Lecture 21: MATH 342W: Introduction to Data Science and Machine Learning

Sergio E. Garcia Tapia*

April 29th, 2025 (last updated April 30, 2025)

Bagging

Recall bagging, depicting in Figure 1. How do we do validation for bagged models? Consider the first bootstrap sample $\mathbb{D}_1$, and the model we compute on it:

$$g_1 = \mathcal{A}(\mathbb{D}_1, \mathcal{H})$$

Then $\mathbb{D} \setminus \mathbb{D}_1$ is out-of-sample for g_1. Thus, the metrics computed on $\mathbb{D} \setminus \mathbb{D}_1$ are honest. We can do the same for g_2 computed on bootstrap sample $\mathbb{D}_2$ (likely different 1/3 of data missing than $\mathbb{D}_1$). We call this metric **out-of-bag (oob)**. If we do this M times, we have M models, each missing a slightly different 1/3 of the data (see Figure 2). Thus for each unit, we can collect the models for which that unit is out-of-bag and use it them to compute an out-of-bag metric for g_{avg}:

$$g_{avg} := \frac{1}{M} \sum_{m=1}^{M} g_m$$

*Based on lectures of Dr. Adam Kapelner at Queens College. See also the course GitHub page.

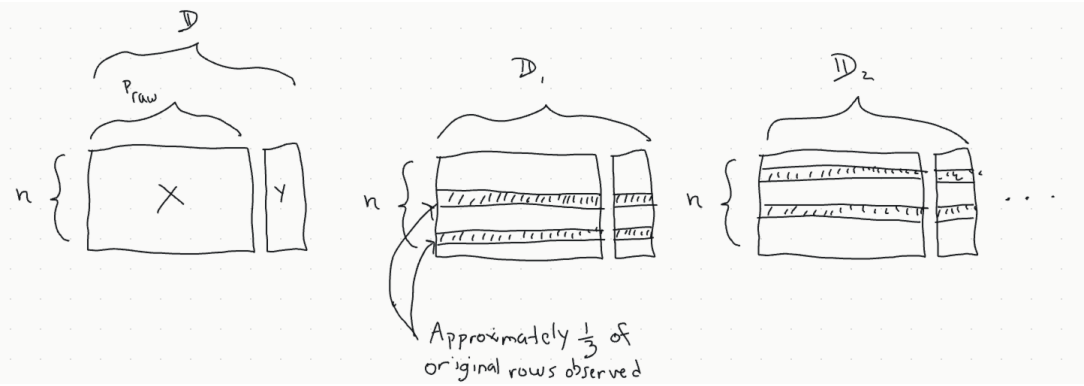

Figure 1: Depiction of bagging. Each bootstraped sample contains about $\frac{2}{3}n$ of the original data.

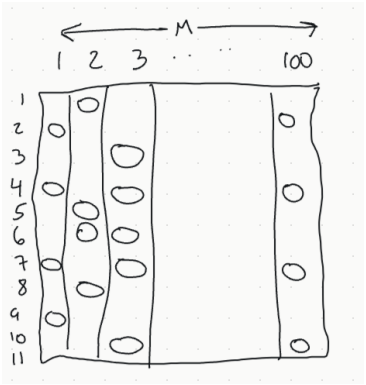

Figure 2: Out-of-bag data in bootstrap samples. Each circle depicts which data point is missing from a given bootstrap sample.

Example. Given M bootstrap models with index $\{1, 2, \ldots, M\}$, suppose models $17, 37, 88$ do not contain the observation $i = 3$ (i.e., the data point $(\boldsymbol{x}_3, y_3)$ is out of bag for them). Then we can use models g_{17}, g_{37}, g_{88} to compute $\hat{y}_{17}, \hat{y}_{37}, \hat{y}_{88}$ on $\boldsymbol{x}_3$. Thus, the out-of-bag value of g_{avg} for observation $i = 3$ is

$$\hat{y}_{oob,3} = \text{Average}(\hat{y}_{17}, \hat{y}_{37}, \hat{y}_{88})$$

We do this computation for all $i = 1, 2, \ldots, n$. In the sample above, the out-of-bag prediction was computed using 3 values. However, you will generally observe about $\frac{1}{3}M$ values included in the out-of-bag calculation.

Bagging is usually done with trees ($N_0 = 1$) because bias is low.

Random Forests

Recall ρ is the correlation between the trees. Thus, reducing ρ means reducing decoupling between the trees. In 2001, Breiman brought the ρ term in the Bias-Variance decomposition formula down even further. Breiman suggested that, instead of doing a full greedy search (as is done in regression trees), we can do the following:

Algorithm 1 (Random Forest Algorithm)**.** Use the bagged tree model with one change to the CART algorithm. Pick $m_{\text{try}} < p_{\text{raw}}$ and only search a random subset of $\{1, 2, \ldots p_{\text{raw}}\}$ of size m_{try} at each node. The "reasonable" defaults for m_{try} are $m_{\text{try}} = \lfloor p_{\text{raw}}/3 \rfloor$ for regression, and $\lfloor \sqrt{p_{\text{raw}}} \rfloor$ for classification.

Notice that the transition to random forests involves a change in the algorithm ($\mathcal{A}$), not $\mathcal{H}$ or the features.

Example. Let $p_{\text{raw}} = 10$, $m_{\text{try}} = 3$. For each root node, sample a subset of $m_{\text{try}} = 3$. For example, $\{x_3, x_7, x_9\}$, and do a greedy split based on those candidate splits only.

If m_{try} is p_{raw}, then we have done nothing, since we would have the original regression

tree algorithm. If m_{try} is too low (say, $m_{\text{try}} = 1$), then you risk having a bias that is too large due to underfitting. In spite of the "defaults" in place, you really should use model selection to optimize m_{try}, which is effectively a hyperparameter.

Random forests contain two types of randomness: in the units and in the features. The term forest comes from the fact that we have multiple trees.

R Demos

See `QC_MATH_342W_Spring_2025/practice_lectures/lec21.Rmd`.

Missingness

Suppose we have a data set $\mathbb{D}$ for which some units have missing features. That is, if we make a matrix X containing the values of the features of all the units, then there are missing entries:

In R, these would be `NA` values. Note that there is no missingness in the response y (otherwise, it would not be in $\mathbb{D}$). If we have missing data, we cannot do this:

$$\boldsymbol{b} = (X^\top X)^{-1} X^\top \boldsymbol{y}$$

For any of the algorithms $\mathcal{A}$ that we have discussed, we cannot have missing data. In the following lecture, we will discuss strategies for dealing with missing data in more detail.

The simplest strategy is called **listwise deletion**, where we drop all subjects (rows) from $\mathbb{D}$ where there is at least one missing value. You shoudl only do this if n is large and the number of missingness rows is "very small". A down side of this is it incurs more estimation. There is another problem, to discussed next time.