and the linear predictor being minimized around $p = 10$.

Bias and Variance for Classification

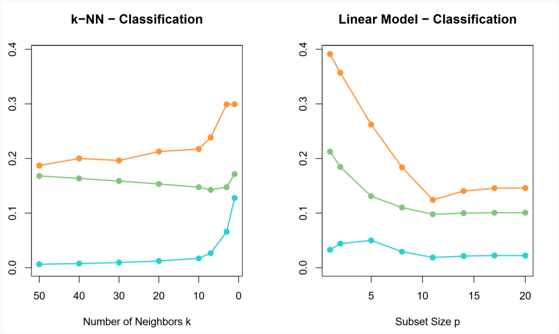


Here we see the expected prediction error, bias^2 and variance.

On the left we have a kNN fit of Case 1 and on the right we have a linear fit of Case 2, but now we use **0-1 loss**.

Notice however that bias and variance no longer add to total loss.

Bias and Variance for Classification



The reason is a follows: At a given input, suppose the true probability of class 1 is .9, but our predictor guesses .6. Then the squared error $(.9-.6)^2$ is large, but the 0-1 error is 0, since we make the correct prediction.

The bias-complexity decomposition works differently depending on the loss function, but it is always there. But the variance terms appearing are a result of using squared error loss.

Some Words about Bias and Variance - Scott Fortmann

Fight your instincts: Many people want to minimize bias even at the expense of variance, thinking that bias indicates a fundamental failure of their model while variance is just a fitting problem. The problem is high variance model only work as long run averages, and often you are working with a single data set. In such a situation, high variance can just a bad a high bias. One should not be improved at the cost of the other.

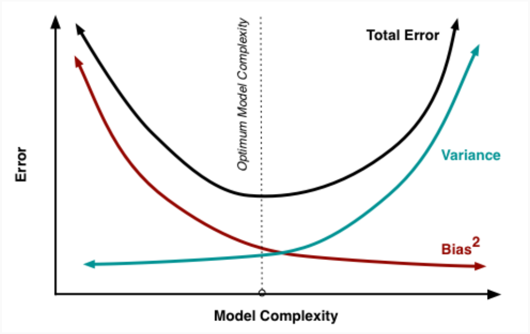
Bagging and Resampling: Variance can often be reduced by using techniques like Bootstrap AGgragatING and resampling to artificially expand your dataset. There is a whole forest of techniques to get around the bias complexity trade off, and we will spend the rest of the semester discussing them.

Some Words about Bias and Variance - Scott Fortmann

Asymptotic Accuracy vs Real Accuracy: As you dig deeper into the literature you will find results about asymptotically bias free algorithms. The idea behind these algorithms is that for a large enough data set, they can always drive the bias to 0. In the real world, we usually don't have a large enough data set. At the end of the day, trust the error you're actually recording.

Overfitting vs Underfitting: The sweet spot in any model is when the bias is decreasing at the same rate at which the variance is increasing.

Some Words about Bias and Variance



Overfitting vs Underfitting: The sweet spot in any model is when the bias is decreasing at the same rate at which the variance is increasing.

References

This lecture covers chapter 5 of Shalevl-Shwarts and Ben-David and section 2.5 and 7.3 of Hastie, Tibshirani and Friedman.

The advice on bias and variance was paraphrased from Scott Fortmann's blog: `http://scott.fortmann-roe.com/docs/BiasVariance.html`

Joseph Gonzalez has an excellent technical lecture on this material here: `https://people.eecs.berkeley.edu/~jegonzal/assets/slides/linear_regression.pdf`