Some Brief Notes on Ensemble Methods:
AdaBoost, Random Forest, and Gradient Boosting

Alex Duda

I. Introduction

In general, for binary classification of numerical data, ensemble methods can be quite strong for a variety of reasons. Here we recall three prominent ensemble methods [1] - AdaBoost Classifier, Random Forest Classifier, and Gradient Boosting Classifier. For each we will detail some recent real-world applications, potential strengths, potential limitations, and some scenarios where the method is often productively applied. This is *not* meant to be comprehensive but to serve as a useful collection of notes for the machine learning practitioner. It may be updated in the future[1].

II. AdaBoost Classifier

A. Real-World Applications

1) Network intrusion detection [2]
2) Driver fatigue detection, based on EEG signals [3]
3) Credit card fraud detection [4]

B. Potential Strengths

1) Interpolation achieved after relatively few iterations
2) Generalization error continues to drop even after interpolation is achieved and maintained
3) Being fast compared to other similar models
4) Simple and easy to program
5) Few hyperparameters to tune (consider this could also be a weakness)
6) Flexible (can combine with any learning algorithm)
7) No prior knowledge needed about weak learner,
8) Provably effective (only need to find a rough rule of thumb, weak learners just barely better than random guessing)
9) Versatile (can be used with text, numeric, discrete, data)

[1]**Original Version:** 2020-11-18, **Updated:** 2021-01-25

C. Potential Limitations

1) It depends on data and weak learner
2) As suggested by theory, AdaBoost can fail if weak classifiers are too:
   a) Complex, which can lead to overfitting
   b) Weak, which can lead to underfitting OR overfitting (resulting from low margins)
3) As shown empirically, AdaBoost seems weak when presented with uniform noise; thus, if one's dataset is noisy, it may perform weakly.
4) When irrelevant features are included, AdaBoost's performance can suffer.
5) Generally, AdaBoost is not optimized for speed (especially when compared to XGBoost, for example).
6) With regard to hyperparameter tuning, there is not much to tune beyond:
   a) Maximum depth of weak learners,
   b) Learning rate,
   c) Number of iterations

D. Applicable Scenarios

1) Numeric features
2) Binary discrete classification
3) Low-noise dataset
4) When time is not a strong factor (can wait for training and inference)
5) When computational resources are limited
6) When there is not copious time for hyperparameter tuning
7) When user is not necessarily knowledgeable enough to perform very nuanced hyperparameter studies

III. Random Forest Classifier

A. Real-World Applications

1) Behavior-based intrusion detection systems based on Random Forest [5]
2) Predict whether a given COVID-19 patient would recover or not [6]

B. Potential Strengths

1) High accuracy
2) Relatively robust against noise and outliers
3) Each weak learner is generated independently (by focusing it on a given data subset), which means the potential to parallelize the training; thus, if one had access to parallel computing resources one could run a parallel implementation of random forest which could have better accuracy achieved in less time!
4) Fast
5) Can perform implicit feature selection
6) Simple to implement
7) Simple to visualize
8) Numerous hyperparameters to tune:
9) Number of features (lower number of features should be chosen, usually around one third, to avoid overfitting)
    a) Number of trees (large number of trees helps)
    b) Maximum tree-depth (relatively ”low”)
    c) Whether to bootstrap samples
    d) Minimum number of samples left in a node before a split (relatively ”low”)
    e) Minimum number of samples left in the final leaf node
10) When compare to AdaBoost, Random Forest:
    a) Is less impacted by noise
    b) Provides better generalization, via reduced variance
        i) This can be achieved (as guaranteed by Central Limit Theorem) by approaching generalization error limit with increasing tree growth.

C. Potential Limitations

1) Increased hyperparameter tuning is needed due to a higher number of relevant, performance-impacting, parameters (those this might be considered a strength to some)
2) Introduction of randomness into training and testing data, which may not be appropriate for all data sets
3) More knowledge is needed from user to tune RF (when compared to AdaBoost)
4) It should not be used for time series data or other data where one wants to avoid look-ahead bias, as well as order and continuity of samples needs to be preserved

D. Applicable Scenarios

1) When time allows for hyperparameter tuning
2) When inserted randomness will not strongly alter data
3) When user is familiar with method and impact of tuning parameters
4) When one prioritizes stated strengths (high accuracy, implicit feature selection, etc.)
5) When one has access to parallel computation, which can allow for much larger models to be run, as the method is parallelizable

IV. Gradient Boosting Classifier

A. Real-World Applications

1) Classifying scenes obtained by high resolution imagery from satellite sensors [7]
    a) Note that this particular application is not just a normal gradient boosting framework using decision trees! In fact, it uses a gradient boosting scheme with convolutional neural networks (CNNs) to obtain a CNN ensemble that outperforms state-of-the-art scene classifiers
2) Selecting features in data with many features [8]; this can help with
    a) Identifying which features are most important,
    b) Vastly reducing number of features, which can
        i) Improve model accuracy
        ii) Reduce training time
3) Predicting clicks on Ads at Facebook [9]
4) Learn hierarchical distributed representations by stacking several layers of regression GBDTs as its building block [10]

B. Potential Strengths

1) Easy to interpret results
2) Implicit feature selection
3) Stage-wise-additive model; existing weak learners are frozen; new weak learners are added one-at-a-time to help correct errors of previous model, provides good control to crafting a ensemble

4) Arbitrary differentiable loss functions could be used, which means that the technique could be used not only for binary classification (logarithmic loss) but also regression (squared error), multi-class classification, etc.

C. Potential Limitations

1) Each of the three main parameters (shrinkage, tree depth, tree number) need to be fit properly to get good performance, which sometimes can take some work (when compared to the simpler tuning of Random Forests)
2) Greedy algorithm that can overfit a training dataset quickly (especially exacerbated by noisy data), such overfitting can be helped via:
   a) Tree constraints - ensure weak learners remain ”weak enough” (in general, the weaker the learners are kept, the more of them will be needed and this will help)
   b) Shrinkage (aka ”learning rate”) - weight the contribution of each new weak learner to ensure each has a reduced influence and learning is slowed (which will drive the need for more weak learners)
   c) Random sampling (aka ”Stochastic Gradient Boosting”) - reduce the correlations between weak learners in the sequence by drawing a training data subsample at each iteration (without replacement) and using it to fit the new weak learner being created during that iteration.
   d) Penalized learning - use a modified weak learner (a regression tree instead of a classical decision tree like CART) and regularize the leaf weight values
3) The weak learners are built sequentially (since later models depend on earlier ones, this cannot be done in parallel) which could mean longer training times than other approaches (like random forest as discussed previously)
   a) Other implementations, like XGBoost addresses some of these issues by enabling parallelization, distributed computation, cache optimization, out-of-core-computing, etc.
4) Sensitive to outliers - since a new weak learner focuses on fixing the errors of the previous weak learner, it may be too impacted by outliers!
5) Does not natively support categorical features; needs some kind of pre-processing step that converts categorical features into numerical features (like one-hot encoding). Some modified versions do attempt to improve support for categorical features, as with LightGBM and CatBoost.
6) Missing data can lead to much slower training
   a) There are techniques that can help really speed up training (like replacing missing values with 0 or some kind of better predicted value)

D. Applicable Scenarios

1) Records with missing or ill-formatted entries were already removed
2) Data does not have high noise
3) Time is not a major factor (can afford to wait a little)
4) There are few (if any) outliers (perhaps some t-SNE style clustering might help during a data exploration step to determine)
5) Categorical features have been converted via one-hot encoding
   a) Features with more than 5 levels might cause issues; consider making adjustments
6) Ensure data set does not exacerbate the limitations but seems like it will benefit from the strengths

E. Model Description

As numerous recent results have shown variants of GBCs to be *very* powerful [11] [12], it seems appropriate to provide a bit more detail on how this particular model works [13] [14] [15]. Perhaps at some time in the future similar descriptions will be provided for the other models.

1) Conceptual: Gradient Boosting Classifiers (GBCs) should be thought of as a sequential process that, over time, constructs a model (made up of a sequence of finely tuned ”weak models”) that is better and better at making correct class predictions. In fact, GBCs produces a collection of models, whose predictions can be intelligently combined to produce an overall prediction that is much better than any of the base models could achieve on their own, as well as much better than many well-known single ”strong models”. Conceptually, we could enumerate the steps as follows:

1) GBC starts off with a guess (a naive prediction) for the classes.
2) This guess is compared to the real classes
3) An error between the guess and the actual is computed
4) A new model is constructed that can help reduce the error
5) This new model is then intelligently added to the existing model to form an ”ensemble” (the collection of models thus far constructed)
6) The updated ensemble is used to generate a new set of predicted classes (a new ”guess”)
7) One continues to step 2 and repeats until a user-specified termination, which can be determined in a number of different ways (like a certain number of new models have been generated, a certain performance level is reached, or performance improvement has slowed down)

GBC is highly flexible and allows one to tune a variety of parameters based on one’s understanding of one’s data, to make a strong result likely.

2) Nonspecialist but Technical: GBC involve three primary components:

1) Loss function (which will be optimized),
2) Weak base-learner to generate predictions
3) Sequential additive model to add base-learners in a manner that minimizes the loss function

GBCs complete the following steps:

1) Initialize ensemble with some kind of naive prediction model (for example, using the simple ”odds”)
2) Compute ensemble ”residuals” (the error) as a function of the ensemble predicted classes and the actual predicted classes
3) Generate a new weak base-learner with inputs (training data) and labels (residuals from step 2) that minimizes the specified loss function (correlates with the direction of the negative gradient of the loss function)
4) Sequentially add the optimized base-learner to the ensemble, using a specified learning rate (which determines how much the prediction from the new base-learner should impact the ensemble’s prediction)
5) Use the updated ensemble to generate a new set of class predictions
6) Go back to step 2 (repeat this process for M iterations, where M represents the number of base-learners that will sequentially be added to the ensemble)

V. Conclusion

All three ensemble methods are quite powerful. If one is looking to solve a binary classification problem for numerical data, these would serve as an appropriate start. However, keep in mind that for a particular situation, various modifications might need to be made and implications examined. For instance, if one has *imbalanced classes* (where one does not have equal numbers of training samples for each class [16]), there are numerous potential remedies [17]. For example when using Python machine learning library scikit-learn, for the RF case :

1) Special modifications to regular models can be adopted:
   a) Class weighting - whereby one changes the weight that each class has when computing the impurity score of a given chosen split point (change RandomForestClassifier model parameter class_weight to ’balanced’)
   b) Bootstrap class weighting - whereby one changes the class weighting based on class distribution in each bootstrap sample (change RandomForestClassifier model parameter class_weight to ’balanced subsample’)
2) Special models can be leveraged from other libraries (like imblearn.ensemble library):
   a) Random under-sampling - whereby one performs data sampling on the bootstrap sample to change class distribution; in particular, one can undersample the

majority class for each bootstrap sample (see BalancedRandomForestClassifier from imblearn.ensemble library)

Also, though commonly used, intuitively, one-hot encoding seems to have potential drawbacks like generating many orthogonal features that [18] [19]:

1) Appear to have no particular relationship (which may have been potentially useful information) - as when for a given category there are numerous, related, choices
2) Make scaling up the model challenging

One should keep in mind such consideration, among others, when applying these ensemble methods.

References

[1] J. Nikulski, "The ultimate guide to adaboost, random forests and xgboost," https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f, Towards Data Science, 2020, accessed: 2020-11-18.

[2] W. Hu, W. Hu, and S. Maybank, "Adaboost-based algorithm for network intrusion detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 577–583, 2008. [Online]. Available: https://doi.org/10.1109/TSMCB.2007.914695

[3] J. Hu, "Automated detection of driver fatigue based on adaboost classifier with eeg signals," *Frontiers in Computational Neuroscience*, vol. 11, p. 72, 2017. [Online]. Available: https://doi.org/10.3389/fncom.2017.00072

[4] K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi, "Credit card fraud detection using adaboost and majority voting," *IEEE Access*, vol. 6, pp. 14 277–14 284, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2806420

[5] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Comput. Surv.*, vol. 51, no. 3, May 2018. [Online]. Available: https://doi.org/10.1145/3178582

[6] C. Iwendi, A. K. Bashir, A. Peshkar, R. Sujatha, J. M. Chatterjee, S. Pasupuleti, R. Mishra, S. Pillai, and O. Jo, "Covid-19 patient health prediction using boosted random forest algorithm," *Frontiers in Public Health*, vol. 8, p. 357, 2020. [Online]. Available: https://doi.org/10.3389/fpubh.2020.00357

[7] F. Zhang, B. Du, and L. Zhang, "Scene classification via a gradient boosting random convolutional network framework," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1793–1802, 2016. [Online]. Available: https://doi.org/10.1109/TGRS.2015.2488681

[8] H. Rao, X. Shi, A. K. Rodrigue, J. Feng, Y. Xia, M. Elhoseny, X. Yuan, and L. Gu, "Feature selection based on artificial bee colony and gradient boosting decision tree," *Applied Soft Computing*, vol. 74, pp. 634 – 642, 2019. [Online]. Available: https://doi.org/10.1016/j.asoc.2018.10.036

[9] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. n. Candela, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, ser. ADKDD'14. New York, NY, USA: Association for Computing Machinery, 2014, p. 1–9. [Online]. Available: https://doi.org/10.1145/2648584.2648589

[10] J. Feng, Y. Yu, and Z.-H. Zhou, "Multi-layered gradient boosting decision trees," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 3551–3561.

[11] Y. Freund, "Real-world applications of boosting," 2012, accessed: 2020-11-18.

[12] A. J. Wyner, M. Olson, J. Bleich, and D. Mease, "Explaining the success of adaboost and random forests as interpolating classifiers," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 1558–1590, Jan. 2017. [Online]. Available: https://dl.acm.org/doi/10.5555/3122009.3153004

[13] Z. Zhang, "Boosting algorithms explained," https://towardsdatascience.com/boosting-

algorithms-explained-d38f56ef3f30, Towards Data Science, 2019, accessed: 2020-11-18.

[14] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in Neurorobotics*, vol. 7, p. 21, 2013. [Online]. Available: https://doi.org/10.3389/fnbot.2013.00021

[15] J. Brownlee, "A gentle introduction to the gradient boosting algorithm for machine learning," https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/, Machine Learning Mastery, 2020, accessed: 2020-11-18.

[16] ——, "A gentle introduction to imbalanced classification," https://machinelearningmastery.com/what-is-imbalanced-classification/, Machine Learning Mastery, 2020, accessed: 2020-11-18.

[17] ——, "8 tactics to combat imbalanced classes in your machine learning dataset," https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/, Machine Learning Mastery, 2020, accessed: 2020-11-18.

[18] P. L. Saint, "How do gradient boosting algorithms handle categorical variables?" https://blog.dataiku.com/how-do-gradient-boosting-algorithms-handle-categorical-variables, Dataiku, 2020, accessed: 2020-11-18.

[19] D. Martin, "Are you getting burned by one-hot encoding?" https://kiwidamien.github.io/are-you-getting-burned-by-one-hot-encoding.html, kiwidamien.github.io, 2019, accessed: 2020-11-18.