Pruning and post-fitting ensembles

Additional material for the book
Statistical Analysis Techniques in Particle Physics:
Fits, Density Estimation and Supervised Learning
(Wiley-VCH, 2013)
by Ilya Narsky and Frank C. Porter

December 16, 2013

Section 14.4 of the book discusses pruning decision trees. Similarly, weak learners can be often removed from an ensemble without a significant loss of the predictive power. Simplifying an ensemble can make it more robust to overfitting and reduce the memory footprint. This simplification is called *pruning*, *regularization*, or *shrinkage* in the literature.

To prune, one could inspect the learning curve on cross-validated or test data and chop off learners above a certain index. For example, the learning curve for boosted decision stumps in Fig. 15.3 shows some degradation past 50 stumps. The 51st and later stumps can be removed.

A related technique is *post-fitting*. After an ensemble is constructed, coefficients (weights) of the weak learners are optimized using a certain criterion. This criterion may, but is not required to, be identical to the loss minimized by the ensemble such as, e.g., the exponential loss for GentleBoost.

Pruning and post-fitting can be carried out at once. After the learner weights are (re)optimized, some of them can take small values. The respective learners can be then omitted from the ensemble.

Post-fitting a criterion (loss) motivated by physics can be an attractive way of improving the ensemble performance. Suppose the analysis aims at maximizing $s/\sqrt{s+b}$, where s and b are the expected signal and background. The analysis could then proceed in two steps. First, an ensemble could be grown by minimizing the exponential loss. Second, the signal significance could be maximized by searching for the best threshold on the soft score predicted by the ensemble. The second step could be replaced by simultaneous optimization of the learner weights **and** threshold.

An easy problem arises if the criterion of interest is a convex function of the learner weights with analytic expressions for the gradient and Hessian. Typically, criteria motivated by physics satisfy neither of these conditions. Because an ensemble often has a few dozen to a few thousand weak learners, the problem is high-dimensional. The learner weights must be non-negative and, since multiplying all weights by the same factor cannot change the ROC curve, must satisfy a constraint on their sum. Large-scale derivative-free optimization of a non-convex function with parameter bounds and linear constraints is hard.

[ZWT02] assign a random weight to every learner in an ensemble and evolve these weights using a genetic algorithm. An individual in the evolving popu-

lation is a weight vector $\boldsymbol{\alpha} \equiv \{\alpha_t\}_{t=1}^T$ with $0 \leq \alpha_t \leq 1$. [ZT03] fix the weight vector $\boldsymbol{\alpha}$ and evolve a vector of bits representing inclusion in the ensemble prediction. Other plausible techniques are simulated annealing and pattern search, although we are not aware of their applications to ensemble pruning. These algorithms allow optimization of an arbitrary criterion, but convergence of these algorithms for large-scale problems can be poor. Generally, any improvement in the criterion over the value computed for the initial weight assignment should be considered as success.

An alternative approach is pruning by a computationally simple heuristic algorithm. [MD97] plot average error against interrater agreement for all pairs of weak learners in an ensemble and find a convex hull of points closest to the origin. Learners contributing to the points on this hull are retained, and the rest is discarded. Although fast and intuitive, this algorithm can lead to a significant loss of the predictive power. Other heuristic algorithms of this type can be found in [Kun04] and [Rok10].

In our experience, the most reliable way of pruning ensembles is by minimizing a convex, twice differentiable loss with a penalty term. In the book, we refrain, with few exceptions, from discussing regression models. Here we mention regression ensembles because they allow for efficient regularization. For binary classification, signal can be coded as 1 and background can be coded as 0. A regression ensemble such as random forest [Bre01] or LSBoost [Fri01] can be then grown by minimizing the mean squared error (MSE). The ensemble prediction is computed by averaging predictions from individual regression trees, $f(\boldsymbol{x}) = \sum_{t=1}^T \alpha_t h_t(\boldsymbol{x})$. Class labels can be predicted by introducing a threshold on the regression response: Assign an observation to class 1 if the response is above 0.5 and assign to class 0 otherwise. The penalized MSE loss is then

$$\tilde{L}_{\text{mse}} = \sum_{n=1}^N w_n \left(y_n - \sum_{t=1}^T \alpha_t h_t(\boldsymbol{x}_n) \right)^2 + \lambda \sum_{t=1}^T \alpha_t \tag{1}$$

for a non-negative parameter λ and non-negative learner coefficients α_t. Above, w_n are observation weights adding up to one, $y_n \in \{0, 1\}$ are true class labels, and $\boldsymbol{x}_n$ are points in the training set. Penalizing on the L_1 norm of the vector of coefficients is called *lasso* regularization. By duality, minimizing $\tilde{L}_{\text{mse}}$ is equivalent to solving

$$\min_{\boldsymbol{\alpha}} \sum_{n=1}^N w_n \left(y_n - \sum_{t=1}^T \alpha_t h_t(\boldsymbol{x}_n) \right)^2 \quad \text{subject to} \quad \sum_{t=1}^T \alpha_t \leq c. \tag{2}$$

The non-negative parameter c decreases as λ increases. The geometry of the lasso constraint is a simplex with vertices at $+c$ and $-c$ along each coordinate. As c decreases, the coefficients are shrunk toward zero with some being set exactly to zero due to the simplex geometry. The fraction of zero coefficients increases as c continues to decrease. The lasso penalty can produce a highly sparse solution without a significant loss in the predictive power. An efficient coordinate-descent algorithm for minimization of $\tilde{L}_{\text{mse}}$ is described in [FHT10].

The lasso penalty can be applied to the exponential loss as well. A generic tool for constrained optimization such as the `fmincon` utility available from the Optimization Toolbox in MATLAB can be used, if the data are not too big, to

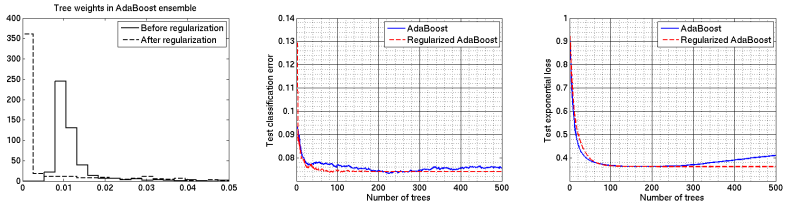

Figure 1: Distributions of tree weights (left), test classification error (middle) and test exponential loss (right) before and after lasso regularization.

solve

$$\min_{\boldsymbol{\alpha}} \sum_{n=1}^{N} w_n \exp\left(-y_n \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}_n)\right) \quad \text{subject to} \quad \sum_{t=1}^{T} \alpha_t \le c \tag{3}$$

for class labels $y_n \in \{-1, +1\}$.

Regularization of the exponential loss is demonstrated using the K/π BaBar data described in Section 15.2.4. An ensemble of 500 trees with minimal leaf size 1000 is grown by AdaBoost with learning rate 0.1. This ensemble gives 7.6% test error. The optimization problem (3) is solved numerically by `fmincon`. Parameter c is set to one half of the L_1 norm of the initial coefficient vector, $c = \sum_{t=1}^{T} \alpha_t^{(0)}/2$. The ensemble is regularized using the same data on which it is trained. Due to regularization, the exponential loss measured on the training data increases from 0.23 to 0.30.

Distributions of the tree coefficients before and after regularization, and the respective learning curves are shown in Fig. 1. The narrow peak in the histogram for the regularized coefficients indicates a large fraction of the trees with weights close to zero. To display the learning curves after regularization, we sort the weak learners (trees) by their regularized coefficients in descending order. The improvement in the test error and exponential loss is slight. Almost the same performance could be obtained by stopping AdaBoost after 100 iterations. Notice however that regularization prevents overtraining seen in the exponential loss curve at large iterations.

References

[Bre01] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[FHT10] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010.

[Fri01] J. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[Kun04] L. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, 2004.

[MD97] D. Margineantu and T. Dietterich. Pruning adaptive boosting. In *Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1997.

[Rok10] L. Rokach. *Pattern Classification Using Ensemble Methods*, volume 75 of *Machine Perception and Artificial Intelligence*. World Scientific, 2010.

[ZT03] Z.-H. Zhou and W. Tang. Selective ensemble of decision trees. In *Proceedings of the 9th international conference on Rough sets, fuzzy sets, data mining, and granular computing*, RSFDGrC'03, pages 476–483. Springer-Verlag, 2003.

[ZWT02] Z.-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artificial Intelligence*, (137):239–263, 2002.