3/10/21

$$y_i = g(\vec{x}) + h^*(\vec{x}) - g(\vec{x}) + h^*(\vec{x}) - f(\vec{x}) + f(\vec{x}) - f(\vec{x})$$

errors: estimation misspecification ignorance

$\varepsilon(\vec{x})$

$e(\vec{x})$

$e_i := y_i - \hat{y}_i = y_i - g(\vec{x_i})$ $i = 1, \ldots, n$, rows of $\mathcal{D}$.

These $e_1, \ldots, e_n$ called "in-sample "residuals" because they come from $\mathcal{D}$ (the sample). Thus, SSE, RMSE, R^2, SSR are all "in-sample".

Thus, they can all be "fake". Why? Because we saw that if we make up new x-vectors, we can get any answer we want. We can make R^2 arbitrarily high and RMSE arbitrarily low. Thus, $e_1, \ldots e_n$ are not honest estimates of our prediction errors/prediction accuracy.

Imagine $\frac{\mathcal{D} \text{ past}}{\text{P* future}}$ | time ↓

a good way to honestly estimate prediction error/prediction accuracy is to compute $\vec{e}_* = \vec{y}_* - \vec{\hat{y}}$ ie on data you did not use to build your model g.

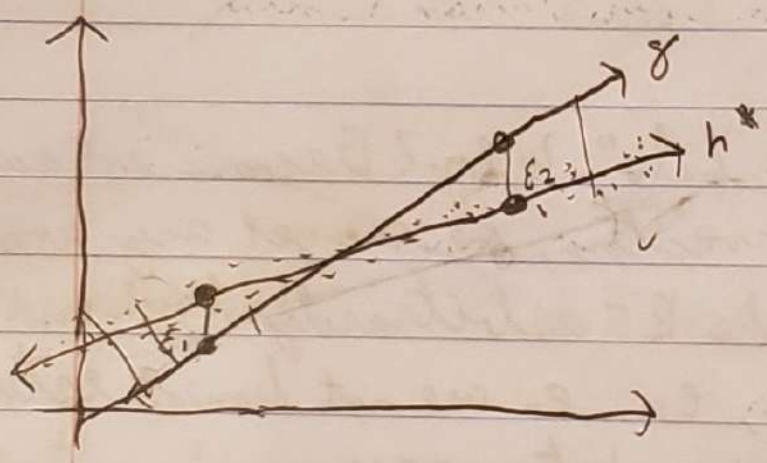
$\vec{y}_*$ cannot be overfit

For this to be true, we need another assumption, "stationarity" ie that $t(z \sim vec)$ stays the same and the relationships between the z's and the x's remain the same which means the function f doesn't change with time t. An example of a non-stationary relationship is stock prices explained by some variables z that are causal at one moment

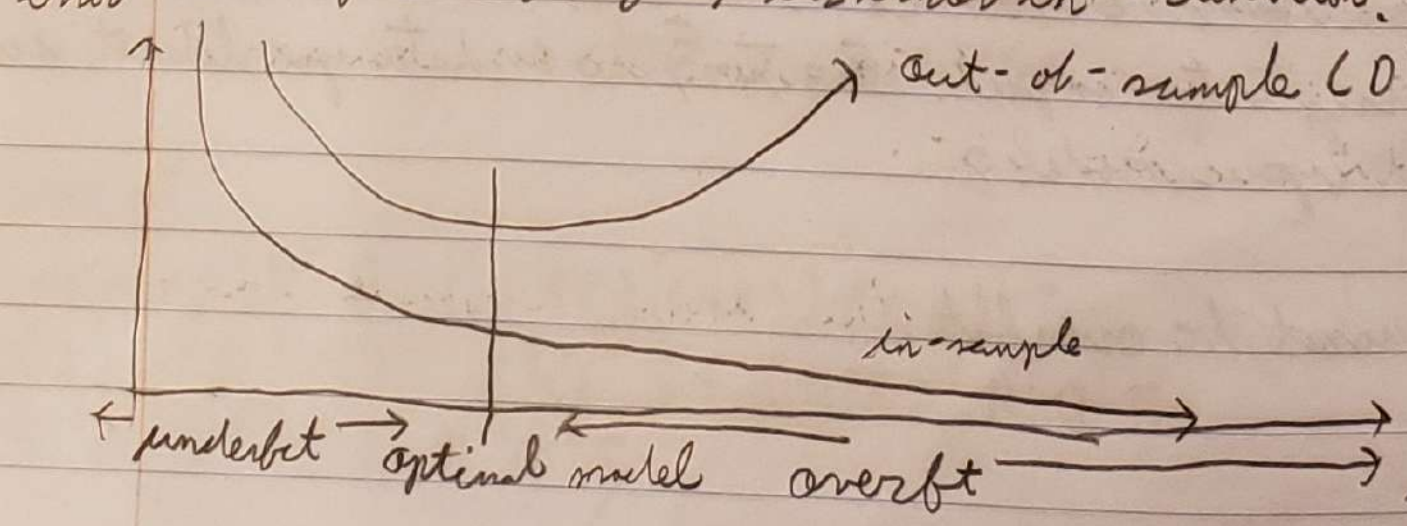
What is underfitting? Not learning as much as you can given your collection of x's

Overfitting in one dimension ($p=1$) with two observations ($n=2$)

$$y = h^*(x) + \varepsilon$$



Overfitting is fitting the epsilons which you know is a bad idea since epsilon = misspecification error + error due to ignorance. Fitting any degree of epsilon leads to a poorer model. Overfitting doesn't change h^* or f, it only changes $g = ?$ its error in estimation.



We need out of sample (honest) metrics:

$$\text{oos RMSE} = \sqrt{\frac{1}{n_* - (p+1)} \text{oos SSE}}$$

$$\approx \sqrt{\frac{1}{n_*} \text{oos SSE}}$$

$$\text{oos SSE} = \sum_{i=1}^{n_*} e_{*i}^2$$

$$\text{oos } R^2 = 1 - \frac{\text{oos SSE}}{SST_*} \quad SST_* = \sum (y_{*i} - \bar{y}_*)^2$$

oos error metrics require computation on data from the future (which you don't have). So how can we possibly compute oos error metrics? So.... assume stationarity and partition our original data set into:

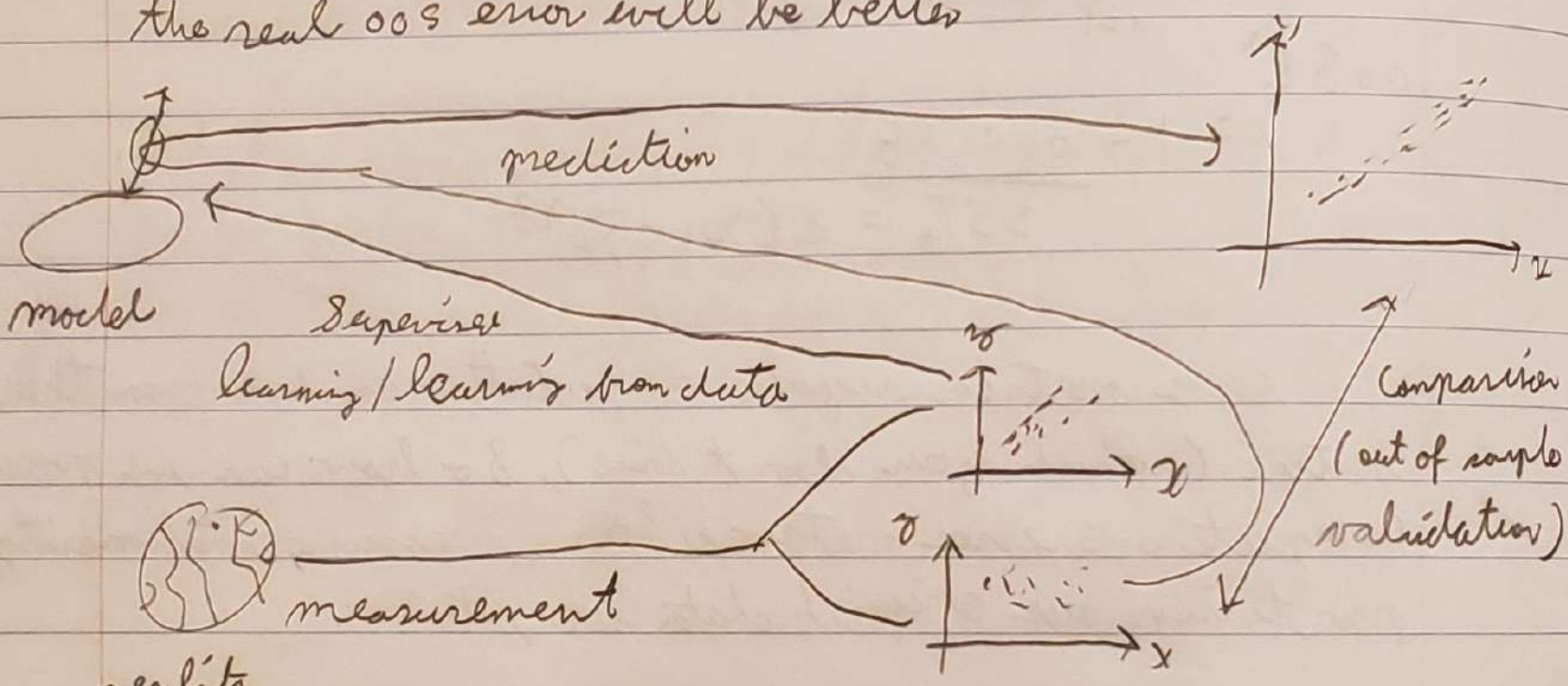
$$\mathbb{D} = \mathbb{D}_{train} \cup \mathbb{D}_{test}$$

Then use $\mathbb{D}_{train}$ the way we've been using $\mathbb{D}$ all along which is use it to create your model: $g = A(\mathbb{D}_{train}, \mathcal{U})$ and then use $\mathbb{D}_{test}$ to compute the honest oos error metrics to inform us of our model's predictive accuracy in the future (ie give us an estimate of our generalization error). Two issues:

1. What proportion of the data do you use for training?
2. How to do the split?

1. really has no answer but people generally use 80% or 90%
2. usually done randomly or use if there's a time trend in $\mathbb{D}$.

$n_{train} < n$, is there a cost to this? Yes, estimator error increases and thus, the generalization error estimate (oos error metric) will look a tad worse than they will be when using your model (since the model you will use in the future will be built with all D, all n). So your oos error is conservative, the real oos error will be better



$h^*(\vec{x}) - g(\vec{x}) = X\vec{\beta} - X\vec{b} = X(\vec{\beta} - \vec{b})$.

Estimation error in OLS

as b-vector diverges from the beta-vector, the estimation error goes up. One way to measurement it is via $\|\vec{\beta} - \vec{b}\|^2$

In the above procedure, there is only one oos validation comparison, If you see that you overfit by seeing that eg RMSE << oos RMSE, it is too late!, You can't correct the model honestly anymore. So... this is a big problem and we need to solve it

But brit, let's talks about reducing misspecification error.