DEEP BOOSTING — Wilks Statistics Seminar

LECTURE #1

OCTOBER 2, 2015

LECTURER: MEHRYAR MOHRI (COURANT INSTITUTE, GOOGLE RESEARCH) SCRIBE: KIRAN VODRAHALLI

Contents

1 Introduction

Professor Mohri is the co-author of Foundations of Machine Learning, and is on the advisory board of JMLR.

This work is joint with Corinna Cortes (Google Research), Vitaly Kuznetsov (Courant), Umar Syed (Google Research).

The title of this talk is Deep Boosting - why Deep? It does not have to do with deep learning. It also does not have to do with 'Deep Boosting Essence'. It is going to have to do with boosting, therefore let us start with some words about ensemble methods.

2 Ensemble Methods

These are very general methods in machine learning by combining base classifiers: These include bagging, AdaBoost, Stacking, other averaging schemes...

These in practice tend to be very effective. Very often, the winning schemes use an ensemble methods. These methods also have good theory. They favor from margin guarantees which typically hold for convex combinations over a base classifier set H. For instance, boosting stumps or decision trees with very limited depth. The ensemble combination is just elements of the convex hull of H (non-negative combinations).

Theorem 2.1. *Margin Bound.*
Let H be a family of real valued functions, fix $\rho > 0$. Then for any $\delta > 0$ with probability $1 - \delta$, for all

$f \in \text{conv}(H)$

$$R(f) \leq \hat{R}_{S,\rho}(f) + \frac{2}{\rho}\mathcal{R}_m(H) + \sqrt{\frac{\log 1/\delta}{2m}}$$

where $\hat{R}_m(H)$ is the Radamacher complexity.

So our question is, can we use a much richer or deeper base classifier set. Another question is about AdaBoost: coordinate descent applied to an objective function $F(\alpha) = \sum_{i=1}^{m} e^{-y_i f(x_i)}$. The ensemble margin bound is a justification for AdaBoost, but actually, it turns out that AdaBoost does not maximize the margin. There are some other algorithms that maximize the margin, but actually, these algorithms do not do as well as AdaBoost. This mystery is well-known in learning theory. One could ask is if maximizing margin is what we want to do, but we will come back to this question later. Suspicion number one is that in order to maximize margin the other algorithm actually uses deep trees; perhaps minimal margin is not the right thing to do.

The main question: Can we design algorithms that can succeed when using a complex base classifier set, like deep trees, or other complex hypothesis sets (family of polynomials of high degree, or any other family)? We go through theory, algorithms, results and model selection.

This talk is not just about boosting actually, but more about model selection (Voted Risk Minimization).

3 Theory

Suppose you can partition your complex base classifier set into families $H_1, H_2, \cdots H_p$. You could also decompose H into unions of hypothesis sets. Then when you look at a convex combination, each h_t falls into some H_i. If you could consistently pick h_t out of the most complex hypothesis set, then you would be prone to overfitting. What if you don't do that? What if you use complex hypotheses, but do so infrequently? That is a little bit the idea behind this work. It is to draw hypotheses for very complex H_is but do so by allocating less mixture rate to complex hypothesis sets. But how can we quantitatively determine the amounts of mixture weights to give.

Theorem 3.1. *Learning guarantee.*
The previous bound gives that the generalization error is upper bounded by empirical margin loss, the complexity of the full complexity of H. This term is remarkable since the complexity term explictly depends on the mixture weights α_t - it is an expectation of the Rademacher complexities. Fix $\rho > 0$.

$$R(f) \leq \hat{R}_{S,\rho}(f) + \frac{4}{\rho}\sum_{t=1}^{T} \alpha_t \mathcal{R}_m(H_{k_1}) + \tilde{O}\left(\sqrt{\frac{\log p}{\rho^2 m}}\right)$$

with high probability where f is a convex combination of hypothesis classes ($f = \sum_{t=1}^{T} \alpha_t h_t \in \mathcal{F}$). ρ is basically a measure of the complexity as well. This is an expectation bound in some sense and requires non-standard analysis of Rademacher complexity.

The bound serves as a quantitative guide for controlling weights assigned to more complex sub-families. This bound can be used to inspire a problem.

Here is the setup. We have $H_1, \cdots, H_p$ which are disjoint sub-families of functions taking values in $[-1, 1]$. We also assume (not necessary) that we have symmetric sub-families, i.e. $h \in H_k$ iff $-h \in H_k$. This just simplifies the presentation. Finally, $r_j = \mathcal{R}_m(H_{k_j})$, with $h_j \in H_{k_j}$. This is just the Rademacher complexity of H_{k_j}.

The learning bound suggests seeking $\alpha \geq 0$ with $\sum_{t=1}^{T} = 1$ to minimize the loss function. Let $u \to \phi(-u)$ be a decreasing convex function upper bounding the indicators. Then we move the constraint to the objective and use the fact that families are symmetric, this results in a convex objective. We can choose ϕ to be exponential loss (AdaBoost) or log-loss (logistic regression).

Then you apply coordinate descent to this convex objective. For coordinate descent, you have to describe direction and step. You can find direction by finding an expectation of an error, and you can get a closed-form for step-size, in general you can do line search. The program takes only 22 lines.

This algorithm tends to make things sparse.

4 Connections with Previous Work

In the case where you set $\lambda = \beta = 0$, DeepBoost coincides with AdaBoost if you use exponential loss. If you use logistic function, it coincides with additive logistic regression. When $\lambda = 0, \beta \neq 0$, DeepBoost coincides with l_1-regularized AdaBoost with exponential loss, and with logistic loss it is l_1-regularization of logistic regression. λ controls using the Rademacher complexities, β controls sparsity.

We naturally compare deep boosting with AdaBoost, l_1-regularized AdaBoost, Logistic Regression, and l_1-regularized Logistic Regression.

A nice thing about our bounds is that we can accurately estimate the Rademacher complexities and use them as part of the algorithm. For kernel methods, you can give an upperbound on Rademacher complexity using the trace of the kernel function. Alternatively, you can use VC-dimension or growth function (growth function of hypothesis set is maximal size of set that can be fully shattered - it is a function of the sample size) as an upper bound.

5 Experiments

For instance for boosting stumps, you can easily give growth function bounds. For the datasets, we can use datasets as favorable as possible to AdaBoost: UCI Irvine dataset, OCR dataset, and some MNIST datasets. Here we just use exponential loss and compare with the variants of AdaBoost. It is typically better. For more complicated decision trees, you can also give an upperbound on Rademacher complexity. Here we do all the comparisons. We are also finding λ, β via cross-validation (as with AdaBoost). We cross-validate and pick best depth for AdaBoost as well. At Google, we have run with millions of examples and seen similar improvements. People have reported similar improvements at other companies. We also compared about gradient-boosting, it is doing much better than that for classification. I have some concerns about the theory behind gradient-boosting as well.

6 Multi-Class Learning Guarantee

We have proven a tight bound in the number of classes for multi-class learning. Fix $\rho > 0$, then with high probability for f in the convex hull of H:

$$R(f) \leq \hat{R}_{S,\rho}(f) + \frac{8c}{\rho} \sum_{t=1}^{T} \alpha_t \mathcal{R}_m(\Pi_1(H_{k_t})) + \tilde{\mathcal{O}}\left(\sqrt{\frac{\log p}{\rho^2 m}}\right)$$

These have similar results on experiments, same flavor as before. Similarly we get results on multinomial logistic regression.

These ideas are connected to other algorithms, like structural maxent models (Cortes, Kuznetsov, MM and Syed, ICML 2015): feature functions are chosen from very complex families. Also related to Deep Cascades.

7 Model Selection

In a way, there is a more general problemt hat one wants to explore. How to select hypothesis set H? H too complex, you don't get a generalization bound and you get overfitting. If you get a bound, it might be insufficiently complex. So you have to balance between estimation and approximation errors.

Structural Risk Minimization, proposed by Vapnik tells you that if you had a family of hypothesis sets, you want to believe the base classifier is in one of those guys. If an oracle told you which one, then you could just go to that one. You can minimize empirical error plus a penalty term (function of hypothesis complexity and sample size). Then you can give nice generalization bounds for structural risk minimization, and you are better off if you know the base classifier falls into that hypothesis set. This forces you to commit to one hypothesis set. What happens if you don't do that? That is essentially the **Voted Risk Minimization** setup. Don't commit to a hypothesis at the start: Instead, all yourself to consider hypothesis functions that make use of all of those hypotheses potentially, and use the penalty depending on how much you use each of those hypothesis sets, and you can get much better guarantees than those you get in structural risk minimization.

8 Conclusion

In short, I presented a new way of doing ensemble algorithms using increasingly complex families. The analysis is data dependent, it drives the algorithm, good empirical results, many existing algorithms that can be enhanced directly in this way (maxent, decision tree, etc.). They compare favorably with AdaBoost and Logistic Regression and their l_1-regularized variants (they are a generalization of these). You can also generalize to the ranking problem.

9 Questions

Elad Hazan: Generalization error bounds do not suffer by number of hypothesis classes, however complexity would have to go over all to find weak learner. In Vapnik you would stop at some point

Response: But you don't know at one point. You can only say you stop at that very far point in the horizon if you have 0 empirical error; otherwise how do you know that you have to stop? There is no guarantee on how far you have to go. That is an important part of this work left to explore a lot more: How to explore this large space in an efficient way. I am claiming you can search efficiently. In the method here, we are using a heuristic search because we cannot explore everything. But those heuristics are what have been used in the past for AdaBoost. In the past, AdaBoost combined with those heuristics were beating everything we are aware of - including neural networks.

Han Liu: In the regression setting, I say the final predictor will be a linear combination of the predictors in your dictionary - then what kind of theory can you establish in terms of prediction? Then the prediction performance will be comparable to the best one in your ensemble, then you may not be much better than the best one. Your result says your prediction will be a little better performance to the best one. But why not do cross-validation?

Response: This is true - but we are talking about a very large ensemble here. Thus, since you can make that hypothesis very broad, your approximation error may be very small - if you are doing as best in class, you are doing pretty well. Cross-validation takes a really long time on a large hypothesis class.

There is nothing you can do about approximation error that I know of. The only thing one can do in life is either give up, or what can I do if I make the hypothesis set grow a lot with the hope that generalization error is small, and this is what the work is trying to do, and work with something very broad and still manage it.

Samory: In which sense is this averaging guaranteed to yield something better than structural risk minimization?

Answer: I wrote generalization bounds for this question. In a way, if you like, the solution is not cross-validation because your hypothesis is size.

Elad: The issue is you have limited resources - you need to do better optimization.

Samory: I want to know that with high probability that this algorithm always does better than structural risk minimization - is this uniformly for every case for a large enough interesting set of problems?

Professor Mohri: Suppose you get a complicated hypothesis H, how do you partition it into the cases? I don't know the answer, but it is interesting as a question.