Diagnosing Breast Cancer using Random Forests

Matheus Jun Ota
University of New Hampshire
mo1045@wildcats.unh.edu

Abstract

According to the World Health Organization breast cancer is the top cancer in women. Moreover, its incidence is increasing in some countries as a consequence of factors like the growing of urbanization and increase in life expectancy. Using a minimally invasive fine needle, researchers have been able to create data around the characteristics of the patient cells to discriminate between benign and malign breast tumors. We used this data to train a machine learning classifier written in R and based on a Decision Trees and Random Forests. The performance of our implementation was analyzed using the average accuracy of our algorithm, Receiver Operating Characteristic curves and the Features Importance.

Introduction

The quest of Supervised Learning algorithms is to infer a function from a labeled dataset. More formally, given an input vectors x_i and an associated value y_i, we want to generate the function f that best represents the data, and use it to predict the output y_j for a new input x_j. Many algorithm exists to do this task, one of these methods is called Decision Trees.

Decision Trees(Quinlan, 1986) are appealing because they are intuitive(meaning that unlike other methods like Neural Networks, they can be easily read by someone with no prior knowledge on the machine learning field), and performing the classification task is really fast(unlike other algorithms like K-Nearest Neighbors). Although the advantages are huge, decision trees can suffer from overfitting. Some methods have been suggested for overcoming this problem by having a bound on the purity of the leaves or pruning the fully grown tree. Another approach would be to use a set of multiple trees built using randomization techniques to perform the classification. The algorithm could then use the average of the classification on each tree as the final output, this technique was proposed by Leo Breiman(2001) in his seminal paper "Random Forests".

The goal of this paper is to use the Random Forests technique to predict if the breast tumor founded in a patient is malign or benign. This work was only possible because researchers from University of Wisconsin Hospitals have shared their data on their efforts for diagnosing breast cancer. Using computer vision techniques they managed to collect a diverse set of parameters about the cells characteristics of the patients. Using these as the features for our Random Forest classifier we managed to get an average accuracy of 94%.

Breast Cancer Dataset

The world health organization points out that the cornerstone for breast cancer control is early detection. In order to perform this diagnosis, medics have applied invasive methods in their patients. To solve this problem, Street, Wolberg & Mangasarian(1992) have proposed using a fine needle to collect information about the cells in the patients, then apply machine learning techniques in this data to infer if the cells are cancerous or not.

In order to prepare the images they first obtained a small drop of fluid from the breast tumor and put it on a glass slide, then they mounted a camera atop of a microscope to take images of the cells. Once they have these images, they applied computer vision techniques to find the boundaries of the cells nucleus. The techniques used to found these boundaries are called Active Contour Model or "snakes", and are common in computer vision to find boundaries on noisy 2D images.

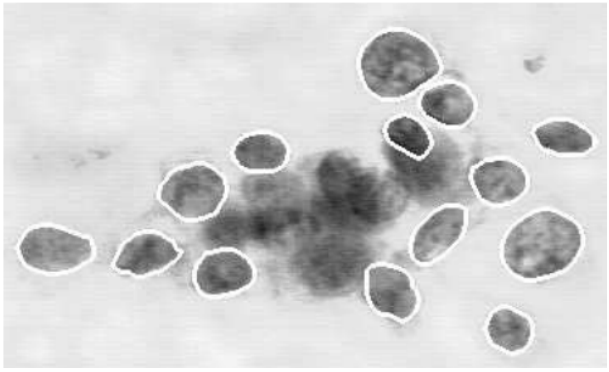

Figure 1: Snakes After Convergence to Cell Nucleus Boundaries(Street, Wolberg & Mangasarian, 1992)

Obtained these boundaries, they collected a series of features about the cells nuclei, which are described below.

Radius: The average radius of each cell nuclei.

Perimeter: The length of the snake.

Area: The number of pixels inside the snake. Pixels in the boundaries contribute as a half.

Compactness: $perimeter^2$ / $area$

Smoothness: The difference of a radial line and the mean of the lines around.

Concavity: They drawn a convex shape around the snake and measured the concavity degree by checking how much "inside" the convex shape some points of the snake are.

Concave Points: Similar as the "Concavity" feature, but just counting the number of points instead of measuring them.

Symmetry: They founded the major line r that passes through the center of the snake. Then they measured the difference of the lines perpendicular to r, starting from r and going up to the snake boundary, in both directions.

Fractal Dimension: They used the "coastline approximation" described by Mandelbrot to approximate the shape of the snake.

Texture: The variance of the grey scale pixels inside the snake.

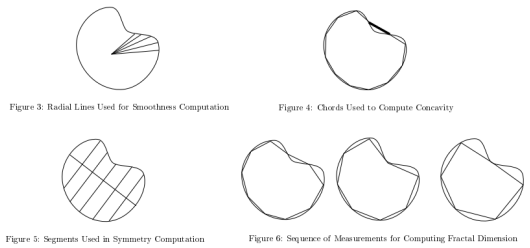

Figure 2: Visual explanation of some features

Decision Trees

Decision Trees are pretty straightforward to be read, even by someone without machine learning background. To classify a new example, we just need to make the tests on each node and proceed to the according child. Once we are in a leaf we have the class for our example. This process can be best illustrated on the following decision tree, plotted by us for the breast cancer diagnosis domain.

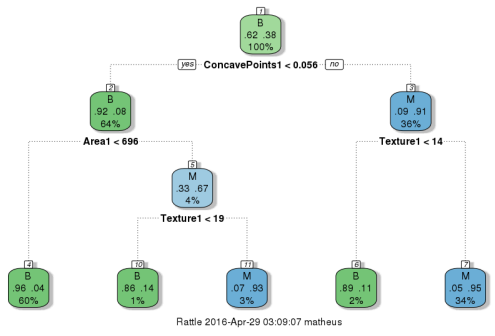

Figure 3: Example of a Decision Tree obtained for Breast Cancer Diagnosis(plot made with RATTLE package)

Given this tree, classifying is a really fast process in the computer, unlike other algorithms like K-Nearest Neighbors, where, with a simple approach, the computer needs to compute the distance of an example to all the other data examples. The problem is when creating the tree. It turns out that creating an optimal Decision Tree, the one that best splits the data, is an NP-complete problem(Hyafil & Rivest, 1976). Therefore, current Decision Trees algorithms constructs the trees using a greedy approach: we split on the feature that currently best separates the data, with no guarantee of having an optimal Decision Tree in the end.

function $DecisionTreeLearning(examples, attributes, parent_examples)$
  if $examples$ is empty **then return** $Plurality - Value(parent_examples)$
  else if all $examples$ have same classification c **then return** c
  else if $attributes$ is empty **then return** $Plurality - Value(examples)$
  else
    $A \leftarrow argmax_{a \in attributes} Importance(a, examples)$
    $tree \leftarrow$ a new decision tree with root test A
    for $v_k \in A$ **do**
      $exs \leftarrow \{e : e \in examples \wedge e.A = v_k\}$
      $subtree \leftarrow DecisionTreeLearning(exs, attributes - A, examples)$
      add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
    end for
  end if
end function

Figure 4: The algorithm for building a Decision Tree(Russel & Norvig, 2010). The function $PluralityValue$ returns the most common class in a set.

The selection of the feature that "best separates the data" can be done by using some techniques from Information Theory, more specifically the idea of entropy(Shannon & Weaver, 1949). This fundamental quantity measures how uncertain we are about a given variable, therefore, acquiring information corresponds to a reduction in the entropy, which is defined as follows.

$$H(V) = -\sum_k P(v_k) \log_2 P(v_k)$$

Where each v_k is a possible value for variable V.

Since we have a measure of how uncertain we are about a given variable, we can compute the entropy of the class variable in our dataset. Then we split in the features that best reduces our entropy, since this means that we are making our remaining dataset less uncertain.

We could apply the same reasoning with a measure that computes how "impure" the outcomes of a variable are. This measure is called the Gini Index, and it is actually the measured used in the RPART(Therneau & Atkinson, 2015)

package that we used for our implementation of the Decision Tree algorithm. The formula for the Gini Index is as follows.

$$G(V) = -\sum_{k} P(v_k)(1 - P(v_k))$$

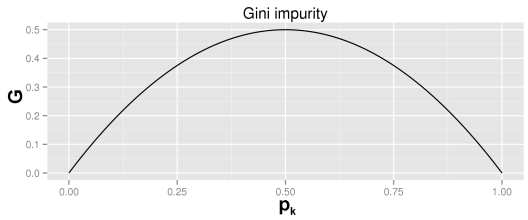

Figure 5: Plot of the Gini Index for a binary variable. The reader should notice how the curve reaches its peak when $p(v_k)$ is at 0.5. This is as expected, because it is the same point where the variable is more "impure".

One problem with this approach for building decision trees is overfitting. For example, the reader can notice on the presented algorithm that the building process will only stop when all examples in a node have the same classification. This makes the tree too specific for the training set, implying that it will probably not generalize correctly for the test set. There is two ways of attacking this problem. The first is pretty simple, instead of stopping the $DecisionTreeLearning$ routine when all examples are of the same classification, we could use some bound(say for example, when 90% of the nodes are of the same classification) to stop. Another way would be to prune the fully grown tree according to some measure of how good our tree perform on the training data and how complex(how much nodes) our tree have. The algorithm that performs this pruning is called Minimal Cost-Complexity Pruning(Breiman et al., 1984) and the measure used is called the Cost-Complexity, which is defined as follows.

$$R_\alpha(T) = R(T) + \alpha|T|$$

Where $R(T)$ is the Resubstitution Error and accounts for our performance in the training data, $|T|$ is how much nodes our tree have and α is the complexity parameter.

Random Forests

Another way to overcome the problem of overfitting, and improving the accuracy of our classifier would be to use Random Forests(Breiman, 1982). The idea behind Random Forests is to create various trees, with little differences between them. Then we pass our new example through all the trees and average(if our output is continuous) or take the maximum(if our output is discrete) of the result.

In order to create the "little differences" between them, we insert randomness in two steps of the process for building the trees. First, each tree is constructed using a random sample of the original training set. Second, instead of choosing from all the features the one that gives the maximum information gain, we choose from a random subset of the features. Below is the pseudo-code for the Random Forests algorithm.

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = majority\ vote\ \{\hat{C}_b(x)\}_1^B$.

Figure 6: Pseudo-code for the Random Forests algorithm(Hastie, Tibshirani & Friedman, 2009). The "bootstrap" term is used to describe the random sample to make explicit that the sampling process is with replacement, or in other words, an example may be selected more then once.

Implementation

The dataset used can be downloaded from the Machine Learning Repository of the University of California, Irvine(http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29). The code was implemented using R programming language, with the packages RPART(Recursive Partitioning) and PARTY(A Laboratory for Recursive Partytioning).

To test the algorithm we used 10-fold Cross Validation, meaning that we separated our dataset into 10 buckets, and at each iteration we used one bucket as the test set and the remaining buckets as the training set. We then computed the average accuracy, AUC(the area under the ROC curve) and the importance of some features.

Empirical Results

Firstly, we needed to find how much trees our Random Forest should have. To find this we ran the Random Forests algorithm with different number of trees(1, 10, 10^2, 10^3, 10^4) and computed the average accuracy, which is simply the number of example it classified right divided by the total number of examples in the dataset. The graph showed us that the accuracy of the algorithm reaches its peak when there is 100 trees in the Random Forest. Therefore, we compared the accuracy of the Decision Tree algorithm with a Random Forest with 100 trees.

Decision Trees	Random Forests(100 trees)
0.9138471	0.9419799

Table 1: Average accuracy for Decision Trees and Random Forests applied to the Breast Cancer Diagnosis dataset.

Another common way of verifying the performance of classifiers(specially the ones with medical purposes) is using

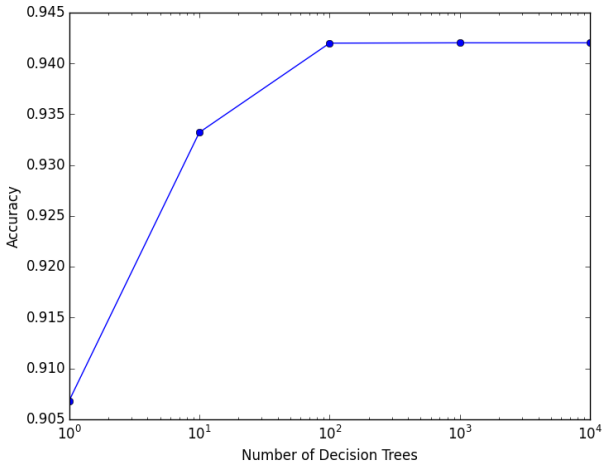

Figure 7: Graph of the accuracy by the number of trees in the Random Forest.

the ROC(Receiver Operating Characteristic) curve. Which is a plot of the True Positive Rate by the False Positive Rate, this way one can verify about the sensitivity and specificity of the test. In the breast cancer for example, we would like it to have a high True Positive Rate, even if to accomplish this we would need a little higher False Positive Rate. One measure that can be used given the ROC is the Area Under the Curve(AUC), where AUC of 1 is the perfect classifier and AUC of 0.5 is an useless test.

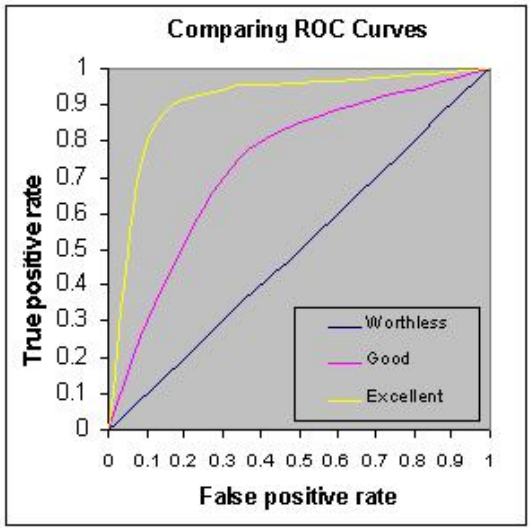

Figure 8: Example of ROC curves.

The following are the plots for the ROC for the Decision Trees and the Random Forests algorithms applied to the breast cancer dataset. There are multiple lines in the graphs because we plotted the ROC for each iteration in the 10-fold cross validation. The table shows the average AUC for each algorithm.

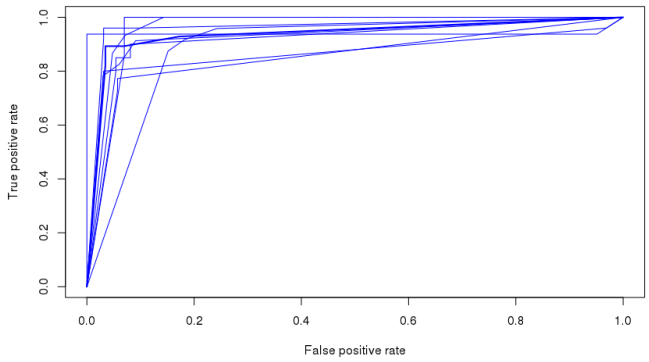

Figure 9: The ROC for the Decision Tree algorithm.

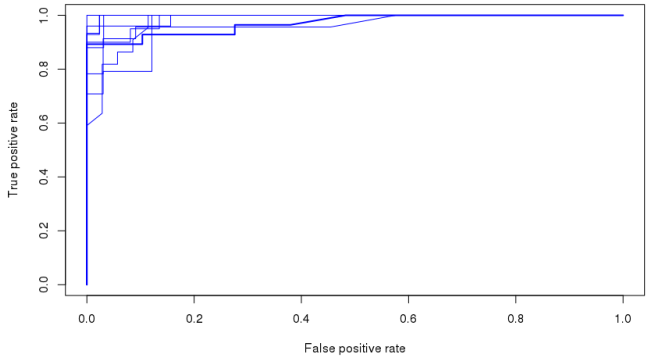

Figure 10: The ROC for the Random Forests algorithm.

Decision Trees	Random Forests(100 trees)
0.9233482	0.9866841

Table 2: Average AUC for Decision Trees and Random Forests applied to the Breast Cancer Diagnosis dataset.

Finally we also plotted how removing a feature will decrease the accuracy and the gini value of the leafs. This data is collected when building a Random Forest: instead of just wasting all examples that are not part of the bootstrap sample, we test our current classifier in these examples.

Conclusion

As expected, Random Forests performed slightly better in this dataset, but both Decision Trees and Random Forests have shown to be pretty strong classifiers, with accuracy and AUC over 90%. Implying that fine needles can indeed be used for early detection of breast cancer, instead of other invasive methods.

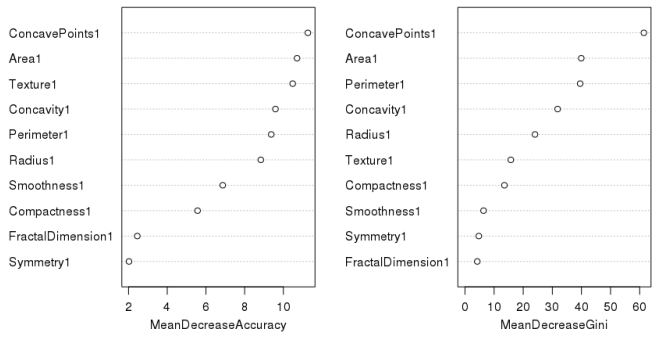

Figure 11: Graph of features importance.

Although these are pretty good results, our accuracy is still smaller then the one reported by Street, Wolberg & Mangasarian(1992) of 97%. Meaning that their approach using Multi-Surfaced Methods may be better for classifying breast tumors. On the other hand, they report using just the Area, Smoothness and Texture features for their classifier. Our built Decision Tree and the graph of features importance shows that probably Concave Points is the most relevant feature when diagnosing the cancer.

Future Work

For the future we would like to compare our results with other Machine Learning algorithms like Support Vector Machines and Neural Networks. We expect the result to be as good, or worst, then the ones obtained here. Because as reported by Delgado, Cernadas, Barro & Amorim(2014), Random Forests are likely to be the best classifier, at least for the datasets in the UCI database. But we still believe that implementing these methods are worth the effort for the purpose of learning.

We also would like to implement the same Random Forest algorithm to other cancer datasets like the Lung Cancer Data Set(http://archive.ics.uci.edu/ml/datasets/Lung+Cancer).
We believe that the Breast Cancer Dataset is pretty polished and with no need of using strong feature engineering techniques. For example, all the instances already had all their features filled with a number, and therefore there was no need to think about what to do when examples in the dataset are incomplete. Working with more "rough" datasets we would also need to work out the feature engineering process.

Acknowledgments

Thanks for professor Christopher Amato and the teacher assistant Sammie Katt for teaching the path to start playing with machine learning and probabilistic AI. We would also like to thanks the University of California, Irvine and Street, Wolberg & Mangasarian for making the incredible breast cancer dataset available to anyone interested.

References

Russell, S., Norvig, P., & Intelligence, A. (2010). A modern approach. *Artificial Intelligence. Prentice-Hall, Englewood Cliffs*, 25, 27.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Hierarchical clustering. *The elements of statistical learning, 2*.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5-32.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC press.
Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1), 81-106.

Street, W. N., Wolberg, W. H., & Mangasarian, O. L. (1993, July). Nuclear feature extraction for breast tumor diagnosis. In IS&T/SPIE's *Symposium on Electronic Imaging: Science and Technology* (pp. 861-870). International Society for Optics and Photonics.

R Development Core Team (2008), R: A language and environment for statistical computing.

Williams, G. J. (2011), Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery. *Use R!*, Springer.

Therneau, T., Atkinson, B., & Ripley, B. (2015). Recursive partitioning and regression trees.

Hothorn, T., Hornik, K., Strobl, C., & Zeileis, A. (2010). Party: A laboratory for recursive partytioning.

Sing, T., Sander, O., Beerenwinkel, N., & Lengauer, T. (2005). ROCR: visualizing classifier performance in R. *Bioinformatics, 21*(20), 3940-3941.

Hyafil, L., & Rivest, R. L. (1976). Constructing optimal binary decision trees is NP-complete. *Information Processing Letters, 5*(1), 15-17.

Fernndez-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems?. *The Journal of Machine Learning Research, 15*(1), 3133-3181.