Supervised Machine Learning vs. Traditional Regression: Similarity between Two Cost-Cutters

STUFF TO KNOW BY HEART - EVEN WHEN DRUNK!

1. ***Cost functions*** *measure degree of prediction errors: low when errors are small, high when they are large*
2. *Supervised learning is about achieving* ***low Hypothesis-vs.-Target costs***
3. ***Training*** *is mathematical optimization that makes cost decrease, i.e. make errors smaller*

In a supervised Machine Learning framework, just as in a traditional Regression setting, we give the machine Input $\mathbf{X}$ and observed "**right-answer**" Target Output $\mathbf{Y}$ and assume a hypothetical transformation function structure $h(\mathbf{X}, \mathbf{W})$, where $\mathbf{W}$ are adjustible coefficients/parameters so-called "**weights**", that transforms $\mathbf{X}$ to Hypothesized Output $\mathbf{H}$ that hopefully closely mimicks $\mathbf{Y}$.

- **Example 1**: x_1 = a house's area, x_2 = neighborhood's average income, x_3 = city's crime rate; h = predicted house price; y = actual house price
- **Example 2**: x_1 = a lady's expenditure coffee/alcohol/cigarettes, x_2 = her expenditure on maternity nutrition, x_3 = number of pregnancy-related vouchers she redeems, x_4 = her speed of completing one shopping trip; h = predicted **probability** that she is pregnant; y = whether she is actually pregnant. (Savvy data-driven businesspeople, if you have not heard of the Target story before, where have you been??)
- **Example 3** (nice "bigger data" example with multiple predictions): x_1 = a Booth MBA guy's most recent restaurant bill, in USD'000, when bringing a Harris Public Policy girl out, x_2 = TRUE/FALSE for whether he drove his (borrowed) car to pick her up, x_3 = TRUE/FALSE for whether he bought flowers, x_4 = number of hours he studied Micro last week, x_5 = TRUE/FALSE for whether he boasted about interviewing with JPStanleySachs for a fat job, x_6 = his student plus credit card debt, in USD'00,000; h_1 = probability he's gonna propose this week, h_2 = probability that Topel and Dean Kumar will kick him out next month, h_3 = probability that he'll be personally bankrupt, to be bailed out by JPStanleySachs for being too smart to fail; $\mathbf{Y}$ = actual outcomes corresponding to $\mathbf{H}$

The extent of $\mathbf{H}$'s similarity/closeness to $\mathbf{Y}$ - which is the criterion to judge how well a supervised Machine Learning model learns to mimick $\mathbf{Y}$ from knowing $\mathbf{X}$ - is numerically measured by a certain specified scalar cost function $c(\mathbf{H}, \mathbf{Y})$. This function c's value should be small when $\mathbf{H}$ is very "similar" or "close" to $\mathbf{Y}$, and large otherwise. In almost all supervised Machine Learning models in practical use nowadays, $c(\mathbf{H}, \mathbf{Y})$ is a continous real-valued function and its partial derivative $\frac{\partial c}{\partial \mathbf{H}}$ with respect to $\mathbf{H}$ is computable as a certain function $d(\mathbf{H}, \mathbf{Y})$. Costs are usually measured on an average per-case basis.

The task of helping a supervised Machine Learning model learn - or so-called "**training**" it - involves mathematical optimization procedures that adjust weights $\mathbf{W}$ to make the average per-case $\mathbf{H}$-vs.-$\mathbf{Y}$ cost decrease when we let the model see more and more cases of inputs and corresponding "right-answer" target outputs. Once trained until its cost has decreased to an acceptably low level, a model will make only small errors and hence be a good tool for predicting the output $\mathbf{y}$ from a not-yet-seen input $\mathbf{x}$.

Note that we are using the rather gentle phrase "acceptably low" instead of the stronger word "minimized". This is because when a supervised Machine Learning model does achieve the absolutely smallest possible error rate during the "training" process, it will have over-learned: not only will it have learned the overall rules of the game (which are useful when generalizing to new cases), it will have also **memorized various irrelevant idiosyncracies** specific to the training data (which **hurts** its generalization ability when facing new cases). We'll discuss this so-called "**over-fitting**" issue separately.