alessia_project

Alessia

2023-06-06

BAGGING PREDICTORS, LEO BRAIMAN

WHY BAGGING WORKS

Numeric Prediction

Instability is important in bootstrap aggregating because it helps improve the performance of the ensemble model by reducing variance and increasing accuracy. Bagging is a technique that combines multiple models (often decision trees) trained on different bootstrap samples of the dataset to make predictions. The key idea behind bagging is to introduce diversity among the base models by using different subsets of the training data. This diversity is achieved through the bootstrap sampling process, where each base model is trained on a randomly selected subset of the original data, allowing for variations in the training sets. If the base models used in bagging are stable, meaning they produce similar predictions when trained on slightly different datasets, the ensemble model's performance may not significantly improve. This is because the models will produce similar predictions and will not provide enough diversity to effectively reduce variance. On the other hand, if the base models are unstable, meaning they are sensitive to small changes in the training data, the ensemble model will benefit from the aggregation of these diverse models. The variations among the models can help to capture different aspects of the data and lead to improved generalization and prediction accuracy. Therefore, instability, or the ability of the base models to exhibit different behaviors with different training data, is important in bagging as it contributes to increased diversity and better performance of the ensemble model.

Let each (y,**x**) case in L be independently drawn from the probability distribution P. Suppose y is numerical and $\Phi(\mathbf{x}, L)$ the predictor. Then the aggregated predictor is the average over L of $\Phi(\mathbf{x}, L)$, i.e. $\Phi_A(\mathbf{x}) = E_L[\Phi(\mathbf{x}, L)]$. Take x to be a fixed input value and y an output value. Applying the inequality $E(Z^2) \geq (E(Z))^2$ then $E_L[(y - \Phi(\mathbf{x}, L)^2] \geq (y - \Phi_A(x))^2$. Integrating both sides over the joint y,**x** output-input distribution, we get that the mean_squared error of $\Phi_A(\mathbf{x})$ is lower than the mean-squared error averaged over L of $\Phi(\mathbf{x}, L)$. How much lower depends on how unequal the two sides of $[E_L\delta(\mathbf{x}, L)]^2 \leq E_L\delta^2(\mathbf{x}, L)$ are. The effect of instability is clear. If $\delta(\mathbf{x}, L)$ does not change too much with replicate L the two sides will be nearly equal, and aggregation with not help. The more highly variable the $\delta(\mathbf{x}, L)$ are, the more improvement aggregation may produce. But δ_A always improves on δ. Now Φ_A depends not only on **x** but also the underlying probability distribution P from which L is drawn, i.e. Φ_A= $\Phi_A(\mathbf{x}, P)$. But the bagged estimate is not $\delta_A(\mathbf{x}, P)$ but rather $\delta_B(X) = \delta_A(\mathbf{x}, P_L)$ where P_L is the distribution that concentrates 1/N at each point $(y_n, \mathbf{x}_n) \in L$. Then if the procedure in unstable, δ_B can give improvement through aggregation. On the other side, if the procedure is stable, then δ_B will not be as accurate for data drawn from P as $\delta_A(\mathbf{x}, P)$. There is a cross-over point between instability and stability at which δ_B stops improving on $\delta_A(\mathbf{x}, P)$ and does worse.

Classification

In classification, a predictor $\delta(\mathbf{x}, L)$ predicts a class label j $\in$ {1,. . . ,J}. Denote Q(J|**x**)= P($\Phi(\mathbf{x}, L) = j$). The interpretation of Q(J|**x**) is this: over many independent replicates of the learning set L, Φ predicts class label j at input **x** with relative frequency Q(J|**x**). Let $P(J|\mathbf{x})$ be the probability that input **x** generates class j. Then the probability that the predictors classifies the generated state at $\mathbf{x}$ correctly is $\sum_j Q(J|\mathbf{x})P(J|\mathbf{x})$. The overall probability of correct classification is r= $\int[Q(J|\mathbf{x})P(J|\mathbf{x})]P_x(d_x)$ where $P_x(d_x)$ is the $\mathbf{x}$ probability distribution. Note that for any $Q(j|\mathbf{x})$, $\sum_j Q(J|\mathbf{x})P(J|\mathbf{x}) \leq \max\_jP(J|\mathbf{x})$ with equality only if

$$\begin{cases} 1 \quad if \quad P(j|\mathbf{x}) = max_i P(i|\mathbf{x})) \\ O \quad else \end{cases}$$

The predictor $\Phi^*(\mathbf{x})$= $argmax_j P(J|\mathbf{x})$ leads to the above expression for $Q(J|\mathbf{x})$ and gives the highest attainable correct classification rate: r^*=$\int max_j P(j|\mathbf{x})P_X()$. Call Φ order-correct at the input $\mathbf{x}$ if $argmax_j Q(j|\mathbf{x}) = argmax_j P(j|\mathbf{x})$. This means that if input $\mathbf{x}$ results in class j more often than any other class, then Φ also predicts class j at $\mathbf{x}$ more often than any other class. The aggregated predictor is: $\Phi_A(\mathbf{x}) = argmax_j Q(j|(x))$. For the aggregated predictor the probability of correct classification at $\mathbf{x}$ is $\sum_j I(argmax_i Q(i|\mathbf{x}) = j)P(j|\mathbf{x})$. If Φ is order_correct at $\mathbf{x}$, then the previous formula equals $max_j P(j|\mathbf{x})$. Letting C be the set of all inputs $\mathbf{x}$ at which Φ is order-correct, we get for the correct classification probability of Φ_A the expression:

$$r_A = \int_{\mathbf{x}\in\mathbf{C}} max_j P(J|\mathbf{x})P_x(d\mathbf{x}) + \int_{\mathbf{x}\in\mathbf{C}'}[\sum_j I(\Phi_A(\mathbf{x}) = j)P(j|\mathbf{x})]P_x(\mathbf{x}))]$$

Even if Φ is order-correct at $\mathbf{x}$ its correct classification rate can be far from optimal. But Φ_A is optimal. If a predictor is good in the sense that it is order-correct for most inputs $\mathbf{x}$, then aggregation can transform it into a nearly optimal predictor. On the other hand, unlike the numerical prediction situation, poor predictors can be transformed into worse ones. The same behavior regarding stability holds. Bagging unstable classifiers usually improves them. Bagging stable classifiers is not a good idea.

Using the learning set as a test set

In bagging trees, the training set L_B is generated by sampling from the distribution P_L. Using L_B a large tree T is constructed. The standard CART methodology find the sequence of minimum cost pruned subtrees of T. The best pruned subtree in this sequence is selected either using cross-validation or a test set. The idea of a test is that it is formed by independently sampling from the same underlying distribution that gave rise to the learning set. In the present context, a test set is formed by independently sampling from P_L, i.e. we can get a test set by sampling with replacement from the original learning set L. Infact, consider sampling with replacement a large number of times from L, each case $(y_n, \mathbf{x_n})$ will be selected about the same number of time as any other case. Thus, using a very large test set sampled from P_L is equivalent to just using L as a test set.

A LINEAR REGRESSION ILLUSTRATION

Forward variable selection

With data of the from $L = (y_n, \mathbf{x_n}), n = 1...N$ where $\mathbf{x} = (x_1, ..., x_M)$ consists of M predictor variables, a popular prediction method consists of forming predictors $\delta_1(\mathbf{x})...\delta_M(\mathbf{x})$ where $\delta_m(\mathbf{x})$ is linear in $\mathbf{x}$ and depends on only m of the M x-variables. Then one of the $\delta_m(\mathbf{x})$ is chosen as the designated predictor. A common method for constructing the $\delta_m(\mathbf{x})$, and one that is used in our simulation, is forward variable entry. If the variables used in $\delta_k(\mathbf{x})$ are $(x_{1k}, ..., x_{mk})$ then for each m $\notin m_1, ...m_K$ form the linear regression of y on $(x\_m1, ..., x\_mk + 1, x_m)$, compute the RSS(m) and take x_{mk+1} such that m_{k+1} minimizes RSS(m) and $\delta_k + 1(\mathbf{x})$ the linear regression based on $(x_{m1}, ..., x_{mk+1})$. What is clear about this procedure is that it is an unstable procedures. The variables are competing for inclusion in the δ_m and small changes in the data can cause large changes in the δ_m. Forward variable selection is considered an unstable procedure due to the following reasons:

1. Dependency on the order of variable selection: in forward variable selection, the procedure starts with an empty model and adds one variable at a time based on certain selection criteria. However, the initial selection of variables can significantly impact the final result. If the initial set of selected variables differs, the progressive selection of variables can lead to different models. Therefore, the selection of initial variables can cause instability in the results.

2. Sensitivity to correlated variables: forward variable selection may be influenced by correlated variables. When adding a variable to the model, it may appear to provide significant informative value, but it could be strongly correlated with other variables already present in the model. This correlation can lead to an overestimation of the importance of the added variable and result in selecting a suboptimal model.

3. Impact of small data variations: forward variable selection can be sensitive to small variations in the input data. Even a slight modification of the input data can lead to a completely different variable selection or a different model. This instability can make it challenging to reproduce the results.

To mitigate the potential instability of forward variable selection, techniques such as cross-validation or bootstrap can be used to assess the stability of selected variables and the robustness of the resulting model. These techniques allow for a more accurate estimation of model performance and more reliable identification of the most relevant variables.

Simulation structure

The simulated data used in this section are drawn from the model y= $\sum_m \beta_m x_m + \epsilon$ where ϵ is N(0,1). The number of variables M=30 and the sample size is 60. The subset selection is nearly optimal is there are only a few large non-zero β_m , and its performance is poor if there are many small but non-zero β_m. Subset selection methods can indeed perform well when there are only a few large non-zero coefficients. This is why subset selection can be effective in such cases:

1. Focus on important predictors: subset selection methods aim to identify a subset of predictors that are most relevant for predicting the response variable. When there are only a few predictors with large non-zero coefficients, subset selection can successfully identify and include those important predictors in the model.

2. Improved model interpretability: by selecting a subset of predictors with large non-zero coefficients, subset selection can lead to a more interpretable model. The resulting model will focus on the most influential predictors, providing insights into the relationship between these predictors and the response variable.

3. Reduction in model complexity: by selecting a smaller subset of predictors, subset selection reduces the complexity of the model compared to including all predictors. This reduction in complexity can improve model generalization, making the model more robust and less prone to overfitting.

It's important to note that subset selection methods, like forward selection or backward elimination, are not guaranteed to find the exact optimal model in all cases. They are heuristic approaches that explore different combinations of predictors. The optimality of the selected model depends on factors such as the quality of the selection criteria and the sample size. In practice, it's always recommended to evaluate the performance of the selected model using cross-validation or other validation techniques like bootstrap. This helps to assess the model's predictive accuracy and determine if additional refinement or regularization techniques are needed. In summary, when there are only a few large non-zero coefficients, subset selection methods can be effective in identifying the most important predictors and constructing a simpler, interpretable model. However, it's crucial to validate the model and consider other factors to ensure its performance and reliability.

Three sets of coefficients are used. In the first set of coefficients there are only three non-zero β_m. In the second set of coefficients there are 15 non-zero β_m. In the third set there are 27 non-zero β_m, all relatively small. For each set of coefficients, the following procedure was replicated 250 times: 1. Data L=$L = (y_n, \mathbf{x_n}), n = 1...60$ was drawn from the model y=$\sum_m \beta_m x_m + \epsilon$. 2. Forward entry variables was done using L to get the predictors $\delta_1(\mathbf{x})...\delta_M(\mathbf{x})$. The mean_squared prediction error of each of these was computed giving $e_1...e_M$. 3. Fifty bootstrap replicates L_B of L were generated. For each of these, forward step-wise regression was applied to construct predictors $\delta_1(\mathbf{x}, L_B)...\delta_M(\mathbf{x}, L_B)$. These were averaged over the L_b to give the bagged sequence $\delta_1^B(\mathbf{X})...\delta_M^B(\mathbf{X})$. The prediction errors $e_1^B...e_M^B$ for this sequence were computed.

These computed mean-squared-errors were averaged over the 250 repetitions to give the sequences $\bar{e}_m^S, \bar{e}_m^B$. For each set of coefficients, these two sequences are plotted vs m in Figure1 a,b,c.

Discussion of simulation results

Looking at Figures1 a,b,c an obvious result is that the most accurate bagged predictor is at least as good as the most accurate subset predictor. In the first set of coefficients (with only three non-zero β_m), subset selection is nearly optimal and there is no improvement. In the second and third set of coefficients there is substantial improvement. *Bagging can improve only if the unbagged is not optimal.* Note that in all three graphs there is a point past which the bagged predictors have larger prediction error than the unbagged. The explenation is this: linear regression using all variables is a fairly stable procedure. The stability decreases as the number of variable used in the predictor decreases.*For a stable procedure $\delta_B = \delta_A(\mathbf{x}, P_L)$ is not as accurate as $\delta(\mathbf{x}, P)$.* As m increases, the instability increases and there is a cross-over point at which δ_m^B becomes more accurate than δ_m.

CONCLUDING REMARKS

Bagging class probability estimates