![Pomona College logo]

SENIOR THESIS IN MATHEMATICS

Local Prediction Confidence for Classification Random Forests

Author:
Xiaotong Gui

Advisor:
Dr. Johanna Hardin

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

September 20, 2019

Acknowledgements

Contents

Chapter 1

Introduction

Random forests are an ensemble learning model widely used in classification and regression tasks. They gained popularity within the machine learning community due to their robust performance on high dimensional data (Hastie et al., 2009). Through producing de-correlated trees and leveraging an aggregated decision, random forests are able to greatly reduce variance of models describing data with correlated explanatory variables.

While random forests have great predictive power, their inference mechanism in classification tasks are not well-understood. Unlike in regression setting where construction of prediction intervals is well-defined, the prediction confidence of a new observation in random forests is often approximated by the overall predictive performance over previous data. Sensitivity analysis gives us the extra ability to break down prediction accuracy by classes, but metrics such as AUC score still does not speak to individual sample variability. Shafer and Vovk (2008) introduced conformal prediction as a measure of how unsual the sample is from the training data, and Bhattacharyya (2013) further applied the method in random forests. However, conformal prediction is computationally inefficient as it requires running multiple k nearest neighbor procedures on the test data, making applications on big data inaccessible.

Despite more recent literature on constructing confidence intervals on regression forests, few are directly on classification random forests. Wager et al. (2014) created confidence intervals in regression random forests using sub-sampling method to approximate standard error. Lu and Hardin (2018) in Lu's senior thesis explored prediction intervals in regression forests. However, the methods proposed in regression literature are immensely useful for

our study of prediction confidence in classification forests. Both Wager et al. (2014) and Lu and Hardin (2018) introduced a proximity measure that evaluates the similarity between the new observation and training data using properties of the bootstrapped trees in random forests. We further include this idea into our construction of classification confidence.

We propose a local confidence metric that is a direct improvement to existing metrics that are widely used in classification tasks. In Chapter 2- 3, we present prediction intervals in linear regression and the random forests machinery. Chapter 4 reviews current existing confidence metrics and discuss their shortcomings. In Chapter 5, we propose a local confidence metric based on a similarity measure between the new observation and the training sample. In Chapter 6, we show that our purposed method is computationally efficient, close to the true confidence and can better capture the heterogeneity of the feature space. To support that claim, we carefully define prediction confidence in classification random forests and empirically compare our proposed metric with other confidence metrics. The overall goal of this thesis is to construct a confidence metric that is relatively easy to understand, fast to implement and fairly unbiased.

Chapter 2

Confidence and Prediction Intervals

Confidence and prediction intervals are well-defined inference mechanisms for the well-studied linear regression. In this chapter, we will focus on the frequentist definitions and examine the theoretical foundation. The theoretical setup is based on James et al. (2014) and Neter et al. (1996).

Confidence intervals for population parameters are constructed at a select level chosen by user. It means that if we repeatedly take samples from the population, then a certain proportion of the intervals constructed will contain the true population parameter.

Definition 2.0.1 *Let $\alpha \in [0, 1]$. A $(1 - \alpha)100\%$ confidence interval for a population parameter is an interval created by a procedure such that among the intervals constructed on repeated samples taken from the population, $(1 - \alpha)100\%$ of them contain the true parameter.*

A confidence interval gives a range of estimated value for a population parameter, whereas a prediction interval estimates a range of values for the individual response of a new observation. The notion of taking repeated samples also holds true in prediction intervals.

Definition 2.0.2 *Let $\alpha \in [0, 1]$. A $(1-\alpha)100\%$ confidence interval for a new response is an interval created by a procedure such that among the intervals constructed on repeated samples taken from the population, $(1 - \alpha)100\%$ of them contain the true response of the new observation.*

2.1 Example: Linear Regression

2.1.1 Simple Linear Regression

We will start building the interval concepts from simple linear regression. We assume the normal error regression model:

$$Y_i = \beta_0 + \beta_1 x_i + \epsilon_i \ \ for \ i = 1, ...n \tag{2.1}$$

where
Y_i is the response of the ith observation.
β_0 and β_1 are regression parameters
x_i is the value of the explanatory variable of the ith observation
$\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$

2.1.2 Interval Estimation for $E[Y_h]$

A common objective in regression inference is to estimate the mean of Y at a certain level of x. Let x_h denote the level of x for which we wish to estimate the mean response. The mean response when $x = x_h$ is denoted by $E[Y_h]$. The estimate of $E[Y_h]$ is given by:

$$\hat{y}_h = \hat{\beta}_0 + \hat{\beta}_1 x_h \tag{2.2}$$

The sampling distribution of $\hat{y}_h$ is normal with mean and variance:

$$E[\hat{y}_h] = E[Y_h] \tag{2.3}$$

$$\sigma^2(\hat{y}_h) = \sigma^2 \left(\frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right) \tag{2.4}$$

Unknown σ^2 is estimated using $s^2(\hat{y}_h)$, the estimated variance of $\hat{y}_h$

$$s^2(\hat{y}_h) = MSE \left(\frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right) \tag{2.5}$$

where

$$MSE = \frac{1}{n} \sqrt{\sum_{i=1}^n (y_i - \hat{y}_i)^2} \tag{2.6}$$

The $(1 - \alpha)100\%$ confidence interval for the mean response of Y_h at x_h is then

$$\hat{y}_h \pm t_{(\alpha/2, n-2)} \sqrt{MSE \left(\frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum (x_i - \bar{x})^2} \right)} \tag{2.7}$$

2.1.3 Interval Estimation for a New Response

A prediction interval gives a range of plausible values for individual responses as opposed to a population mean. Therefore it includes the variance of the data at hand. Using the same set up from the previous section, the $(1 - \alpha)\%$ prediction interval for the response of a new observation at x_h is:

$$\hat{y}_{h(new)} = \hat{y}_h \pm t_{(1-\alpha/2;n-2)} s(\hat{y}_{h(new)}) \tag{2.8}$$

where an unbiased estimator of $\sigma^2(\hat{y}_{h(new)})$ is

$$s^2(\hat{y}_{h(new)}) = MSE + s^2(\hat{y}_h) \tag{2.9}$$

$$= MSE\left(1 + \frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum(x_i - \bar{x})^2}\right) \tag{2.10}$$

Therefore the $(1 - \alpha)100\%$ prediction interval for a new response at x_h is

$$\hat{y}_h \pm t_{(\alpha/2,n-2)}\sqrt{MSE\left(1 + \frac{1}{n} + \frac{(x_h - \bar{x})^2}{\sum(x_i - \bar{x})^2}\right)} \tag{2.11}$$

Frequentist confidence and prediction intervals are conceptualized from taking numerous samples from the population. Later we will use a similar idea to define prediction confidence in classification random forests. Since we want a confidence score on a new response rather than population mean, our construction of prediction confidence in classification random forests echoes that of prediction intervals in linear regression.

Chapter 3

Random Forests

Random forests consist of decision trees grown on bootstrap samples drawn from the same data set using a subset of randomly selected explanatory variables at each split. In this section, we will examine classification random forests and identify their unique properties that enable us to create a local confidence metric in the following chapters. Decision trees, bootstrap sampling methods, and bagging will be explained.

3.1 Decision Tree

Classification trees make predictions by partitioning the feature space into non-overlapping regions and assigning labels given by the most occurring class in a specified region. The best split is made by minimizing a criteria function. This top-down, greedy approach is known as recursive binary splitting (James et al., 2014).

The splitting criteria, formally defined as a loss function, is a measure of of how well the split differentiates the observations in a region. Classification trees use the Gini index to evaluate each split.

Definition 3.1.1 *Let $z_1, ... z_n$ be a set of n training observations each with p explanatory variables and a categorical response variable with K levels. For the Rth region of the predictor space, the **Gini index** is given by*

$$G_R = \sum_i^K \hat{p}_{Rk}(1 - \hat{p}_{Rk})$$

where $\hat{p}_{Rk}$ represents the proportion of training observations in the Rth region that are from the kth class.

Gini index is a measure of total variance across the K classes. In other words, it measures nodes purity and a small value indicates that the node is dominated by observations from the same class.

3.1.1 Tree Growth

Algorithm 1 Grow a Classification Tree

---

procedure SPLIT(set Z of observations with p predictors $x_1, x_2, ..., x_p$, region R of the predictor space, int minNodeSize)

    if $|Z|$ <minNodeSize **then return** R

    end if

    for j in 1,2,...p **do**

        Define $R_1(j, s) = \{x | x_j \in s\}$ and $R_2(j, s) = \{x | x_j \notin s\}$

        where s is a subset of values x_j can take

    end for

    Identify (j', s') among all (j,s) that minimizes $G(R_1(j, s)) + G(R_2(j, s))$

    split($\{z \in Z : x \in R_1\}, R_1(j', s'), minNodeSize$)

    split($\{z \in Z : x \in R_2\}, R_2(j', s'), minNodeSize$)

end procedure

Definition 3.1.2 *Let T be a classification tree. A **node** of T is a region created by each split in the tree construction algorithm. A node can be further divided in the next recursive step.*

Definition 3.1.3 *Let T be a classification tree. A **terminal node** of T is the region returned by the tree growth algorithm. A terminal node cannot be further split and determines the final prediction of a new observation.*
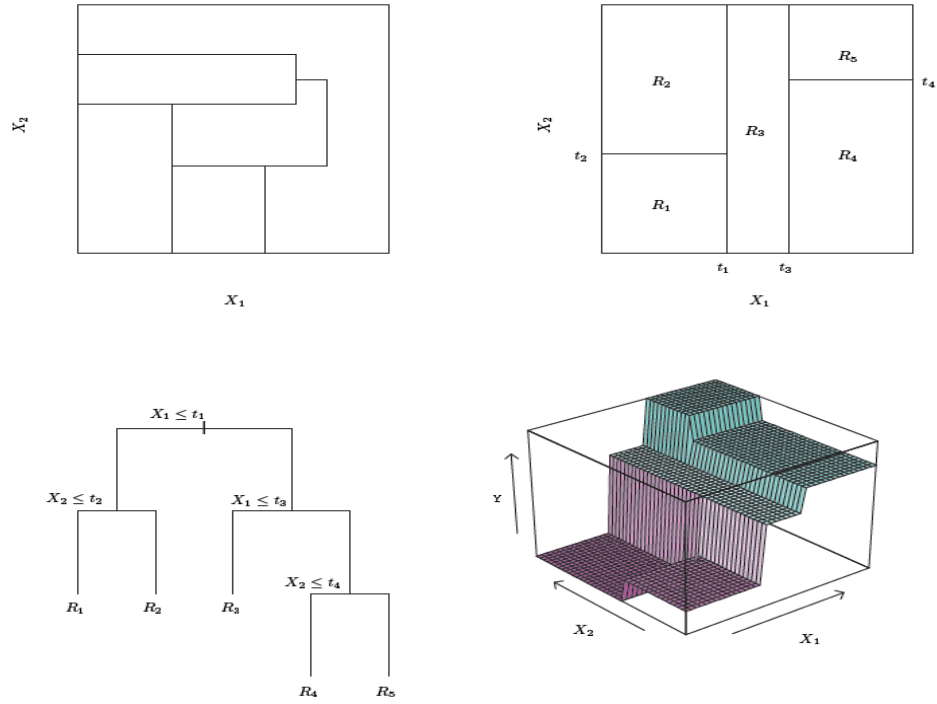
Figure 3.1: A visualization of recursive binary splitting and tree growth. Figure from ISLR (James et al., 2014)

3.1.2 Tree Prediction

After growing a tree based on the training samples, we assign labels to the terminal nodes based on the most occurring classes of the observations in the terminal nodes. The final prediction of a new observation is decided by the assigned label of the terminal node the observation falls into.

There are many great things about using decision trees in classification tasks such as easy interpretability and computation efficiency. However, trees are often non-robust and have large variance from sample to sample. In the next section, we will introduce random forests, a much more powerful predictive model that use trees as building blocks.

3.2 Random Forests

3.2.1 Bootstrapping and Bagging

Bootstrapping is widely used in statistics to approximate the unknown sampling distribution of a statistic. In the context of decision trees, the bootstrap can be particularly useful to improve the predictions themselves. The procedure of bootstrap is: take repeated samples which are the same size as the data with replacement. The underlying assumption is the original data are an accurate representation of the population and the bootstrap procedure allows us to emulate the process of obtaining new sample sets from the population. Through constructing bootstrap samples and averaging or majority vote on the predictions built on each sample, we can greatly reduce the variability of our predictions from different training samples. This technique is called bagging.
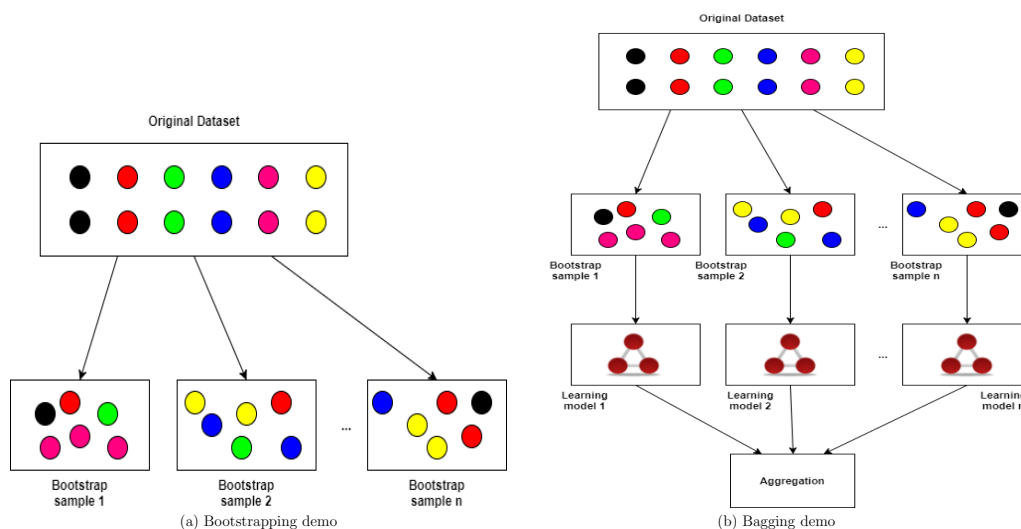

(a) Bootstrapping demo   (b) Bagging demo

Figure 3.2: An illustration of bootstrapping and bagging. This figures come from this tutorial on towardsdatascience.com

.

3.3　Constructing Random Forests

The procedure of constructing a random forest follows from bagging. The only difference is, instead of looking through the entire predictor space, for each tree at each split, we only evaluate a randomly chosen subset of predictors to evaluate the splitting criteria. The procedure for random forest prediction is as follow:

1. Bootstrap B samples from the data $z_1, ...z_n$.

2. Grow classification trees $T_1, T_2, ...T_B$ where T_b is constructed on the bth bootstrap sample with m randomly chosen predictors $(0 < m < p)$ considered at each split.

3. For a new observation, make predictions using each of the B trees. The final prediction is the majority class of all B predictions from B trees.

Since at each split only a random subset of the predictors are considered, the bagged trees will not be dominated by a single strong predictor. In such random forests are able to generate de-correlated trees, making the aggregated prediction less variable and more robust.

3.3.1　Out-of-Bag Estimation

Each bootstrap sample is sampled with replacement from the original data, thus some observations from the original data might appear in the bootstrap sample while some might not. This property enables us to directly estimate random forests' predictive performance on new data. We will lay out the following definitions.

Definition 3.3.1 *Let φ be a random forest with B trees $T_1, ..., T_B$. For the bth decision tree T_b, a training observation z_i is **out-of-bag** if z_i is not in the bth bootstrap sample.*

Proposition 3.3.1 *On average, each bagged tree makes use of around 2/3 of the training observations.*

The proof to Proposition 3.3.1 is fairly straightforward. $Pr(z_i$ not in a bootstrap sample) $= (1 - \frac{1}{n})^n$. As $n \to \infty$, the probability converges to $\frac{1}{e} \approx \frac{1}{3}$. Therefore for each bagged tree, around 1/3 of the training data are out-of-bag.

Definition 3.3.2 *The out-of-bag prediction of a training observation z_i is the majority vote over all predictions of z_i by trees where z_i is out-of-bag. Each training observation yields around $B/3$ such predictions toward the final out-of-bag prediction.*

We can use the out-of-bag prediction accuracy for all training data to estimate the predictive performance of the model on new observations since the out-of-bag prediction of each training sample is yielded only on trees that were not used to fit the sample. We will revisit out-of-bag prediction error estimation in the next chapter in our discussion of prediction confidence. We will also use the out-of-bag concept to construct local confidence in Chapter 5.

Chapter 4

Prediction Confidence in Machine Learning Models

In classification tasks, in addition to the actual predictions, we are also interested in knowing how confident the model is in making the individual predictions. In other words, how likely is the model to be correct in predicting a new observation? Having an estimate of prediction confidence is particularly crucial in many real-world situations where classification results drive high-stake decision-making.

This chapter will discuss existing metrics that are commonly used in the machine learning community to estimate prediction confidence and introduce the motivation behind our novel proposed method.

4.1 Overall Training Accuracy

The most common approach to quantify the learning model's predictive ability is to look at the training accuracy.

Definition 4.1.1 *Let $z_1, ...z_n$ be the set of training data. Let $y_1, ...y_n$ be their true labels, and $\hat{y}_1, ...\hat{y}_n$ be the predicted labels. The training accuracy of the model is*

$$Acc_{training} = \frac{\sum_{i=1}^{n} \mathbb{I}(\hat{y}_i = y_i)}{n} \tag{4.1}$$

However, assessing a model's prediction capacity on future data based on the data used to fit the model is problematic. Since the model is tuned

to fit the training data closely, its prediction accuracy of new data will be overestimated by the training accuracy.

4.2 Test Accuracy

Since we are interested in the model's performance on unseen data, a better idea for assessing accuracy is to create a test data set independent of model fitting. If we randomly sub-sample a part of our data and make predictions after the model is fitted, the prediction accuracy on the test data set will approximate that on future new observations, assuming that the test data is representative of the population.

Definition 4.2.1 *Let $z_1*, ...z_m*$ be the set of test data. Let $y_1*, ...y_m*$ be their true labels, and $\hat{y_1}*, ...\hat{y_m}*$ be the predicted labels. The test accuracy of the model is*

$$Acc_{test} = \frac{\sum_{i=1}^{m} \mathbb{I}(\hat{y_i}* = y_i*)}{m} \tag{4.2}$$

However, splitting data into train and test will result in fewer data for the training set, making the training set less representative. This problem will become especially prominent with a small sample size.

4.3 Out-of-bag Training Accuracy

In random forests, instead of a train-test split we can also assess prediction accuracy on new data with training observations only using out-of-bag predictions. Since each bootstrap sample uses approximately 2/3 of the data, and each training observation is not included in the building of around 1/3 trees, we can conveniently use the training set to assess our model's prediction accuracy on new data. We define this metric out-of-bag training accuracy.

Recall the definition of out-of-bag predictions from 3.3.2.

Definition 4.3.1 *Let $\hat{y}_{-1}, ...\hat{y}_{-n}$ be the out-of-bag predictions for training observations $z_1, ...z_n$. The **out-of-bag training accuracy** is*

$$Acc_{oob} = \frac{\sum_{i=1}^{n} \mathbb{1}(\hat{y}_{-i} = y_i)}{n} \tag{4.3}$$

A shared problem with the confidence metrics we have discussed so far is none of them directly considers the prediction confidence of individual new observations. The underlying concept behind training, test and out-of-bag accuracy is we can approximate the prediction accuracy of a set of future data using accuracy of a set of known data. However, the model's ability to predict observations in different feature spaces could be immensely different.

We will illustrate our point with an example. Suppose our data consists of two clusters, with one mixed and one homogeneous classes. Our learning models will perform poorly on the mixed group, resulting in around 50% accuracy. However, the model will predict almost perfectly on the homogeneous cluster. Therefore, although the overall accuracy is 75%, that number itself is not representative of the true prediction accuracy of the individual samples in either of the two different clusters.
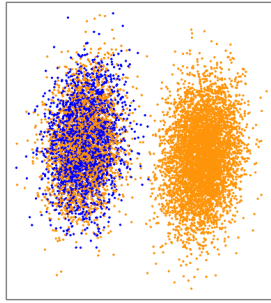
Figure 4.1: An example of when an overall accuracy metric falls short

Therefore, we are motivated to find a prediction confidence metric of individual new data, drawing parallels to the idea of constructing prediction intervals in linear regression.

Interestingly, the aggregate nature of random forests allows us to obtain a confidence estimation of individual observations.

4.4   Class Probabilities

Since random forests are an ensemble learner, a common approach to estimate the prediction confidence of a new observation is to use the proportion of trees that voted the final prediction class. This proportion is also interpreted as the the probability of the prediction being in the predicted class.

Definition 4.4.1 *Let φ be a Random Forests with B trees $T_1, ..., T_B$ and our data has a categorical response of K levels. For a new observation z_h, if the predicted response is $k \in \{1, ..., K\}$, then the probability that z_h has a response of k is*

$$p_h = \frac{1}{B} \sum_{b=1}^{B} \mathbb{1}(\hat{y}_b = k) \tag{4.4}$$

where $\hat{y}_b$ is the predicted response from the bth tree. We also refer to this probability as class probability.

The class probability p_h is 1 if the predicted responses by all trees are k, meaning that the model is very confident in its prediction of z_h. By the majority vote rule and pigeonhole principle, p_h must not be smaller than $\frac{1}{k}$. The smaller p_h is, the less confident we are about the prediction.

However, in calculating class probability, we are considering all trees including those built on bootstrap samples that are irrelevant or distracting to the new observation to be predicted. Having predictions from those trees could add noise to our estimate of prediction confidence.

To conclude, all the metrics examined in this chapter use a global approach to estimate the prediction confidence of a new observation. That is, the metrics indiscriminately use the model's ability to predict all observations to estimate the confidence of one new observation. We argue that a global metric is not necessary, and if we only look at observations locally, we can have a direct improvement on our confidence metric.

Chapter 5

Local Prediction Confidence

Following the previous chapter, we propose a local prediction confidence metric for individual observations. We define local prediction confidence in a frequentist fashion, as the percentage of the random forests models, among repeated samples, that predict the response class correctly given a specific feature space. We approximate the local prediction confidence of a new observation with the model's prediction accuracy on other training observations, weighted by the similarity between the training observations and the new observation . This locally weighted approach has been adopted in optimizing forests building in the context of generalized random forests (Athey et al., 2016). It is theoretically reasonable to use the same approach to estimate prediction confidence to filter noises irrelevant to individual observations.

5.1   Define Local Confidence

In Lu and Hardin (2018), a prediction confidence interval is constructed with locally weighted error. We will adopt a similar idea in our construction of a local confidence metric.

Definition 5.1.1 *A training observation is an **out-of-bag cohabitant** with the new observation in a tree if*

- *the training observation is out-of-bag in this tree*

- *the training observation ends up in the same terminal node with the new observation*

The oob-cohabitance frequency can be used as a relevance measure of a training observation to the prediction of the new observation (Athey et al., 2016). We believe that the more frequent the training observation out-of-bag-cohabits with the new observation, the more likely they come from the same class label. Therefore, more weights should be given to random forests' predictive accuracy on the more relevant training observations.

Now let's define our local confidence score.

Definition 5.1.2 *Let $z_1, ... z_n$ be the training observations. Let z_h be the new observation we want to predict. Let w_i be the number of times a training observation z_i out-of-bag-cohabits with z_h.*

Let $\hat{y}_{-i}$ be the out of bag prediction for z_i. Then

$$local\_confidence(z_h) = \frac{1}{\sum w_i} \sum_{i=1}^{n} w_i \mathbb{1}(\hat{y}_{(-i)} = y_i) \qquad (5.1)$$

The intuition behind our construction is that we will estimate the prediction confidence of a new observation with the oob prediction accuracy of local training samples. If a training observation, when it is out-of-bag, never falls in the same terminal node with the new observation, then its out-of-bag prediction accuracy will not be considered in the local confidence score. In this way, our metric is able to exclude the predictive performance of training observations that come from a different class. On the other hand, the more frequent a training observation out-of-bag cohabits with the new observation, the more likely they come from the same class.

A local confidence metric has an advantage over a global metric when data exhibits significant heterogeneity, or the model's predictive power varies greatly in different feature space. The following chapter will create multiple scenarios and use simulations to compare different metrics.

17

Chapter 6

Evaluation

This chapter will introduce simulation methods to compare different prediction confidence metrics. We will create scenarios where some metrics outperforms the others and the other way around.

6.1 Approximate True Prediction Confidence

The first step in comparing metrics is creating a benchmark. In other words, how can we know the "true" prediction confidence of a new observation?

We begin with simulating a population with designed characteristics, and repeatedly drawing smaller sub-samples to make predictions on the new observation. The approximated "true" prediction confidence will be the proportion of correct predictions among the repeated samples.

In real-life research, scientists most likely get only one data sample. A confidence metric is obtained from that one particular sample, and there is no way to verify if it is biased or not. However, through simulating a population and repeatedly taking sub-samples, our method is able to assess the random forests' ability to predict the new observation based on different training samples. Ideally, if we have the ability to collect all possible samples from the population, then the prediction accuracy of the models trained by all samples represents the random forests' confidence of predicting the new observation. This construction is extremely similar to the definition of confidence intervals in linear regression.

We define the proportion of correct predictions among all possible training data the "true" prediction confidence, and claim that the result generated

by our algorithm well approximates the true confidence.

Algorithm 2 Simulate True Prediction Confidence

---

procedure TRUECONFIDENCE(A population of size N, sample size n, new observation z_h)

Simulate a population of size N and initialize count = 0

for j in 1,2,...num_iterations **do**

Take a sample of size n from the population without replacement.

Train a random forest on the sample.

Use the random forest to make a prediction on z_h. count + 1 if the prediction is correct.

end for

Return $true\_confidednce$ = count/num_iterations

end procedure

We used Ranger (Wright and Ziegler, 2017), an R package for fast implementation of Random Forests, to train the models. All the random forests are tuned with default values specified by Ranger, so that the model parameters are controlled across all training samples.

We empirically evaluated the number of iterations needed for the algorithm to converge by comparing the differences of absolute prediction confidence scores between the current iteration and previous iteration. We simulated a population with two Gaussian clusters, 20 features, N = 10,000 and n = 1000, and a test set of size 500.

n_features \num_iteration	50	100	500	1000
10	0.05	0.0269	0.0178	0.008
20	0.0689	0.0332	0.0215	0.0111
50	0.0864	0.0384	0.0279	0.0129

Table 6.1: Average absolute differences of confidence scores compared to the previous iteration. 50 iterations is compared to 10 iterations.

From Table 6.1, we can observe that starting from 100 iterations and above, the scores returned by the algorithm differ much less than scores from the previous iterations. Therefore, we use 100 iterations to achieve fast and accurate implementation of true confidence approximation.

6.2 Simulations

In this section, we will create multiple data scenarios to test the metrics discussed in Chapter 4 and 5. The goal is to give concrete evidence of which confidence metrics most accurately describe population structures. We will also discuss the scenarios when local confidence does not have a substantial advantage.

We used Python's sklearn package to simulate datasets for classification (Pedregosa et al., 2011). Each population consists of clusters of points normally distributed (std=1) about vertices of a hyper-cube or polytope whose dimension and size are based on the number of informative features and the extent of class separation. In this thesis, we only focus on binary classification.

There are multiple parameters that we can control:

- n_features: the number of features.

- n_informative: the number of informative features.

- n_redundant: the number of redundant features.

- n_repeated: the number of repeated features, which are randomly generated as a linear combination of informative features.

- n_classes and n_clusters per class: the number of classes and number of clusters per class. A class can have multiple clusters each with a different spread in feature space. In this thesis, n_classes = 2.

- flip_y: the proportion of class labels that are randomly switched. A large value introduces more noise.

- class_sep: the factor multiplying the hyper-cube size. A larger values spreads out the clusters and make the classification task easier.

- hypercube: true if the clusters are put on the vertices of a hypercube or put on the vertices of a random polytope.

We designed populations using different combinations of the above parameters. They key questions to investigate are:

- How good are the confidence metrics when the classes are homogeneous versus heterogeneous?

- How do confidence metrics change when data is high-dimensional?

- What happens with noisy or highly correlated features?

The simulation procedure consists of three parts: creating a population based on the question we want to investigate, approximating the true prediction confidence, and calculating prediction confidence on a random training sample from the population.

Algorithm 3 Create a Population
---

procedure MAKEPOPULATION(population size N, n_features, n_informative, n_redundant and other data parameters to be evaluated, heterogeneity parameter $p_1,...,p_{n\_cluster}$)

Use sklearn.datasets.make_classification to generate a dataset with the set of parameters we specify in each data scenario.

Within each cluster i, randomly flip p_i of the observations to the other class.

end procedure

Algorithm 4 Simulation Procedure
---

procedure SIMULATION(The set of parameters for population generation,training sample size n, test sample size m)

run *makePopulation* to create a population.

Take m random observations from the population as the test set. We treat this test set as "new" data.

Take n random observations from the population as a real-life training sample.

Use the rest of the population to find true prediction confidence of the new data with *trueConfidence*.

Use the training sample to calculate the confidence metrics and compare them with the true confidence.

Return Acc_{oob}, Acc_{test}, $local\_confidence(test)$, $true\_confidence(test)$

end procedure

6.3   Simulation Results

Example 6.3.1 *We begin with a simple example to visually understand the confidence machinery. Simulate a population of 10,000 observations, two classes with one cluster per class, and two informative features. Randomly switch the class label of half of the observations to the other class in the yellow cluster and the blue cluster remains unchanged.*

simulation($N = 10,000$, $n = 1000$, $m = 500$, $n\_clusters\_per\_class = 1$, and $n\_features=n\_informative= 2$, $p\_yellow= 0.5$, $p\_blue= 0$.
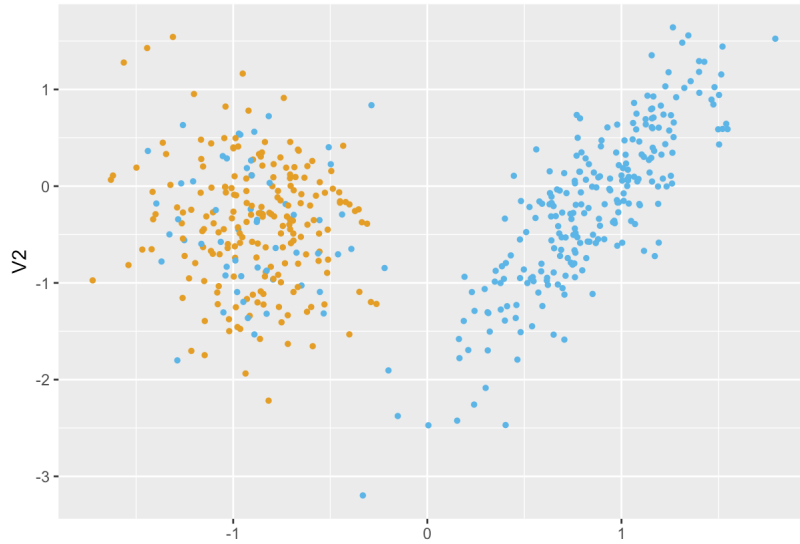

Figure 6.1: Visualization of the population in Example 6.3.1

The random forests are built with n_tree= 50. We pruned the trees by specifying the minimum node size. The most optimal node size was selected by oob-error. In this example, the minimum node size is 20. We can observe that the random forests can perfectly predict the homogeneous cluster but not the heterogeneous one because both yellow and blue classes share the same feature space in this cluster.

We take three observations from the test set for comparisons. One blue point from the heterogeneous cluster on the left, one blue from the homogeneous cluster on the right, and one yellow point. The three test observations and their cohabitants are in the following figure.

(a) The three benchmark test observations highlighted in triangles

(b) Cohabitants of the yellow observation

(c) Cohabitants of the blue observation from the left cluster

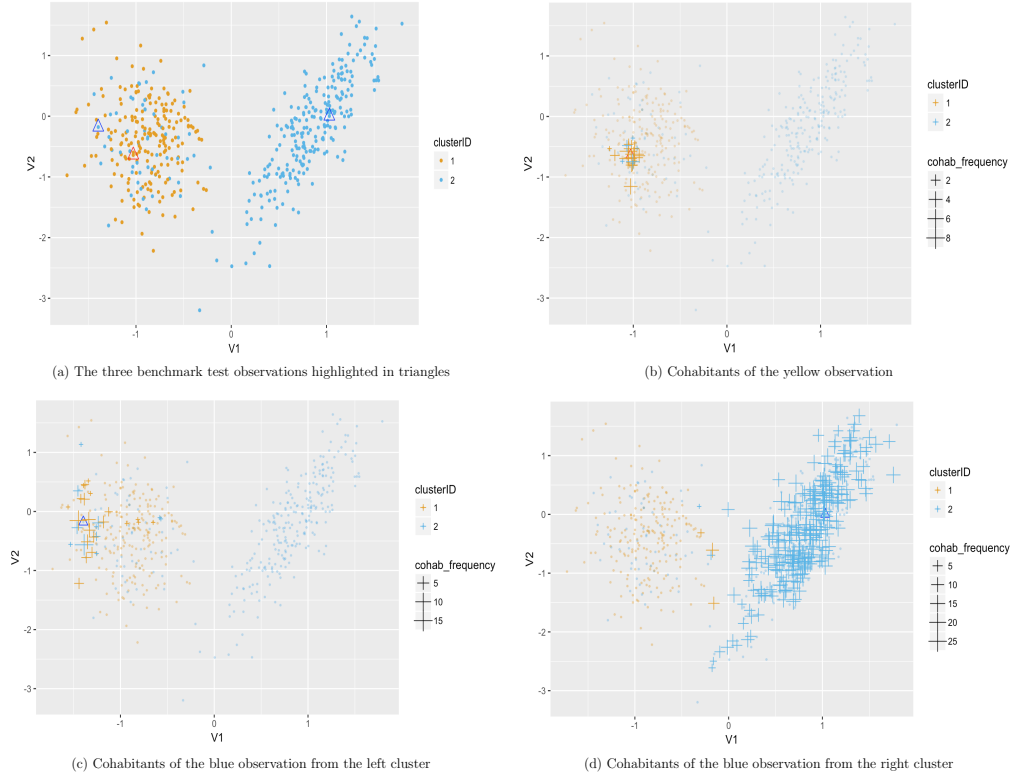(d) Cohabitants of the blue observation from the right cluster

Figure 6.2: A visualization of the cohabitants. All cohabitants are of the "+" shape. The size are determined by the frequency of the cohabitance. A random half of the cohabitants are plotted to avoid over-crowding the plots.

The confidence metrics of the three observations are summarized in Table 6.2 and 6.3.

	blue from heterogeneous	blue from homogeneous	yellow
Acc_{oob} (4.3)	0.842	0.842	0.842
Acc_{test} (4.2)	0.856	0.856	0.856
p_h (4.4)	1	0.56	0.74
$local\_confidence$ (6.1)	0.727	0.996	0.833
$true\_confidence$ (5.1)	0.08	1	0.85

Table 6.2: A summary of confidence metrics on Example 6.3.1

	Actual Yellow	Actual Blue
Predicted Yellow	183	61
Predicted Blue	11	245
Yellow Accuracy	0.943	-
Blue Accuracy	-	0.801

Table 6.3: Confusion matrix of test data

The overall metrics, Acc_{oob} and Acc_{test} fail to differentiate between a blue point from the homogeneous cluster and a blue point from the heterogeneous cluster. Even though the overall accuracy on the entire test set is more than 80%, the model performs badly on the heterogeneous cluster, which is not reflected in the overall metric's score. Even if we break down the overall accuracy by class, the test accuracy of the blue class is still inflated by the homogeneous cluster (by Table 6.3), thus not accurately showing the model's predictive capacity on observations living in the heterogeneous space.

To further compare local confidence and class probability, we plot confidence score versus class probability over the entire test set in Figure 6.3. Between the class probability and the proposed local confidence score, it appears that local confidence is closer to the true prediction confidence especially in not perfectly confident predictions. In Figure 6.3, for observations with true confidence lower than 0.5, the local confidence scores gives much lower estimates than class probability scores. For observations with true confidence greater than 0.5, both scores do fairly well. It is also worthy to note that local confidence score in general has less variance than class probability, making it a potentially better estimator.
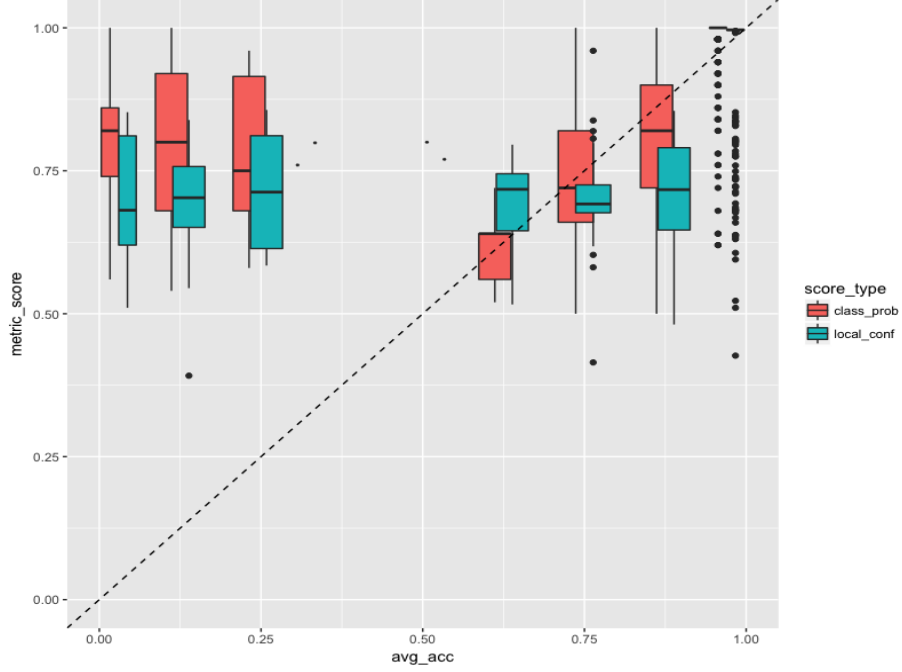
Figure 6.3: Comparisons between local confidence score and class probability on Example 6.3.1. All test observations are plotted. The dashed line is the line of y=x, therefore the closer each test observation's metric is to the dashed line, the better the metric is.

Even though this is too small of an example to justify why local confidence metric is better in wider applications, we do gain a high level intuition of the local confidence machinery. We can also make the initial hypothesis that local confidence does offer a better alternative to overall out-of-bag accuracy to capture the heterogeneity of the feature space, and in some situations perform better than class probability. We will explore more cases and repeated large-scale simulations to further that hypothesis.

Example 6.3.2 *In this example, we seek to investigate the performance of the confidence metrics when the data is high-dimensional. The same basic set-up from the previous example is used with in an increased number of features to 50. The random forests have n_tree = 100 and min.node.size = 30.*

$simulation(N = 10,000, n = 1000, m = 500, n\_clusters\_per\_class = 1,$
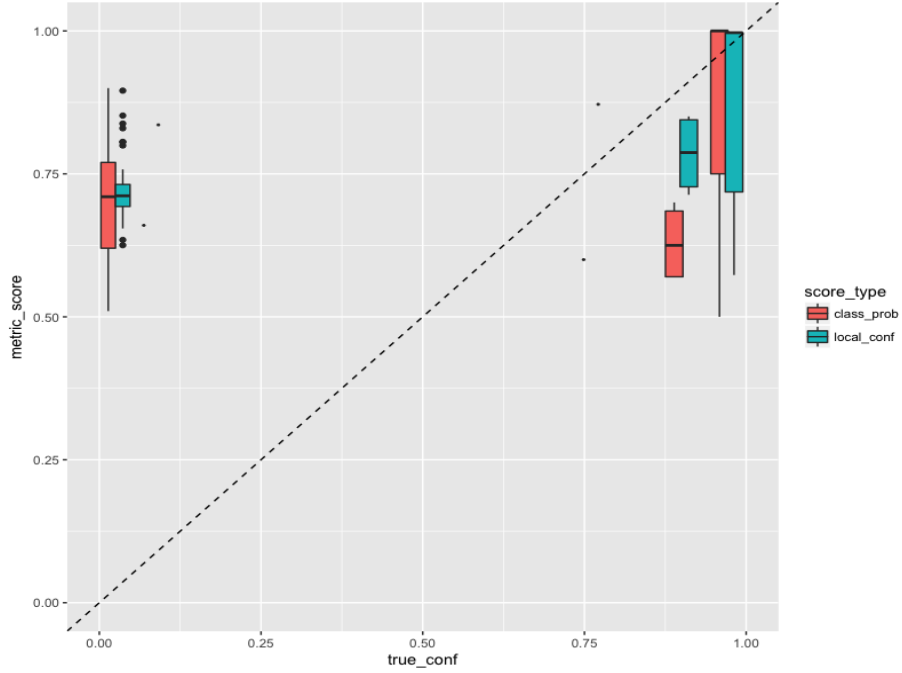$and\ n\_features{=}n\_informative{=}\ 50,\ p\_yellow{=}\ 0.5,\ p\_blue{=}\ 0.$

Figure 6.4: Comparisons between local confidence score and class probability on Example 6.3.2. All test observations are plotted.

The out-of-bag accuracy for all training data is 0.855 and test accuracy is 0.864. We see a slight improvement in overall accuracy because the high dimension enables the model to draw more separable boundaries. The local confidence score, class probability and true confidence are plotted in Figure 6.4. However, the result on high-dimensional data is not entirely consistent with the low-dimensional case since local confidence is not much better than the class probability. For a more rigorous argument, we will soon define metrics to quantify the differences between confidence metrics.

Example 6.3.3 *With a high dimensional data we are now concerned with noisy or correlated features. We will begin with noisy feature only. With*

other parameters in the previous example controlled, we run simulations with 30 informative features and 20 noisy features.

simulation(N = 10,000, n = 1000, m = 500, n_clusters_per_class = 1, and n_features= 50, n_informative= 30, n_repeated = 20, p_yellow= 0.5, p_blue= 0.
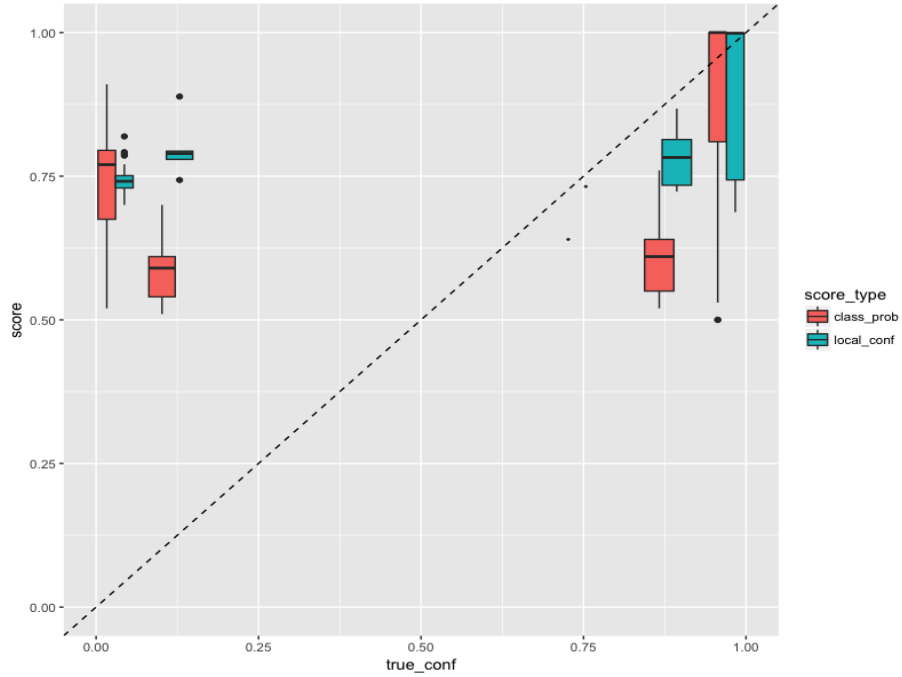

Figure 6.5: Comparisons between local confidence score and class probability on Example 6.3.3 with noisy features. All test observations are plotted.

We observe from Figure 6.5 that the relationship between class probability and local confidence score is not substantially affected by noisy features.

Example 6.3.4 *We next proceed with 30 informative features and 20 corre-lated features.*

simulation(N = 10,000, n = 1000, m = 500, n_clusters_per_class = 1, and n_features= 50, n_informative= 30, n_redundant= 20, p_yellow= 0.5, p_blue= 0.
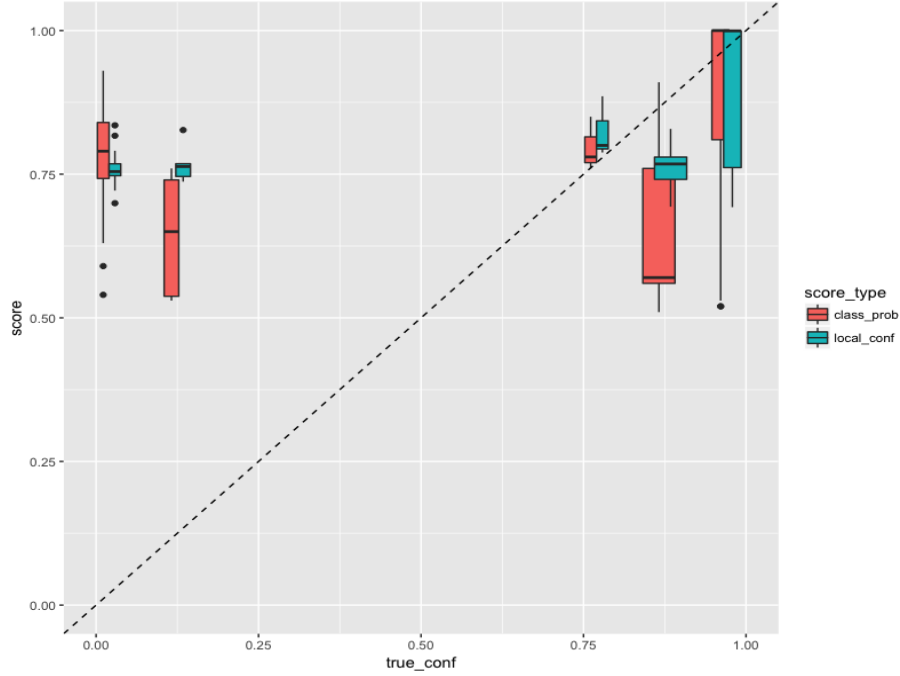
Figure 6.6: Comparisons between local confidence score and class probability on Example 6.3.4 with correlated features. All test observations are plotted.

Figure 6.6 shows that the class probability metric suffers from great variance with the presence of correlated features. In general, local confidence metric has much less variability than class probability. We can infer that this is because cohabitants are mutual: test observations that share a similar group of cohabitants mostly have very similar local confidence score. Small variance is another great property of local confidence score.

With high-dimensional data, local confidence score outperforms class probability when the true confidence is either greater than 0.5 or smaller than 0.1.

Example 6.3.5 *After exploring dimensionality, we will take another step to simulate more interesting data with different levels of heterogeneity in the feature space. Use total number of 50 features, with 20 informative, 20 noise and 10 correlated. Simulate two clusters per class and the clusters have 50% and 20 % homogeneity respectively.*

simulation($N = 10,000$, $n = 1000$, $m = 500$, $n\_clusters\_per\_class = 1$, and $n\_features$= 50, $n\_informative$= 20, $n\_repeated$= 20, $n\_redundant$= 10, p_yellow= 0.5, p_blue= 0.2.

Figure 6.7: Visualization the population in Example 6.3.5

Figure 6.8 shows that in this data scenario local confidence score is a closer approximation of individual prediction confidence than class probability, especially for observations with less than 100% true confidence. We can also see that local confidence is conservative for observations with very high true confidence, in other words, observations in very homogeneous feature space. This can be explained by our definition of local confidence: a new observation has a local confidence of 100% if and only if all its oob-cohabitants
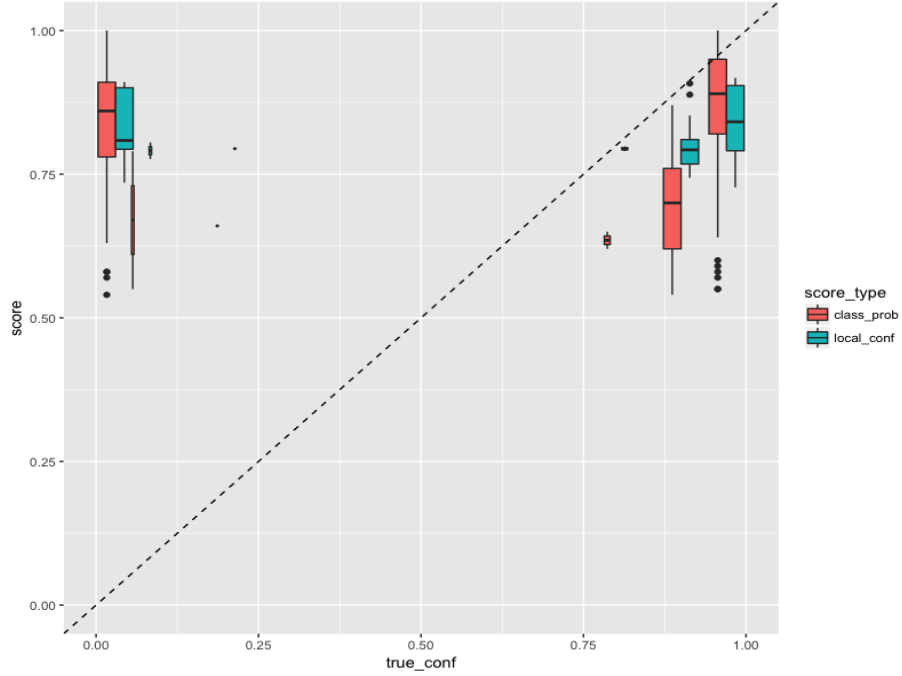
29

Figure 6.8: Comparisons between local confidence score and class probability on Example 6.3.5 with different level of heterogeneity.

are predicted correctly by the model, which is an impossible criteria to meet since we don't perfectly fit the training data.

In the next section, we will strengthen the claim that local confidence is a better estimator of individual prediction confidence by quantifying the differences between the confidence metrics and running comparisons on numerous simulated data.

6.4   Root Mean Square Error

To quantify how well the confidence metrics estimate confidence, we use root mean squared error:

$$\sqrt{\frac{1}{n}\sum_{i}^{n}(p_i - \hat{p}_i)^2}$$

where n is the number of test observations, p is the approximated true confidence, and $\hat{p}$ is either local confidence score or class probability.

Using this error metric, we will compare all the previous examples.

	example1	example2	example3	example4	example5
local_confidence	**0.250**	**0.304**	**0.293**	**0.284**	**0.351**
p_h	0.265	0.306	0.294	0.291	0.355
Acc_{oob}	0.291	0.326	0.317	0.306	0.352

Table 6.4: The root mean square error between the confidence metrics and approximated true confidence for the examples in the previous section.

From the table, we can conclude that for our simulations, out-of-bag accuracy is the worst method to estimate individual prediction confidence compared with local confidence and class probability. Local confidence has smaller root mean square error than class probability in all examples, but the margin is not substantial. We believe that local confidence is a very promising estimator, but we need more empirical simulations to fully support the claim.

6.5 Theoretical Justification

In addition to empirical evaluation, we also seek a theoretic grounding for our proposed local confidence using a probabilistic estimation framework. The probability calibration in this section is based on Olson and Wyner (2018).

Prediction confidence can be interpreted as the probability that the model predicts the response correctly given its feature space. To put it formally, let $z_1, ... z_n$ be the set of training observations, and $z_i = (x_{i_1}, ... x_{i_p}) \ \forall i \in 1, ... n..$ Let $y_1, ... y_n$ be the categorical response variable. Let z_h be a new observation and x_h be its feature space. Then we can define the prediction confidence of z_h as

$$P(\hat{y}_h = y_h | x_h)$$

Using Bayes' Rule, we can further write this probability as

$$P(\hat{y}_h = y_h | x_h) = \frac{P(\hat{y}_h = y_h)P(x_h | \hat{y}_h = y_h)}{P(x_h)}$$
$$\approx \frac{\hat{P}(\hat{y}_h = y_h)\hat{P}(x_h | \hat{y}_h = y_h)}{\hat{P}(x_h)} \tag{6.1}$$

We can interpret the cohabitance frequency as a proximity function, that is $w(z_h, z_i)$, the number of times a training observation z_i out-of-bag cohabs with z_h, is large if z_i and z_h are similar to each other and small vice versa. Then

$$\hat{P}(x_h) \equiv \frac{1}{n} \sum_{i=1}^{n} w(z_i, z_h) \tag{6.2}$$

Furthermore

$$\hat{P}(x_h | \hat{y}_h = y_h) \equiv \frac{1}{n_1} \sum_{i=1}^{n} w(z_h, z_i) \mathbb{I}(\hat{y}_i = y_i) \tag{6.3}$$

where n_1 is the number of oob training samples that are predicted correctly. Let

$$\hat{P}(\hat{y}_h = y_h) = \frac{n1}{n} = Acc_{oob} \tag{6.4}$$

the oob prediction accuracy. Then

$$P(\hat{y}_h = y_h | x_h) \approx \frac{\hat{P}(\hat{y}_h = y_h)\hat{P}(x_h | \hat{y}_h = y_h)}{\hat{P}(x_h)}$$
$$= \frac{\sum_{i=1}^{n} w(z_h, z_i) \mathbb{I}(\hat{y}_i = y_i)}{\sum_{i=1}^{n} w(z_i, z_h)} \tag{6.5}$$
$$= local\_confidence$$

Together with empirical evaluation, we made a convincing case that our proposed local confidence improves current estimation of individual prediction confidence.

Chapter 7

Applications and Future Directions

After empirically and theoretically evaluating our proposed local confidence and the other currently existing confidence metrics, the next pressing question to ask is, to whom is our proposed method useful? What are limitations of the local confidence score in wider applications?

Recall that local confidence represents the prediction confidence of an individual new observations, and that it better captures the heterogeneity in the feature space. Our result will be helpful to researchers whose work emphasizes individual predictions. Estimating individual outcome probabilities has applications in medicine, particularly diagnosis, decision on therapy or prognosis (Dankowski and Ziegler, 2016).

Local confidence does fall short when we have minorities sharing feature space with observations from predominately the opposite class. If the cohabitants are from the other class and are predicted almost perfectly, the minority will always be predicted as the majority and local confidence score will never reflect the true prediction accuracy. However, no other known methods can overcome such shortcoming.

Since the scope of the empirical evaluation in this thesis focuses only on binary classification, looking into the future, more simulations can be done in multiclass classification. In addition, our proposed local confidence can be further tested on real-world data, such as the UCI Machine Learning Datasets, to obtain a universal benchmark for comparisons of confidence metrics in other literature and ours.

The random forests we trained are the very first one created by Breiman

(2001). Yet we should not forget that there have been substantial research advancements in optimizing random forests in recent years. In Wager et al. (2014), a local estimation is proposed to assign more weights to training samples that are similar to the test observations when building the forests. It would be very interesting to see the performance of the local confidence metrics on random forests that are further optimized.

Chapter 8

Conclusion

Despite many literature in prediction intervals in regression, prediction confidence in classification random forests are predominately global metrics. Although class probability provides a probability estimation, very little is discussed on how that probability is related to the true prediction confidence. In this paper, we define local prediction confidence in a similar fashion as frequentist confidence intervals, as the prediction accuracy of one observation over repeated training samples.

We developed local prediction confidence directly using the random forests structure. The local prediction confidence of a new observation is approximated by the prediction accuracy of out-of-bag training observations that frequently oob-cohab with the new observation. We then simulated data sets of different dimensionality and homogeneity and evaluated local confidence along with class probability and out-of-bag accuracy.

We have empirically shown that local confidence better captures the heterogeneity of the feature space than other global metrics. We also observe that local confidence score gives a more accurate estimate of the true prediction confidence than class probability, and the estimator itself has smaller variance. In addition to empirical evaluation, we also showed theoretic justification using probability estimation.

The results offer a promising first step, yet additional work remains to be done. More simulations can be conducted to provide more concrete evidence, and real data set can be used to further test the differences of the metrics.

Chapter 9

Appendix

The R script that calculates local confidence is attached.

```
    #####
# A function that calculates the local prediction confidence of a test dataset
# parameters:
# inbag_counts: the number of times each training observation is
in bag in all trees
# trainNodes: train nodes returned by ranger
# testNodes: the test nodes returned by ranger
# train_response: the response of the training observations
# train_pred: training predictions
# test_response: the response of test observations
# test_pred: test predictions


library(ranger)

# return a frequest list of all training observations
# each element stores the number of times each training observation
is oob-cohab with x
getFreq<- function(inbag_counts,trainNodes,testNodes,x){
  freq_lst<-rep(0,nrow(trainNodes))
  for (b in 1:ncol(trainNodes)){
    node_x<-testNodes[x,b] # the terminal node ID of x
    oob_cohab<-which(trainNodes[,b]==node_x & inbag_counts[[b]]==0)
```

```r
    freq_lst[oob_cohab]=freq_lst[oob_cohab]+1
  }
  return(freq_lst)
}


# get the cohabitant of test observation x
getCohabitant <- function(inbag_counts,trainNodes,testNodes,x){
  freq_lst <- getFreq(inbag_counts,trainNodes,testNodes,x)
  return(which(freq_lst>0))
}


# return a list of lists of each test observation's cohabitant.
getAllCohabitant <- function(inbag_counts,trainNodes,testNodes){
  all_cohabs <- list()
  for (i in 1:nrow(testNodes)){
    cohabs <- getCohabitant(inbag_counts,trainNodes,testNodes,i)
    all_cohabs[[i]]<-cohabs
  }
  return(all_cohabs)
}


# return the confidence score of x
confidence<-function(inbag_counts,trainNodes,testNodes,
train_pred,train_response,x){
  accuracy<-allAccuracy(inbag_counts,train_pred,train_response)
  weight<-getFreq(inbag_counts,trainNodes,testNodes,x)
  score<-sum(accuracy*weight)/sum(weight)
  return(score)
}


#  return the confidence score of all test responses
allConfidence<-function(inbag_counts,trainNodes,testNodes,
train_pred,train_response){
  lst<-rep(0,nrow(testNodes))
  for (x in 1:nrow(testNodes)){
    conf<-confidence(inbag_counts,trainNodes,testNodes,train_pred,
    train_response,x)
    lst[x] = conf
```

```
  }
  return(lst)
}


## get out of bag prediction accuracy for training sample m
oobAccuracy <- function(m,inbag_counts,train_pred,train_response){
  oob_pred <- rep(0,ncol(train_pred))
  for(b in 1:ncol(train_pred)){
    if(inbag_counts[[b]][m]==0){
      oob_pred[b] <- train_pred[m,b]
    }
  }

  oob_pred <- oob_pred[oob_pred>0]

  final_pred <-as.numeric(names(which.max(table(oob_pred))))
  return(final_pred==train_response[m])
}

# get out of bag training accuracy for all training observations
allAccuracy<-function(inbag_counts,train_pred,train_response){
  allAcc<-rep(0,nrow(train_pred))
  for (m in 1:nrow(train_pred)){
    acc<-oobAccuracy(m,inbag_counts,train_pred,train_response)
    allAcc[m]=acc
  }
  return(allAcc)
}
```

Bibliography

Athey, S., Tibshirani, J., and Wager, S. (2016). Generalized random forests. *arXiv:1610.01271*.

Bhattacharyya, S. (2013). Confidence in predictions from random tree ensembles. *Knowledge and Information Systems*, 35(2):391–410.

Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5–32.

Dankowski, T. and Ziegler, A. (2016). Calibrating random forests for probability estimation. *Stat Med*, 35:3949–3960.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2014). *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York.

Lu, B. and Hardin, J. (2018). Prediction intervals for random forests.

Neter, J., Kutner, M. H., Nachtsheim, C. J., and Wasserman, W. (1996). *Applied Linear Statistical Models*. Irwin, Chicago.

Olson, M. and Wyner, A. (2018). Making sense of random forest probabilities: a kernel perspective.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Shafer, G. and Vovk, V. (2008). A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9:371–421.

Wager, S., Hastie, T., and Efron, B. (2014). Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *J. Mach. Learn. Res.*, 15(1):1625–1651.

Wright, M. N. and Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17.