Week 6 - Variable Importance

Wednesday, March 5, 2025 7:36 PM

Motivation

1. One of the successes of modern statistics and machine learning is that we have a wide range of models that work well on high-dimensional data. Out-of-the-box, these models can build predictive rules from a large number of features -- no manual curation needed. However, unless the model performs explicit variable selection, this automation can leave the models' developers and users at a loss for which features were actually relevant.

2. Variable importance measures attempt to answer this question. Most approaches rely on the intuition that a feature is important if "removing it" causes model performance to deteriorate. Of course, the devil is in the details, and there are many ways to formalize this intuition.

3. This intuition might remind you of attribution using Shapley values, but there are two key (important?) differences. First, Shapley values are focused on local explanations, while most (though not all) variable importances attempt to characterize importance globally. Second, Shapley values focus on the estimated model, while variable importance focuses on the estimated model's _performance_.

Variable Importance in Random Forests

1. The phrase "variable importance" is closely associated with random forests, and for good reason -- the random forest paper included not just a method for estimating the model but also a method for estimating variable importance. The important variables were reported throughout the paper's illustrations, and its particular implementation has been influential.

2. Let's briefly review the original proposal before exploring some of its issues (and the follow-up work that addresses them). Random forests work by averaging the predictions from many different decision trees. To make the decision tree predictions less correlated with one another, each is trained on a different subsamples of both training data and features.

3. To compute the importance of feature d within tree t, we compare the out-of-sample prediction accuracy when using the true test data and a version of the test data with feature d randomly permuted:

$$VI_d^t = \frac{1}{N_{OOB}}\left[\sum_{OOB} \ell(y_n, f^t(x_n)) - \ell(y_n, f^t(x_n^\pi))\right]$$

$$x_n^\pi = (x_{n1} \ldots x_{\pi(n)d} \ldots x_{nD})$$

The importance of that feature in the RF averages importances across all trees:

4. Despite matching our initial intuition so well, we have to be extremely careful with how we interpret these importances. The absurdly simple table below comes from Efron's Prediction, Estimation, and Attribution paper.

Table 3: Number of test set errors for prostate cancer random forest predictions, removing top predictors shown in Figure 7.

# removed	0	1	5	10	20	40	80	160	348
# errors	1	0	3	1	1	2	2	2	0

There are 6032 total features in the original random forest. Removing the 348 "most important" of them leads to no noticeable decrease in test set performance!!

5. The trouble is that real data have correlated features. This correlation wreaks havoc on the usual intuition about variable importance. For this reason, a running theme in the literature since the original RF paper is: We need to be precise about what exactly we are estimating.

6. Or, in the case of Strobl et al. (2008), can we be precise about what we are testing? The permutation step in the VI calculation amounts to a test of conditional independence. To see this, first note that if the response were independent of the features, then

$$p(y, X) = p(y)p(x)$$

If we define a test statistic that is large when this factorization doesn't hold, then we will be able to test

$$H_0: y \perp\!\!\!\perp X$$

7. For example, our test statistic could be R^2. If R^2 is no higher on the real data than it is on datasets with the Y permuted, then we don't have any evidence that X is associated with Y. For random forest variable importance, we don't permute the response -- we permute individual features. If we had

$$p(y, X_d, X_{-d}) = p(X_d)p(y, X_{-d})$$

then the two sides of the VI definition would be comparable. Only when this independence is violated will a variable be flagged as independent. So RF's variable importance measure is implicitly testing

$$H_0: X_d \perp\!\!\!\perp y, X_{-d}$$

The issue here is that if X[d] is correlated with other features, then that will also lead to large variable importance.

8. Strobl et al. (2008) suggest to instead develop a permutation scheme that tests

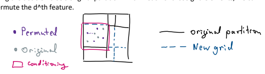
$$H_0: X_d \perp\!\!\!\perp y \mid X_{-d}$$

Feature d has to be associated with y even after controlling for the remaining features. This is accomplished by separately permuting X[d] within neighborhoods

Y	X_j	Z
y_1	$x_{\pi_j(1),j}$	z_1
$\vdots$	$\vdots$	$\vdots$
y_i	$x_{\pi_j(i),j}$	z_i
$\vdots$	$\vdots$	$\vdots$
y_n	$x_{\pi_j(n),j}$	z_n

Y	X_j	Z
y_1	$x_{\pi_{j\mid Z=a}(1),j}$	$z_1 = a$
y_3	$x_{\pi_{j\mid Z=a}(3),j}$	$z_3 = a$
y_{27}	$x_{\pi_{j\mid Z=a}(27),j}$	$z_{27} = a$
y_6	$x_{\pi_{j\mid Z=b}(6),j}$	$z_6 = b$
y_{14}	$x_{\pi_{j\mid Z=b}(14),j}$	$z_{14} = b$
y_{21}	$x_{\pi_{j\mid Z=b}(21),j}$	$z_{21} = b$
$\vdots$	$\vdots$	$\vdots$

If z is continuous, then we can't require exact matches. Instead, the proposal defines a grid using the decision tree's original partition. Within each of those grid elements, we can permute the d^th feature.



A further relaxation is to ignore grid dimensions for features X[-d] that are known not be very correlated with X[d]. This is a bit heuristic, but seems accurate enough in practice, and it does help with ensuring there are more samples to permute.

MDI+

1. We actually didn't need to use permutations to have a notion of variable importance in trees. If we consider how trees are built, we'll see that there's already a notion of importance built into the construction process. Specifically, when a decision tree is grown, it repeatedly computes the "decrease in impurity" associated defining a new split s at an existing tree node t:

$$\Delta(s) = \frac{1}{N(t)}\left[\sum_{n\in t}(y_n - \bar y_t)^2 - \sum_{n\in t_L}(y_n - \bar y_{t_L})^2 - \sum_{n \in t_R}(y_n - \bar y_{t_R})^2\right]$$

t[L] and t[R} are the left and right descendent subtrees obtained by allowing a split along d. N[t] is the number of samples that descend from the current tree node.

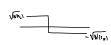
Q: What tree does the drawing above correspond to? How would you modify the drawing to reflect a second split in the left node?

2. The mean decrease in impurity (MDI) for feature d is the weighted mean of these decreases for all splits that involve feature d.

$$MDI_d = \frac{1}{N}\sum_{s:\ \text{split } s \text{ involves } d} N(s)\Delta(s)$$

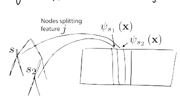
3. From the drawings above, you might already notice that decision trees can be represented as linear regressions onto a basis that's derived from the initial tree estimate. It's not particularly useful for learning the tree (because the basis depends on the tree), but it *is* useful for understanding properties of the model, including variable importances.

$$\psi_s(x) = \frac{1}{\sqrt{N(t_R)N(t_L)}}\left[N(t_R)\mathbb{1}\{x\in t_L\} - N(t_R)\mathbb{1}\{x \in t_R\}\right]$$



4. The fitted values from this regression are exactly the same as the decision tree fits.

$$\hat y = \left[\psi_1(x) \mid \psi_2(x) \mid \ldots \mid \psi_s(x)\right]\hat\beta$$



Further, consider replacing all basis elements that don't involve feature d with their averages.

$$\hat y^d = \left[\bar\psi_1(x) \mid \ldots \mid \bar\psi_{s-1}(x) \mid \underset{\text{uses } d}{\psi_s(x)} \mid \ldots \bar\psi_S(x)\right]\hat\beta$$

The R^2 of this model is exactly the MDI for feature d!!

5. The essential insight of Agarwal et al. (2023) is that we don't have to restrict ourselves to linear regression and R^2. We can use any high-dimensional regression model and any measure of model performance. They call these new measures of feature d's importance MDI+.

6. For example, using ridge regression (still with R^2) leads to more stable variable rankings than ordinary MDI. Further, if we construct regressions like these across many trees in an ensemble, we can derive a new type of random forest (they call it RF+) with analogous feature importance measures.

Importance, Retraining, and Approximations

1. How might we generalize these ideas beyond trees? Suppose that the data is drawn according to X ~ P and that we want to analyze f(X) = E[Y | X]. Also, consider these variants that either ignore or replace feature d:

$$X_{-d} \sim \mathbb{P}_{-d} \qquad \text{(Remove)}$$

$$x^{-d} = (x_1 \ldots x_{d-1}, \mu_d, x_{d+1}, \ldots, x_D) \sim \mathbb{P}^{-d} \qquad \text{(Replace)}$$

$$f_{-d}(x^{-d}) = \mathbb{E}[y \mid x_{-d}] \qquad \text{(Ignore: replace on left and remove from right)}$$

These are defined in Gao et al. (2022). We will also use the notation P^{(-d)} to denote the distribution of X^{(-d)}, though this is not defined in that paper.

2. Then, if V is a measure of model performance, we can define the importance of feature d as:

$$VI_d = V(f, \mathbb{P}) - V(f_{-d}, \mathbb{P}_{-d})$$

In other words, does the model performance deteriorate when we ignore feature d?

3. In reality, we know none of the distributions and conditional expectations needed to compute this variable importance measure. But at least now we can be precise about what to estimate. There are two common ways to approach the problem, and a third interesting compromise introduced by Gao et al. (2022).

4. The *dropout approach* first estimates the performance of a model \hat{f} trained using the entire dataset. Then, it gets new predictions by querying the function on versions of the data with the d^th coordinate replaced by its mean. The two performances are compared. In other words:

$$\widehat{VI}^{DR} = V(\hat f, \hat{\mathbb{P}}) - V(\hat f, \hat{\mathbb{P}}^{-d})$$

This approach is very fast, but it can be misleading (Prop. 3.2) Like in the original variable importance definition for RFs, it can overstate the importance of a variable that is correlated with an important feature, even if conditionally it has no influence on the response.

5. Intuitively, suppose features 1 and 2 are almost perfectly correlated, but that only feature 1 is actually used to generate the response. As long as \hat{beta}[1] + \hat{beta}[2] are close to the true beta[1], then the full model will have good predictions, but the dropout version will perform poorly (especially if \hat{beta}[2] is large). This performance gap leads to high variable importance. If, however, e have a chance to retrain with feature 2 removed, then the estimated \hat{beta}[1] will be close to the true beta[1], and we will have no drop in performance

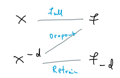
6. Alternatively, in the second step, we could have retrained a new model using a version of the dataset that replaces coordinate d with its mean,

$$\hat f_{-d} = \arg\min_{f\in\mathcal{F}} \frac{1}{N}\sum_{n=1}^{N}\left(y_n - f(x_n^{-d})\right)^2$$

and then evaluate the performance of this new model, which never had access to the real feature d. We can express these steps as:

$$\widehat{VI}^{RT} = V(\hat f, \hat{\mathbb{P}}) - V(\hat f_{-d}, \hat{\mathbb{P}}^{-d})$$

This is called the *retrain approach*. It doesn't suffer from the correlation issue of dropout, but it requires retraining the model D times, which can be intractable.



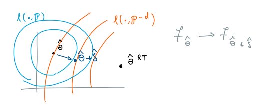
7. The idea of Gao et al. (2022) is to make a one-step update in the direction that full retraining would have taken us. The update will be much cheaper than full retraining, but it will also address the basic issues with dropout. Specifically, if we have a parameterized model, then the model trained using all features corresponds to the estimate:

$$\hat\theta = \arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{n=1}^{N}\left(y_n - f_\theta(x_n)\right)^2$$

The lazy estimation approach is to find one-step updates:

$$\hat\delta = \arg\min_{\delta} \frac{1}{N}\sum_{n=1}^{N}\left(y_n - f_{\hat\theta}(x_n^{-d}) - \delta^T\nabla_\theta f_{\hat\theta}(x_n^{-d})\right)^2 + \lambda\|\delta\|_2^2$$

Notice that the first function evaluation in this expression computes the dropout prediction. Instead of using the dropout prediction directly, though, we use the corrected version:



Comparing the full and one-step models leads to a new estimate of variable importance:

$$VI_d^{LE} = V(\hat f, \hat{\mathbb{P}}) - V(f_{\hat\theta+\hat\delta}, \hat{\mathbb{P}}^{-d})$$

8. This approach works has an especially nice interpretation in the case that f is a neural network. Estimating the one-step update is like fitting a neural tangent kernel (NTK) model, a kind of linearized neural network model. The only difference from NTK is that we use the full model's parameter as our initialization, not a random initialization.

9. The main result of Gao et al. (2022) is a central limit theorem for the estimator VI^{lazy}[d]. By adapting concentration bounds from the NTK literature, they show that these estimators are asymptotically normal with mean VI[d].

$$\sqrt{n}\left(\widehat{VI}_d^{LE} - VI_d\right) \xrightarrow{n\to\infty} N(0, \tau_d)$$

Moreover, a plug-in estimate of the associated covariance yields confidence intervals for the variable importance measure. Of course, the result requires conditions on the data and model class. They are surprisingly simple (only three), but we nonetheless defer these details to the reading.

10. A side note: Lazy estimation has a more heuristic but general analog. Instead of solving the optimization above, we can imagine taking the full model and then "fine-tuning" it on the dropout-transformed data. Taking a few steps is much cheaper than training to convergence, but it can already improve on naive dropout.

Shapley-Based Importance

1. We now have a nice way of estimating VI[d]. But why should we be targeting this quantity in the first place? It has intuitive appeal, but certainly doesn't follow from any first principles. To address this, we can try to connect variable importance to Shapley values. In the same way that ordinary Shapley attribution averages function evaluations over arbitrary subsets of predictors, we will need a version of our conditional expectation that ignores more than just the d^th feature at a time. Specifically, for a subset of features S, define:

$$x^{-S} = (x_S, \mu_{-S})$$

$$f_S(x') = \mathbb{E}[y \mid x_{-S}]$$

f[S] is a submodel that can only use information from features in S.

2. With this notation, we can write the Shapley-based importance introduced by Williamson and Feng (2020):

$$SVI_d = \sum_{S\in[D]\setminus d} \frac{1}{D}\frac{1}{\binom{D-1}{|S|-1}}\left[V(f_{S\cup d}, \mathbb{P}) - V(f_S, \mathbb{P})\right]$$

Like the Shapley values we covered in week 2, this definition satisfies a number of nice properties. For example, it satisfies an additivity property:

$$\sum_{d=1}^{D} SVI_d = V(f, \mathbb{P}) - V(f_\emptyset, \mathbb{P})$$

^ individual feature importances ^ gap between full and null models

3. We will treat SVI[d] as the target of inference. The challenge now is to define a tractable estimator. There are two issues. First, for each d, there are many subsets we have to sum over. Second, for each term within the summation, we potentially have to retrain the model twice, first for f[S] and then f[S U d].

4. The solution is to exactly follow the weighted linear regression perspective that we say in the code example for week 2. Specifically, define an indicator matrix Z that captures feature membership across all subsets S:

$$Z = \begin{array}{c c} & 1 \ \ldots \ D \\ 2^D & \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \end{array}$$

Also define a diagonal weight matrix that reflects the number of active features in each subset:

$$W = \mathrm{diag}\left(1 \Big/ \binom{D-2}{|S|-1}\right)$$

Concatenate the performances of all submodels:

$$\underline{V} = \underbrace{(V_\emptyset, V_1, \ldots V_{[1:D-1]}, V_{1:D})}_{2^D \text{ terms}}$$

Then, SVI[d] across all D features can be obtained by solving the weighted regression:

$$SVI = \arg\min_{\beta\in\mathbb{R}^D}\left\|W^{1/2}(\underline{V} - Z\beta)\right\|_2^2$$

5. We can approximate this ideal by solving the same regression with just a subsample of rows (they also introduce a strict constraint related to additivity, but conceptually I don't see much of a difference). We will still have to retrain a model for each of the subsets that we sample, but this can still be a big computational savings. To be precise, we write our final estimator as:

$$\widehat{SVI}_m = \arg\min_{\beta\in\mathbb{R}^D}\left\|W_m^{1/2}(\underline{V}_m - Z_m\beta)\right\|_2^2$$

Where each W[m], V[m], and Z[m] are analogs of W, V, and Z computed using m random subsets of [1, ..., D].

6. An especially interesting result is that these SVI estimates satisfy a central limit theorem, analogous to the one we saw before for lazy retraining. Specifically, consider the limiting regime where both the number of samples n and sampled feature sets m(n) grows. Then, under appropriate smoothness conditions,

$$\sqrt{n}\left(\widehat{SVI}_{m(n)} - SVI\right) \xrightarrow{n\to\infty} N(0, \Sigma)$$

The covariance can be written in closed form as a function of Z, W, and the derivative of V, and so it's estimable in practice. This allows us to construct confidence intervals for \hat(SVI), which can be helpful for appropriately interpreting the final variable importance measures.