Random forests for facial recognition

Carl Ehrett

May 6, 2019

1 What random forests are

Random forests are an ensemble learning method that uses bagging in conjunction with random feature selection to reduce the variance of a decision tree estimator. Random forests are popular for their supposed resistance to overfitting and related low level of tuning required. Random forests often work well with "out-of-the-box" default settings, so that tuning is often a matter of coaxing fairly small improvements out of the model.

In this paper, I describe the nature and history of random forests in the remainder of Section 1. I explore some of the benefits of random forests in Section 2. I also describe some of the grand claims that are often made about random forests, and explore the truth of those claims, in Section 3. I demonstrate the use of random forests as a facial recognition classifier in Section 4. Throughout this paper, I will focus on the use of random forests as a classifier (which is the use for which random forests are more popular).

1.1 Decision trees

A decision tree is a partitioning of the feature space into hyperrectangles the edges of which are aligned with the axes of the feature space. The decision tree assigns to each subset of the partition a prediction value. If a tree is used for classification, then a point x is classified as being in the category to which a plurality training samples belong in the subset to which x belongs; for regression, x is associated with the mean value of the samples in the subset. An example from Hastie et al. (2009) is shown in Figure 1, where a single tree is visualized in two different ways.

The decision trees used in random forests are not required to be grown following any particular algorithm. The most common algorithm to use for this purpose is the CART algorithm Breiman et al. (1984). All tuning parameters that apply to the decision trees used in random forests are, by extension, tuning parameters for random forests itself. For example, the maximum depth of trees can be tuned, or the minimum number of samples to include in each terminal node, or the minimum improvement (measured e.g. by Gini impurity) required to split a node.

1.2 Bagging

Bagging, or bootstrap aggregation, is a means for reducing the variance of an estimator by combining many versions of that estimator (Breiman, 1996). Sup-

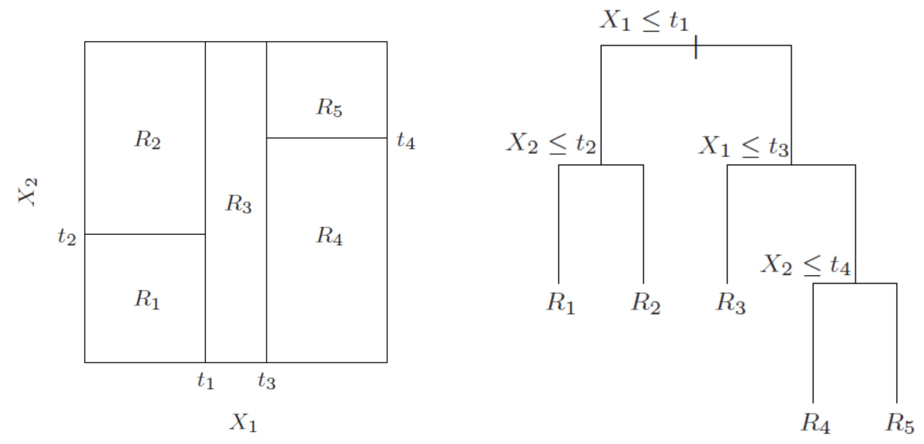

Figure 1: Two representations of the same decision tree on a two-dimensional feature space, such as would be grown using the CART algorithm.

pose one has a method for producing an estimator $\hat{f}(x)$ at a point x using some training data. To produce a bagged version of this estimator, one would do the following.

1. Produce B bootstrap samples of the training data.
2. Fit the model on each bootstrap sample.
3. Get a prediction $\hat{f}^{*b}(x)$ using each bootstrap sample fit $b = 1, \dots, B$.
4. Combine all of these predictions. For regression, $\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$, and for classification $\hat{f}_{\text{bag}}(x) = \text{argmax}_k \sum_{b=1}^{B} \mathbf{1}_{\hat{f}^{*b}(x)}$

Bagging helps to address one of the primary drawbacks of decision trees: their high variance. The variance of an average of B identically distributed variables with variance σ^2 and pairwise correlation ρ is given by:

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2. \tag{1}$$

The second term above is driven to 0 as $B \to \infty$.

1.3 Random feature selection

Notice that (1) above shows that bagging can reduce only the second term in that expression. If σ^2 and ρ are both high, then the resulting variance of the bagged estimator will itself remain high. Therefore, since decision trees do indeed suffer high variance, random forests include random feature selection to reduce the correlation amongst the trees in the forest.

The use of random feature selection in ensembles of decision trees was suggested by Tin Kam Ho (1995). Breiman (2001) combined this idea with bagging to produce the method that is now known (and indeed trademarked by Breiman and his colleague Adele Cutler) as random forests.

To employ random feature selection in growing a decision tree one does as follows. Each time a node is split when growing a decision tree, the splitting variable for that node is chosen from amongst a random subset of $m < p$ of the p predictors. Breiman suggests a default of $m = \sqrt{p}$ for classification and $m = p/3$ for regression, but it is often profitable to tune m rather than relying on these defaults.

2 OOB error and feature importances

By virtue of using bagging and decision trees, random forests boasts some features that improve the utility and interpretability of the technique, including out-of-bag error and methods for estimating feature importances.

2.1 OOB error

Out-of-bag (OOB) error is a means for estimating the generalization error of a random forest without needing either a test set or cross-validation. OOB error is available whenever one uses bagging, as in random forests. To find the OOB error after training a model, one tests each of the training samples x_i using only the bagged models that do not include x_i in the bootstrap sample set upon which that model was trained. For example, in the case of random forest classification, we would test x_i by classifying it using all and only the trees that were grown from a bootstrap sample set which does not include x_i. These are the trees for which x_i was "out of the bag" upon which the trees were trained.

The result is similar to cross-validation, which also involves testing each training sample x_i on versions of the model that were trained on subsets of the training set that exclude x_i. Thus random forests provides an analogue to cross-validation that does not requires neither training the model multiple times, nor setting aside a portion of the data to serve as a test set.

2.2 Feature importances

Decision trees are valued for their high interpretability. In contrast with "black box" methods such as neural nets, decision trees explicitly show how the covariates relate to the estimate. For example, in Figure 1, we can see that a point in the region R_1 gets the estimate that it does because $X_1 < t_1$ and because, as a point with $X_1 < t_1$, that point also has $X_2 < t_2$. All such points, with $X_1 < t_1$ and $X_2 < t_2$, belong in region R_1 and receive a common estimate.

Though random forests are built from decision trees, this form of interpretability is lost through the bagging procedure. Though each tree in the forest remains interpretable in this way, the forest itself, as an average of trees, does not offer the same explicit guide to how a prediction point is assigned its estimate. This is a downside of using random forests instead of a single decision tree.

However, random forests can nonetheless regain some of this interpretability through other means. Specifically, various methods are available for estimating the relative importances of the predictor variables used in random forests. Three such methods are as follows:

1. For each node split while growing a tree, record the improvement (e.g. measured via Gini impurity) generated by the split. For each covariate, average the improvements it produces each time it is used for a split (Hastie et al., 2009).
2. For each tree, find the OOB error, then for each covariate measure the increase in OOB error when you randomly permute that covariate's values. Average these values across all trees (Breiman, 2001).
3. For each node split, record the proportion of training samples that are "downstream" of that split. For each variable, find the average proportion of samples that are "downstream" of its splits (Pedregosa et al., 2011).

These methods can be combined. For example, Pedregosa et al. (2011) combine scores from the first and third methods above to estimate the relative importances of the covariates.

3 Claims about random forests

Thanks to neural nets, no other machine learning method needs to worry about being the most overhyped technique. However, it remains true that random forests are the subject of some fairly grand claims.

For example, Breiman and Cutler claim on their random forests website that "[r]andom forests does not overfit." This claim is false; any learning method can overfit. See the example included in the RMarkdown document accompanying this paper for a simple case of overfitting by random forests. However, even in this pathological case which is designed specifically to cause overfitting, it remains true that the OOB error is a roughly unbiased estimate of the generalization error. Thus, even in a case in which random forests overfits, that overfitting may be easy to detect by comparing the OOB error to the training set prediction error.

The claim that random forests do not overfit has a grain of truth behind it. Namely, random forests do not overfit *by virtue of growing too many trees.* Figure 2, from Hastie et al. (2009), shows a typical example of how test set error changes as a result of increasing B, the number of trees used in the forest. This simplifies the process of tuning random forests, since one need worry only about having "enough" trees, and not about having "too many."

Another claim made by Breiman and Cutler on their website is that, as a classifier, random forests are "unexcelled in accuracy among current algorithms". This is a difficult claim to assess (what exactly would constitute a demonstration of such a claim?), though Figure 2 is one of several examples that Hastie et al. (2009) describe in which random forests are outperformed by gradient boosting.

However, a reasonable stab at a comprehensive examination of the question of which classification algorithm is most accurate is undertaken by Fernández-Delgado et al. (2014). The authors evaluate 179 classifiers using 121 data sets. The authors find that the best performers on these 121 data sets are random forests, though their performance is not statistically significantly superior to that of support vector machines. Still, this study provides at least a substantive piece of support for the rather grand claim on Breiman and Cutler's website.

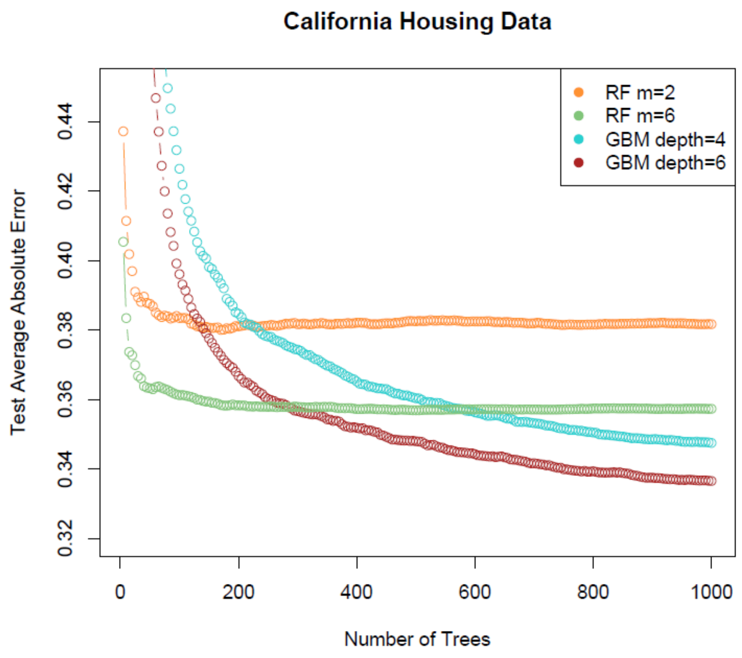

Figure 2: Test error of two different random forest models (and two different gradient boosting models). Notice that the random forest test error appears to converge to its minimum.

4 Application

In this section I describe my application of random forests as a binary facial recognition model, which distinguishes whether a face belongs to myself or to my wife. The resulting model uses a very low number of covariates – only thirty. The model is lightweight enough to operate in real time on webcam input, displaying the classification in real time as an overlay over the webcam image. It can furthermore classify multiple faces in the same image simultaneously.

To interact with my laptop's webcam, I employ the OpenCV Python library (Bradski, 2000). To detect faces within the images gathered from the webcam, I employ the Dlib Python library (King, 2009) in conjunction with the facial landmark database provided by Sagonas et al. (2016). The Dlib-based Python approach I use is inspired by the drowsiness detection application described by Rosebrock (2017). The result of training Dlib's facial landmark detector on that data set is to produce a facial detector which returns the 2-D locations, within an image, of the 68 landmarks shown in Figure 3.

Using these 68 landmarks, I define 30 covariates. Each covariate is either a length (e.g. width of the lower jaw), an area of a triangle defined by three landmarks (e.g. each of the two sides of the nose), or an angle formed by three landmarks (e.g. temple to mouth to lower jaw). All covariates are measured after scaling the face to have length and width 1. For the precise definitions of the 30 covariates, see the Python code included as a supplement to this paper. Figure 4 shows an overlay which gives a sense of the covariates collected from each facial image. I collected 500 training samples each from myself and from

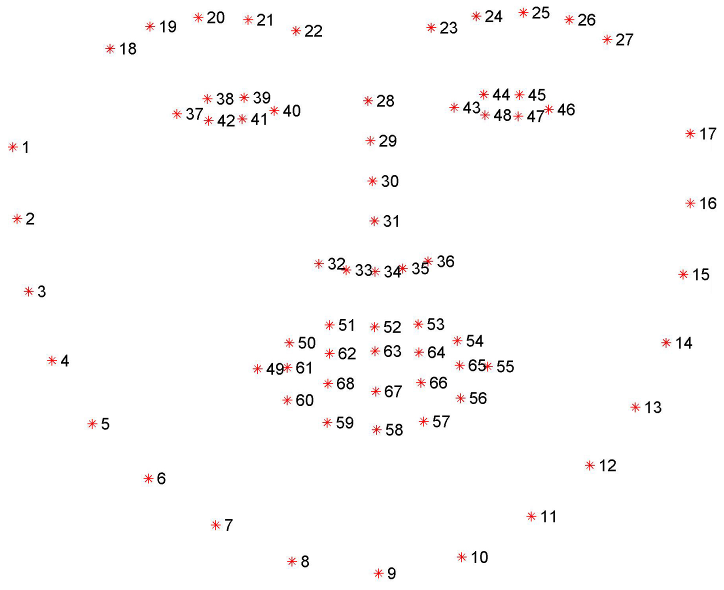

Figure 3: 68 facial landmarks. Image credit: Sagonas et al. (2013).

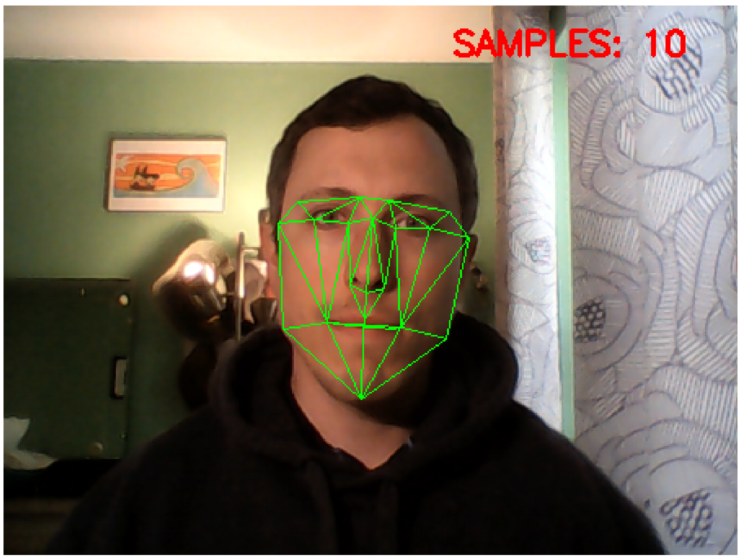

Figure 4: An image taken from the training samples of my face.

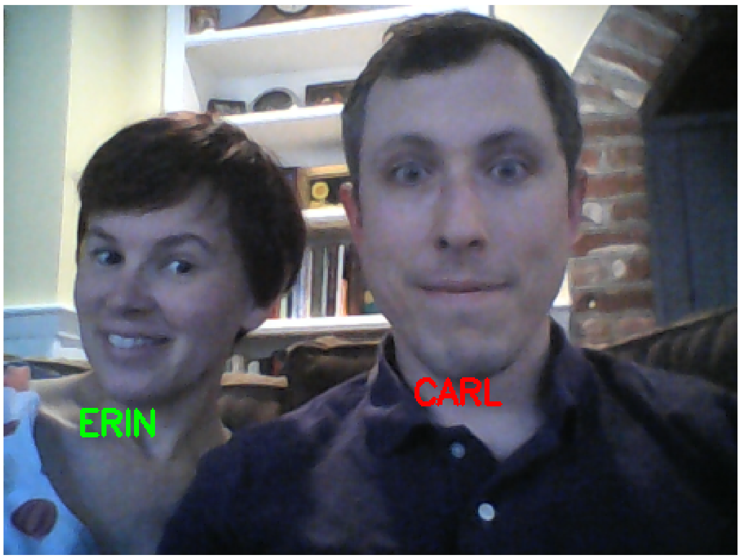

Figure 5: An example of the classifier in action, labelling multiple faces simultaneously on a real-time webcam feed.

my wife.

To fit the model, I used the implementation of random forests in the Scikit-learn Python library (Pedregosa et al., 2011). Default settings worked well, generating 6% OOB error. After tuning, the model reached 3% OOB error. The tuned model deviates from Scikit-learn's defaults only by increasing the number of trees from 10 to 100. Setting maximum tree depths and changing the number of predictors used in random feature selection (from the default of $\lfloor\sqrt{p}\rfloor = \lfloor\sqrt{30}\rfloor = 5$) appeared to have either no effect, or deleterious effects upon the model.

The resulting model performs well on new webcam input, with an error rate similar to that suggested by the OOB error. The classifier is fast enough to operate in real time on webcam input. An example of the classifier in action is shown in Figure 5.

As described in Section 2.2, random forests can be used to estimate the relative importances of the predictors used in arriving at the classification predictions. The feature importances from the facial classifier are shown in Figure 6. An unexpected result is that inner brow width (the distance between the eyebrows) is by far the most important covariate distinguishing my face from my wife's. Lower jaw width is less surprising, but outer mouth height (the distance from the top of the upper lip to the bottom of the lower lip) is somewhat strange to find so high on this list. My suspicion is that this is an artifact of the data collection process. When I collected training samples from my own face, I was alone. When I collected samples from my wife's face, we were conversing. Therefore, the importance of outer mouth height is, I suspect, simply due to the fact that she was speaking in many of her training samples, whereas I was not.

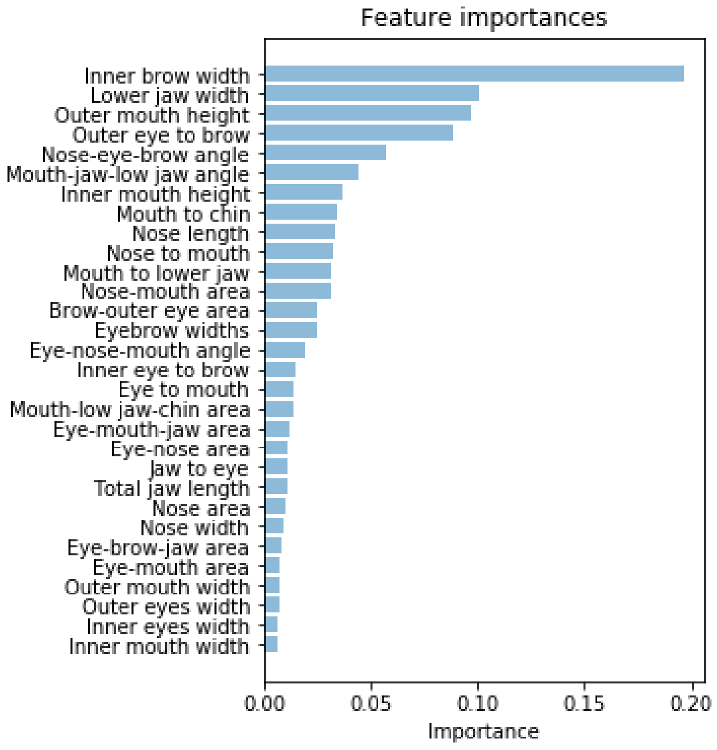

Figure 6: Relative feature importances in the facial classifier.

5 Conclusion

Random forests are a robust method for regression and classification that work well with minimal tuning. They are resistant to (but not immune from) overfitting, and where they have overfit, this is generally detectable by relying on the OOB error. The OOB error provides an estimate of the generalization error without requiring a test set or cross-validation. Random forests sacrifice some of the interpretability of single trees, but they can still be used to explore the relative importances of the covariates used in the model. Random forests can be used to build a lightweight binary facial classifier that can operate in real time on webcam input with high accuracy.

It would be relatively easy to extend this application to more than two faces. It remains to be seen to what extent accuracy would suffer by doing so. It would also be interesting to explore different sets of covariates. Examination of Figure 2.2 suggests that several covariates could be removed without undermining the model. It may also be profitable to add covariates not currently considered by the model. A useful approach would be to define a great many covariates, and then use random forests to estimate the feature importances of these covariates for variable selection.

References

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools.*

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Breiman, L. and Cutler, A. (2019). Random forests.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification And Regression Trees.* Routledge.

Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D., and Fernández-Delgado, A. (2014). Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? Technical report.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, 2 edition.

King, D. E. (2009). Dlib-ml: A Machine Learning Toolkit. Technical report.

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.

Rosebrock, A. (2017). Drowsiness detection with OpenCV.

Sagonas, C., Antonakos, E., Tzimiropoulos, G., Zafeiriou, S., and Pantic, M. (2016). 300 Faces In-The-Wild Challenge: database and results. *IMAVIS*, 47:3–18.

Sagonas, C., Tzimiropoulos, G., Zafeiriou, S., and Pantic, M. (2013). 300 Faces in-the-Wild Challenge: The first facial landmark localization Challenge. Technical report.

Tin Kam Ho (1995). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition.*