# Thoughts on Machine Learning - An economist's perspective

Marius Grünewald

May 14, 2019

Traditionally, economists care a lot about causality. By that, we usually refer to statistical significance and the unbiased estimation of these standard errors.
There are good reasons for doing so. Firstly, we want to check the relevance of variables x and z for variable y. With relevance I mean whether y changes when x changes. Standard errors are among the most essential statistics for evaluating (t-statistics and confidence intervals rely on them). Subsequently, these standard errors determine if we consider x relevant so, putting emphasise on estimating the correct standard errors is quintessential for discovering causal mechanism empirically. After having estimated (the correct) standard errors, researcher can say with relative certainty whether x and z matter to determine y. For instance, x is important and should be included in further research on y while z doesn't have to be included. A different scholar can than modify her own regression by selecting x and an control variable whereas she omits z.
For this particular project, we are interested in forecasting. So, taking on the same idea as above, we now forecast y by multiplying the value of variable x by the estimated coefficient. (Note: Standard errors don't matter any more)
If you are an economist with at least some semesters of studying, this all seems natural and yet, there is another way to go. That way is called machine learning.
In short, we can apply a regularized OLS to determine which variables to include. But let's take a step back first and consider what to distort a forecast with the first approach. First of all, the *Bias*. Here, we assume a variable being irrelevant or not (in an OLS setting) based on the standard errors and excluding it even though it should have been included. Or, we ignored a potentially quadratic impact. Since we are interested in making the best predictions possible, an OLS where we choose the variables (supervised), is likely to underfit the data (not including enough variable for accurate predictions). The second issue is *Variance*. The more variables we include, the higher the variance of each estimated coefficient gets. What does that mean? Basically, we fit the model too well. For example, we have 25 observations and a polynomial of order 25 - we matched each single observation but the explanatory accuracy of a variable for all is low (resulting in high standard errors). Therefore, including too many variables will make predictions resembling the training data/sample too closely. A phenomenon called overfitting. Lastly, the irreducible error. Either exists or doesn't and we can't do much about it expect for changing the way we measure.
Economists almost always spot a trade off when there is one. And yes, we have a trade-off. In the normal OLS setting we would now try an iterative process of selection. For instance slowly expanding the set of controls and evaluating on the increased accuracy (R-squared for instance) and estimated standard errors until you think you have the optimum. Regularized regression does that for you. Lasso, Ridge and Elastic Net are three versions of it. In essence, they maximize the $R^2$ with a penalization term punishing the degrees of freedom. We still minimize the MSE but add something for each variable included. What do we add ? A vector of variable weights (have to be positive!). For now, we shall not discuss how the weights are determined.
The regularized regression gives the estimated coefficients of a set of variables that should be used for forecasting to get the best accuracy-overfitting trade off. Have you noticed that standard errors do not matter in this case? They did earlier for us, so that we can tell out program what to use but now the program teaches itself (unsupervised) what to use. And the mechanism doesn't depend on causal effects (and tbh forecasting doesn't either. Nobody cares about whether a variable picks up the effects of an unobserved variable as long as it is consistent in doing so.). That's nice and saves us a lot of work.
But why don't we do it all the time? You need a lot of data. So when the dataset is too small, the algorithm has not enough time to "learn" and your estimation might be worse off than before. Important note: We need even more data since our sample needs to be split. One part for testing (figuring out the parameters and stuff). The other part to evaluate how well it did in the end (so that you don't test your performance on the same data you learn and maximized everything on - makes sense right?). In other words, we want to know the general skills of our model - not the skills for one particular exercise.