BagBoo: Bagging the Gradient Boosting

3rd in RR (track I and II)
1st in nDCG (track II) and 2nd in nDCG (track I)

Dmitry Pavlov and Cliff Brunk

aka JOKER

aka team_404

Yandex Labs (labs.yandex.com)

{dmitry-pavlov,cliff}@yandex-team.ru

Yandex

- Yandex.ru Yandex.com
- Leading search engine in Russia w/65%+ search market share
- Labs office in Palo Alto, CA
- We are doing a lot of technology innovations
- … and We are hiring!
- Come chat with us or send a message

Я

Bagging (Breiman)

- Ensemble of models
  - Sampling data
  - Voting models
- Random Forest
  - Models are functions of iid random vectors
  - Assume Model = Tree WLOG from now on
- Nice properties
  - Variance reduction
  - Resistance to overfitting
  - Efficient parallelizable computation

Gradient Boosting (Friedman)

- Ensemble of models
  - Each next model learned to optimize residual error
  - Weight in the linear combination are optimized
  - Randomness/Stochastisity similar to Bagging
- Nice properties
  - Bias reduction
- Hard to parallelize

BagBoo: combined Bagging and Boosting

- Combine the best of both worlds:
  - get highly parallelizable algorithm with bias and variance reduction properties

1. **Input***:* Training data *D*, *Nbag* and *Nboo* iter.
2. **Output***:* Random Forest of *Nbag* x *Nboo* trees
3. **For** i=1 *to Nbag* do

   D[i] := *SampleData(D)*; # samp. feats and records

   BT[i] := *BoostedTree(D[i], Nboo)*;

   EndFor
4. **Output:** additive model \sum_i { BT[i] }

BagBoo: highly parallelizable algorithm

- Accurate
  - Excellent results in contests and on TREC benchmarks
- Fast
  - Can train many trees fast
- Gotchas
  - need to control learning rate
  - winning the contest with many trees is great but can be unrealistic in practice
- The idea has been studied before
  - KDD Cup'09,
  - P. Melville, R. Mooney et al,
  - Daria Sorokina's Additive Groves

Data Used for BagBoo Evaluation

Data Set	Queries	N Rows	N Features	N Labels	Label Distr.
TD2004	75	74,146	64	0/1	1.5% 1's
MQ2007	1,692	69,623	46	0/1/2	20% 1's, 6% 2's
IMAT2009	9,124	97,290	245	Multi 0..4	26% non-0s
Yahoo! Chal.	19,944	473,174	704	Multi 0..4	26% 0s

Performance of BagBoo on standard IR/TREC benchmarks

Table 2: Average cross-validated $NDCG1-5$ and MAP for TD2004 data set in LETOR3.0 collection. Numbers in bold found represent the winning method for the metric in a given column. *BagBoo* wins in all metrics. BagBoo fits over 1.1 million trees, which pure boosting can only afford in estimated 48 days on a single CPU, hence boosting performance is not quoted.

Method	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5	MAP
BagBoo	**50.67**	**44.00**	**40.80**	**39.86**	**38.98**	**24.99**
Bagging	45.33	44.00	38.88	37.15	34.48	21.73
Boosting	-	-	-	-	-	-
BoltzRank	47.67	41.33	39.02	37.57	36.35	23.90
ListNet	36.00	34.67	35.73	34.69	33.25	22.31
FRank	49.33	40.67	38.75	35.81	36.29	23.88
AdaRank.NDCG	42.67	38.00	36.88	35.24	35.14	19.36
AdaRank.MAP	41.33	39.33	37.57	36.83	36.02	21.89
RankSVM	41.33	34.67	34.67	34.10	32.40	22.37

BagBoo vs Bagging vs Boosting: Performance and Time Tradeoff

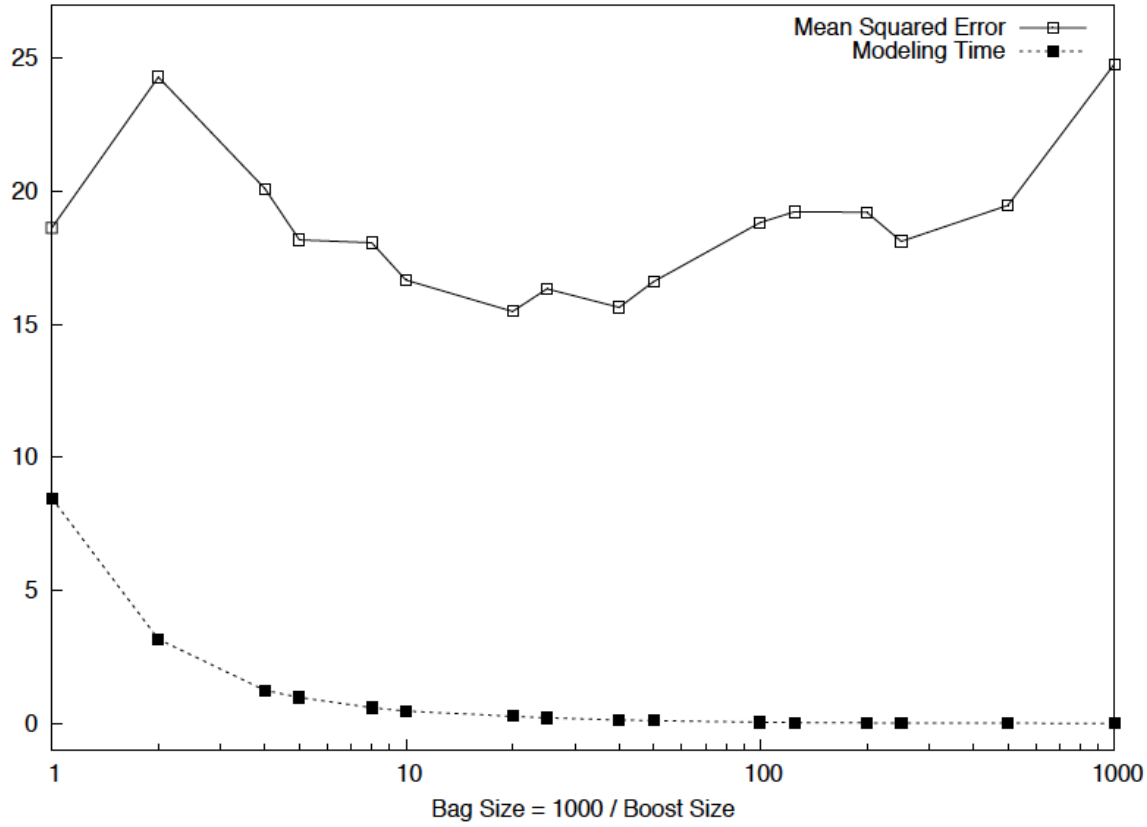

Figure 1: Model accuracy (mean squared error) and offline/modeling time as a function of $NBag$ and $NBoo$ for fixed $T = NBag \cdot NBoo$

Я

Dependence of performance on learning rate (thanks to G. Ridgeway)

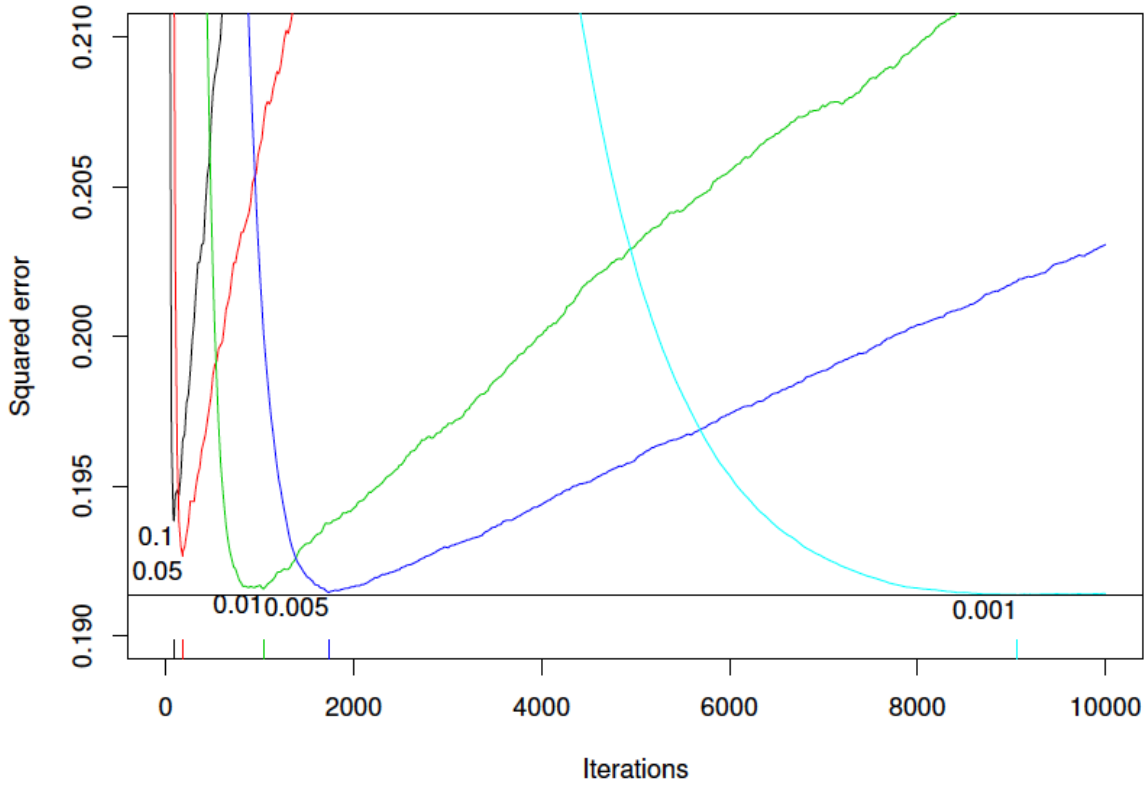

Figure 3: Out-of-sample predictive performance by number of iterations and shrinkage. Smaller values of the shrinkage parameter offer improved predictive performance, but with decreasing marginal improvement.

BagBoo params for ICML

- Bags=600
- Boosts=500
- Tree Depth=12 (!)
- Min_Leaf_Support=10
- Bag Feature Rate = Doc Rate = 100%
- Boost Feature Rate = 80% Doc Rate = 100%
- Learning rate=0.2 (!)
- 2nd track: transfer “from” data 1 : 7 “to” data weight

Я

Many things we tried that didn't quite work…

- Engineering new features, e.g. products of existing features
- Remapping the labels – which makes sense for a point-wise method
- Averaging results of various successful ideas
- Median vs Mean, removing the outlier trees
- Using track II data for track I
- Building different models for Navigational and Non-navigational queries

Cross-validation is a must

- Typical flaw: evaluation on the holdout sample
- Very easy to overfit
  - Especially if it is small
- Expensive
  - Yes! (unless you have a cluster)
  - But ultimately well worth it
  - Still affordable in 3 months time span the competition was ran

Grain of Salt

- Offline vs Online
  - Deal with the top N documents per query vs Billions in real-life
  - Offline results can be misleading (Web is 90%+ junk or irrelevant)
  - The model ranking 24 might turn out best in online retrieval

Acknowledgements = Big Thanks To

- Organizers for a wonderful challenge
  - and giving us internal ID team_404 ;-)
- Yandex team for support, discussions
  - and enduring our hogging the cluster
- All of you for coming and listening