Modeling Process

Applied Machine Learning for Educational Data Science

true

09/27/2021

Contents

[Updated: Tue, Sep 28, 2021 - 14:58:32]

Principle of Parcimony

How many parameters does it take to fit an elephant?

The Principle of Parsimony

Bias - Variance Tradeoff

When we use a model to predict an outcome, there are two main sources of error: model error and sampling error.

Model Error: Given that no model is a full representation of truth underlying observed data, every model is misspecified to some degree. Conceptually, we can define the model error as the distance between the model and true generating mechanism underlying data. Technically, for a given set of predictors, it is the difference between the expected value predicted by the model and the true value underlying data. The term **bias** is also commonly used for model error.

Sampling Error: Given that the amount of data is fixed during any modeling process, it will decrease the stability of parameter estimates for models with increasing complexity across samples drawn from the same population. Consequently, this will increase the variance of predictions (more variability of a predicted value across different samples) for a given set of same predictors. The term **estimation error** or **variance** is also used for sampling error.

The essence of any modeling activity is to balance these two sources of error and find a stable model (generalizable across different samples) with the least amount of bias.

We will do a simple Monte Carlo experimentation to better understand these two sources of error. Suppose that there is a true generating model underlying some observed data. This model is

$$y = e^{(x-0.3)^2} - 1 + \epsilon,$$

where x is a predictor variable that is equally spaced and ranges from 0 to 1, ϵ is a random error component and follows a normal distribution with a mean of zero and standard deviation of 0.1, and y is the outcome variable. Suppose that we simulate a small observed data following this model with a sample size of 20. Then, we use a very simple linear model to represent the observed simulated data.

$$y = \beta_0 + \beta_1 x + \epsilon$$

```
set.seed(09282021)

N = 20

x <- seq(0,1,length=20)

x
```

```
 [1] 0.00000000 0.05263158 0.10526316 0.15789474 0.21052632 0.26315789
 [7] 0.31578947 0.36842105 0.42105263 0.47368421 0.52631579 0.57894737
[13] 0.63157895 0.68421053 0.73684211 0.78947368 0.84210526 0.89473684
[19] 0.94736842 1.00000000
```

```
e <- rnorm(20,0,.1)

e
```

```
 [1]  0.07372726  0.08253427  0.13678980 -0.04993081  0.10368134  0.28473311
 [7] -0.03402811 -0.01834963 -0.02296964  0.02782503 -0.15425785 -0.13371024
[13]  0.03465939  0.21786527 -0.09607842  0.07927619  0.11618340 -0.19217742
[19] -0.07000210 -0.05165884
```

```
y <- exp((x-0.3)^2) - 1 + e

y
```

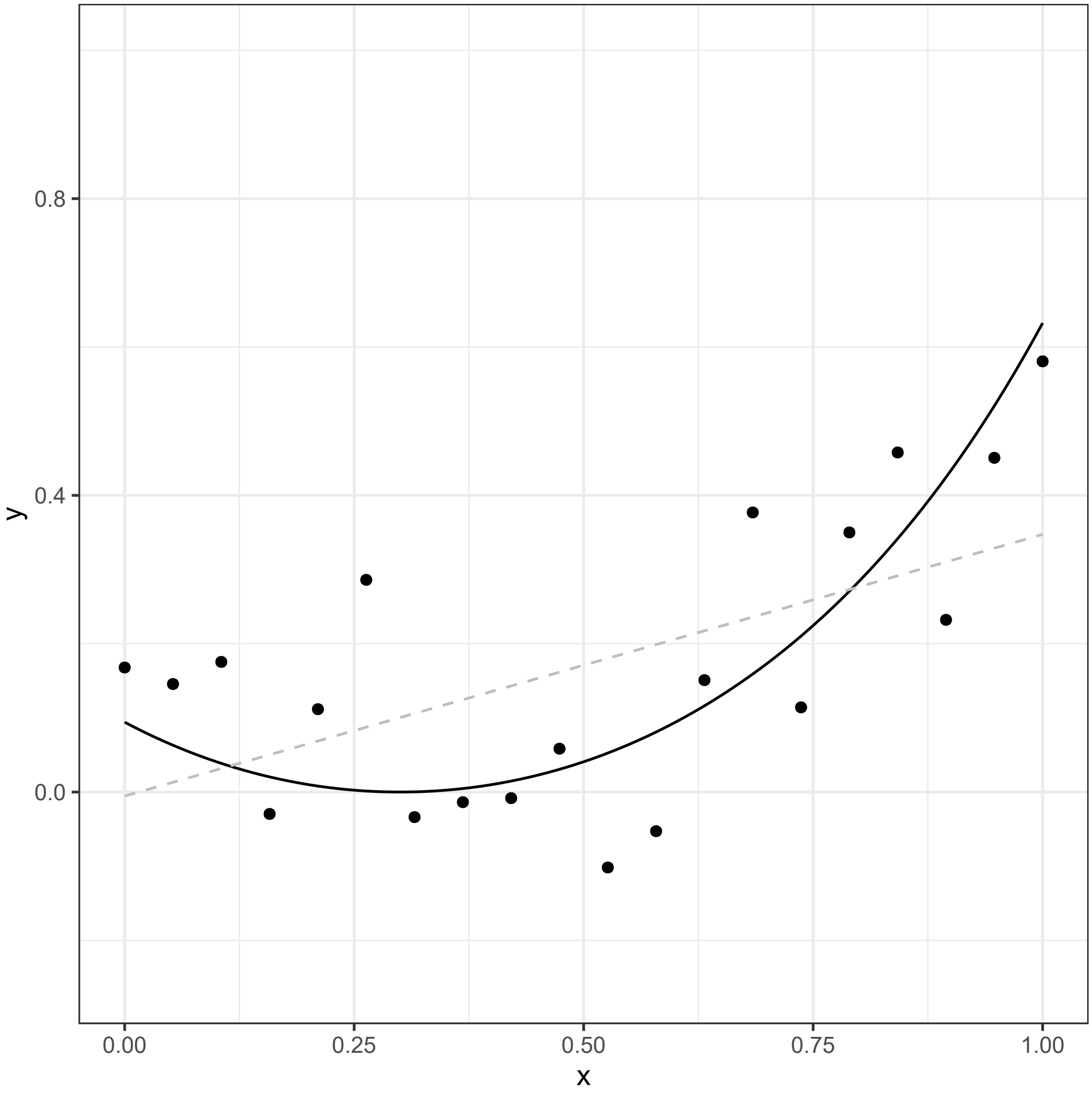
```
 [1]  0.167901540  0.145636360  0.175440469 -0.029531625  0.111719007
 [6]  0.286091377 -0.033778768 -0.013657214 -0.008208013  0.058450844
[11] -0.101704649 -0.052791207  0.150875614  0.376935001  0.114176511
[16]  0.349997226  0.457803769  0.232167369  0.450568829  0.580657384
```

```
mod <- lm(y ~ 1 + x)
mod
```

```
Call:
lm(formula = y ~ 1 + x)

Coefficients:
(Intercept)            x
   -0.00542      0.35272
```

```r
predict(mod)
```

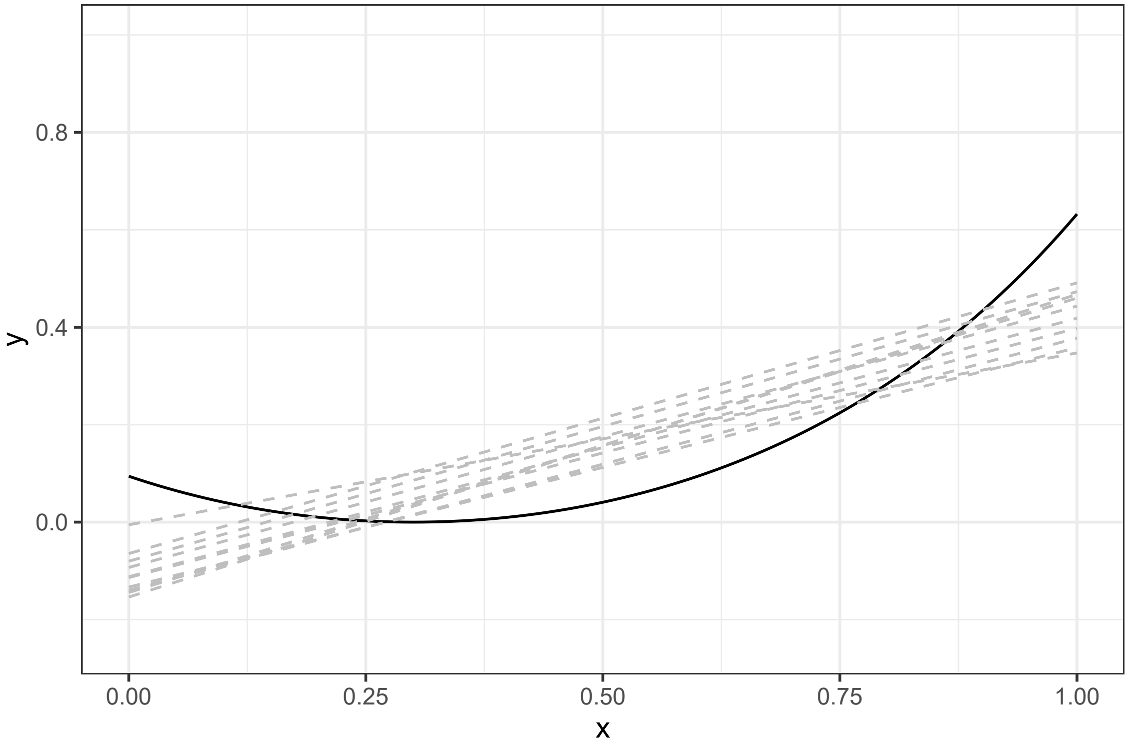
```
           1            2            3            4            5            6 
-0.005420071  0.013143883  0.031707837  0.050271791  0.068835745  0.087399698 
           7            8            9           10           11           12 
 0.105963652  0.124527606  0.143091560  0.161655514  0.180219468  0.198783422 
          13           14           15           16           17           18 
 0.217347376  0.235911330  0.254475284  0.273039238  0.291603192  0.310167146 
          19           20 
 0.328731099  0.347295053 
```



The solid line in this plot represents the true nature of the relationship between x and y. The observed data points do not lie on this line due to random error component (noise). The gray dashed line is how we represent the relationship between x and y if we use a simple linear model.

This demonstration only represents a single dataset. Now, suppose that we repeat the same process 10 times. We will produce 10 different datasets with the same size (N=20) using the exact same predictor values (x) and true data generating model. Then, we will fit a simple linear model to each one of these 10 datasets.

```
set.seed(09282021)

E  <- vector('list',10)
Y  <- vector('list',10)
M1 <- vector('list',10)

N = 20

x <- seq(0,1,length=N)

for(i in 1:10){

  E[[i]]  <- rnorm(N,0,.1)
  Y[[i]]  <- exp((x-0.3)^2) - 1 + E[[i]]

  M1[[i]] <- lm(Y[[i]] ~ 1 + x)
}
```



The solid line again represents the true nature of the relationship between x and y. There are 10 different lines (gray, dashed) and each lines represents a simple linear model fitted to different simulated data from the exact same data generating mechanism. The table below provides a more detailed look at the fitted values from each replication for evvery single x value.

x	y (TRUE)	Rep 1	Rep 2	Rep 3	Rep 4	Rep 5	Rep 6	Rep 7	Rep 8	Rep 9	Rep 10
0.000	0.094	-0.005	-0.144	-0.112	-0.154	-0.065	-0.093	-0.114	-0.133	-0.140	-0.080
0.053	0.063	0.013	-0.113	-0.084	-0.121	-0.035	-0.065	-0.087	-0.108	-0.113	-0.051
0.105	0.039	0.032	-0.081	-0.056	-0.088	-0.006	-0.037	-0.060	-0.082	-0.085	-0.022
0.158	0.020	0.050	-0.049	-0.028	-0.056	0.023	-0.008	-0.033	-0.056	-0.058	0.007
0.211	0.008	0.069	-0.017	0.000	-0.023	0.052	0.020	-0.006	-0.030	-0.031	0.036
0.263	0.001	0.087	0.015	0.027	0.010	0.082	0.048	0.021	-0.004	-0.004	0.065
0.316	0.000	0.106	0.046	0.055	0.042	0.111	0.076	0.048	0.022	0.024	0.094
0.368	0.005	0.125	0.078	0.083	0.075	0.140	0.105	0.075	0.048	0.051	0.124
0.421	0.015	0.143	0.110	0.111	0.108	0.169	0.133	0.102	0.074	0.078	0.153
0.474	0.031	0.162	0.142	0.139	0.140	0.199	0.161	0.129	0.099	0.105	0.182
0.526	0.053	0.180	0.174	0.167	0.173	0.228	0.189	0.156	0.125	0.133	0.211
0.579	0.081	0.199	0.205	0.195	0.206	0.257	0.218	0.183	0.151	0.160	0.240
0.632	0.116	0.217	0.237	0.223	0.239	0.286	0.246	0.209	0.177	0.187	0.269
0.684	0.159	0.236	0.269	0.251	0.271	0.316	0.274	0.236	0.203	0.214	0.298
0.737	0.210	0.254	0.301	0.279	0.304	0.345	0.302	0.263	0.229	0.242	0.327
0.789	0.271	0.273	0.333	0.307	0.337	0.374	0.330	0.290	0.255	0.269	0.357
0.842	0.342	0.292	0.365	0.335	0.369	0.403	0.359	0.317	0.281	0.296	0.386
0.895	0.424	0.310	0.396	0.363	0.402	0.433	0.387	0.344	0.306	0.323	0.415
0.947	0.521	0.329	0.428	0.391	0.435	0.462	0.415	0.371	0.332	0.351	0.444
1.000	0.632	0.347	0.460	0.419	0.467	0.491	0.443	0.398	0.358	0.378	0.473