T-BAG: Bootstrap Aggregating the TAGE predictor

Burak Karsli and Resit Sendag
(bkarsli, sendag)@ele.uri.edu

***Abstract*— TAGE [1] predictor has been widely accepted as the current state-of-the-art in branch prediction. In this work, instead of directly improving predictor's accuracy, we focus on an orthogonal statistical method called bootstrap aggregating, or *bagging*. Bagging is used to improve overall accuracy by using an ensemble of predictors, which are trained with slightly different data sets. Each predictor (can be same or different predictors) is trained using a resampled (with replacement) training set (bootstrapping). Then, the final prediction is simply provided by weighting or majority voting (aggregating). Our submitted predictor achieves 1.919 misp/KI.**

I. BOOTSTRAP AGGREGATING

Bootstrap aggregating (a.k.a, *bagging*), introduced by Breiman [2] in 1996, is a meta-algorithm to improve the stability and accuracy of learning algorithms. It has been shown to be very effective in improving generalization performance compared to individual base models [3]. Bagging is special case of having a hybrid predictor, where predictions from multiple predictors are aggregated using meta-predictors, adder-trees, voting, etc. Bagging works by resampling (with replacement, i.e., some samples may be used more than once) the original training set of size *N* to produce *M* bootstrap training sets of size *N*, each of which is used to train a base model. The predictions by each base model are then aggregated to reach the final prediction. The bagging method is shown in Figure 1. Each predictor's training set contains each of the original training samples *K* times, where

$$P(K=k) = \binom{N}{k}\left(\frac{1}{N}\right)^{k}\left(1-\frac{1}{N}\right)^{N-k} \qquad \text{(Eq. 1)}$$

In this work, we applied bagging to branch prediction. Because original bagging method is offline – that is, all the training data set must already be available –, we need to develop an online version of bagging. Previous work by Oza and Russel [4] modeled sequential arrival of the data by a Poisson(1) distribution and proved the convergence of this method to offline bagging as N→∞. We first used their method in our implementation, which improved performance most of the time. However, we observed that multinomial distribution worked better and hence this method was used in our submission. The situation is more complicated for branch prediction data because bootstrapping must be carried out in a way that suitably captures the dependence structures for the data. Oza and Russel's [4] method assumed that samples were independent of each other, and thus it does not produce good bootstrapping for branch prediction data. There are studies that developed methods for bootstrapping time series [5], which are better fit for branch prediction. Further research is needed to develop better online bootstrapping methods for branch prediction or adopt methods from previous work on bootstrapping for time series data, which is left as future work.

In our bagging implementation, each predictor is updated on each sample *k* times in a row where *k* is a random number generated by multinomial distribution. We illustrate online bagging in Figure 2.

In general, bagging can be applied to any predictor. Group of same predictors (e.g., a number of TAGE predictors) as well as different predictors may be used. Because bagging works better when large number of predictors is used, we target the unlimited-budget track for our submission to CBP-4.

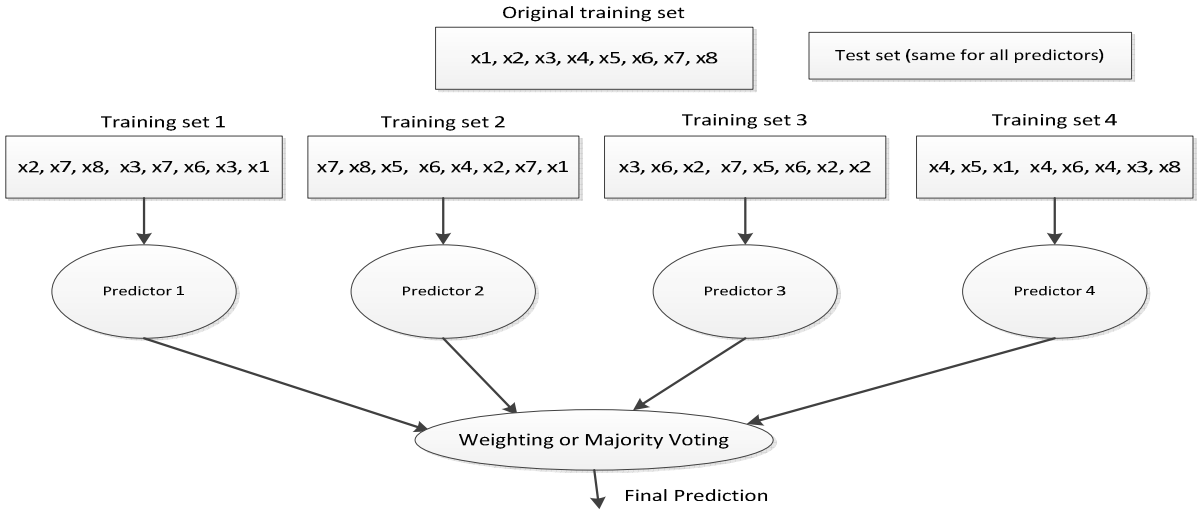

Figure 1: Offline Bagging creates M training sets, one for each predictor, from an original training data set by resampling with replacement. Each predictor then is trained by the bootstrap training set independently. The overall prediction is given by majority or weighted voting.

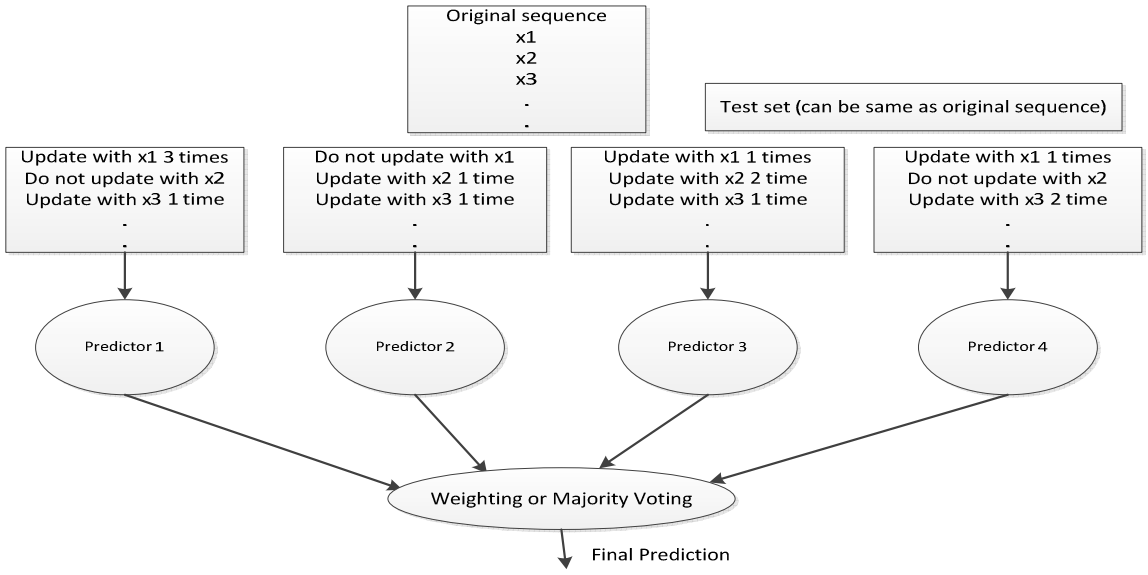

Figure 2: Online Bagging. Each predictor is updated *k* times when presented with a sample.

II. TAGE PREDICTOR

TAGE [1] is the state-of-the-art in branch prediction. It relies on several predictor tables indexed through a hash of increasing global history, path history and branch address. The set of used global history lengths forms a geometric series, which allows using very-long branch histories.

ISL-TAGE [6] was the winner of CBP-3. For this submission, we have used the LSC-TAGE [7] as our baseline predictor since it is currently the best performing[1] predictor. The baseline size that we have used for our submission is about 15MB. The performance does not significantly improve to justify increasing the size further. We have found that 38 tagged tables with history sizes up to 100,000 bits perform the best and therefore it is our baseline. Using 29 tagged components with history sizes up to 5,000 perform very similar. In our baseline, we use a relatively large Statistical Corrector of 672KB although making it 4x to 8x smaller does not significantly degrade the performance. Furthermore, on a misprediction, allocating all available entries was slightly better. Finally, we use 16 bits for as the smallest tag size and 30 for largest. Other parameters are untouched: they are the same as the original LSC-TAGE configuration.

Although, TAGE is the only predictor used as sub-predictor in T-BAG, the bagging method described here is equally applicable to other predictors or to a mixture of predictors.

III. T-BAG

T-BAG uses a number of TAGE predictors (32 for the submission) of approximately the same size (~15MB each for the submission) as sub-predictors. Each sub-predictor provides prediction for the current branch independent of each other. Online bagging is performed by determining whether or not a sub-predictor is updated with the current branch's outcome. This update decision only involves counter update in the sub-predictor tables. Note that this update may occur multiple times for the current branch based on a random number generated as described in Section 1. The branch history, however, is always updated as usual.

Final Prediction Computation: For each sub-predictor, T-BAG remembers the success of its last 16 predictions using a sliding window. The number of correct predictions is used as the weight of the sub-predictor. For a not-taken prediction, the weight is taken as negative and for taken predictions it is positive. The overall T-BAG prediction is the sign of the sum of the weights, negative being not-taken and it is taken otherwise. This method was slightly better than using majority vote for the final prediction and therefore it is used for the submitted T-BAG.

Sub-predictor configurations: Each sub-predictor can be configured differently or the same. We have used the counter width, the number of tagged tables, table indexing with and without PC, and the minimum and maximum history sizes as the four parameters to configure sub-predictors. We have used a fixed total size of about 15MB for each sub-predictor, however. That is, the number of table entries for a 38 component predictor is half the number of entries for a 20-component predictor for most of the tables. For our submitted predictor, counter width is fixed as 3 bits[2]. The minimum history size varies between 5 and 13. The maximum history varies between 1000 and 100,000. Finally, the number of tagged table components in each sub-predictor varies between 20 and 38. When allocating tagged table entries on mispredictions, for such a large predictor,

[1] In our experiments, unlimited size LSC-TAGE was only slightly better than unlimited size ISL-TAGE.

[2] Using 4 bits for the shorter history length tables (<15) and 3 bits for the rest of the tables perform slightly better for a sub-predictor alone, but provides no benefit in T-BAG.

allocating all available entries is slightly better than allocating 3 entries.

Bagging the same sub-predictor: As we mention above, one could use the same configuration for all the sub-predictors. We call this version, *AllSame*. In this case, the only variability in sub-predictor predictions comes from the random updates. In this configuration, the sub-predictor parameters that we have used are: counter width = 3, number of tagged tables = 38, allocation policy = all allocate, the minimum and maximum history lengths = 7 and 100,000, respectively.

Random Updates: In all our random update, *RandUpd*, simulations, table counter updates are performed randomly for 0, 1, or 2^3 times in a row for 20%, 60% and 20% of the time, respectively, using trinomial distribution. That is, 60% of the time update is done as usual, 20% of the time no update is performed and 20% of the time update is done twice in a row.

Use of PC: The original TAGE also uses the PC when forming the hashed index for its tagged components. However, because of its operation and its ability to exploit very long history lengths, the PC does not significantly affect performance. In our experiments, the best TAGE configuration using PC in table indexing and the one that does not use PC achieve the same performance (2.003 misp/KI). Therefore, for the submitted T-BAG, to further increase variability among sub-predictors, **half of the sub-predictors do not use the PC when indexing tagged tables**. To the best of our knowledge, no previous work has studied the effects of not using PC in table indexing.

The number of sub-predictors: Increasing the number of bagged predictors improve accuracy. However, due to the maximum simulation time requirements and restrictions about multithreading the predictor, our submitted predictor, T-BAG, uses 32 sub-predictors. The **total size** of the submitted predictor is **492 MB.**

The sub-predictor configurations in this submission are given in Table 1. All sub-predictors use counter width of 3, all-allocate on mispredictions, and are the same size. The best sub-predictor achieves 2.007 misp/KI alone.

IV. RESULTS

Table 2 shows the results for the TAGE and T-BAG per application. The average predictor accuracy for the submitted T-BAG is **1.919 misp/KI** (Bag32 in Table 2). With the same size TAGE predictor (32x), we obtain 2.003 misp/KI. The baseline-size (~15MB) TAGE predictor used in bagging achieves 2.007 misp/KI.

Figure 3 compares different T-BAG versions. *AllSame_RandUpd* replicates a single sub-predictor N times and aggregates the individual predictions of randomly updated sub-predictors to make the final prediction. *AllDifferent_RandUpd* uses 32 slightly different TAGE sub-predictors as shown in Table 1 and applies random update so that each sub-predictor is updated slightly differently. *AllDifferent* uses 32 different sub-predictors with ordinary updates. The final prediction computation method, as described in the previous section, is the same for all. From the figure, we can see that as the number of sub-predictors increases so as the performance, for all T-BAG versions. *AllSame_RandUpd* achieves **1.952 misp/KI** for 32 sub-predictors. Using random update when all sub-predictors are slightly different, *AllDifferent_RandUpd,* performs best achieving **1.919 misp/KI**. When random update is not applied, *AllDifferent* version achieves **1.932 misp/KI**. For smaller number of sub-predictors, e.g., < 8, *AllDifferent* version performs best, which suggests using more sub-predictors when random update is applied.

Table 1: Sub-predictor configurations

Sub-predictor	Number of Tables	Min history	Max history	Use PC ?
1	24	9	2000	Yes
2	32	7	30000	Yes
3	30	9	10000	Yes
4	29	6	5000	Yes
5	28	8	4000	Yes
6	27	10	3000	Yes
7	25	6	2500	Yes
8	38	5	100000	Yes
9	23	4	2000	Yes
10	23	5	1800	Yes
11	22	3	1600	Yes
12	22	8	1500	Yes
13	21	9	1400	Yes
14	21	10	1300	Yes
15	20	6	1200	Yes
16	20	7	1000	Yes
17	38	12	100000	No
18	32	10	30000	No
19	30	9	10000	No
20	29	11	5000	No
21	28	10	4000	No
22	27	13	3000	No
23	25	11	2500	No
24	24	12	2000	No
25	20	9	100000	No
26	20	10	85000	No
27	20	11	70000	No
28	20	13	55000	No
29	20	12	40000	No
30	20	8	25000	No
31	20	10	10000	No
32	20	7	8000	No

[3] The maximum number of updates can be higher if counter width is larger. For our simulations, counter width is 3, so we use 2 as the maximum number of updates.

Table 2: Per benchmark accuracy in misp/KI

Benchmark	Baseline TAGE	32x-size TAGE	T-Bag32
SHORT-FP-1	0.922	0.919	0.875
SHORT-FP-2	0.408	0.408	0.374
SHORT-FP-3	0.014	0.014	0.014
SHORT-FP-4	0.014	0.014	0.014
SHORT-FP-5	0.007	0.007	0.008
SHORT-INT-1	0.105	0.105	0.1
SHORT-INT-2	3.463	3.455	3.198
SHORT-INT-3	5.366	5.35	5.087
SHORT-INT-4	0.358	0.357	0.322
SHORT-INT-5	0.051	0.063	0.05
SHORT-MM-1	6.404	6.406	6.265
SHORT-MM-2	8.332	8.328	8.102
SHORT-MM-3	0.037	0.037	0.038
SHORT-MM-4	0.984	0.982	0.937
SHORT-MM-5	2.265	2.26	2.122
SHORT-SERV-1	0.607	0.606	0.577
SHORT-SERV-2	0.597	0.599	0.569
SHORT-SERV-3	1.838	1.834	1.718
SHORT-SERV-4	1.367	1.368	1.31
SHORT-SERV-5	1.262	1.261	1.204
LONG-SPEC2K6-00	0.845	0.844	0.79
LONG-SPEC2K6-01	6.375	6.356	6.201
LONG-SPEC2K6-02	0.227	0.227	0.209
LONG-SPEC2K6-03	0.112	0.113	0.109
LONG-SPEC2K6-04	6.346	6.339	6.084
LONG-SPEC2K6-05	4.033	4.017	3.894
LONG-SPEC2K6-06	0.541	0.541	0.515
LONG-SPEC2K6-07	3.49	3.455	3.183
LONG-SPEC2K6-08	0.555	0.55	0.533
LONG-SPEC2K6-09	2.657	2.64	2.478
LONG-SPEC2K6-10	0.423	0.422	0.386
LONG-SPEC2K6-11	0.319	0.318	0.304
LONG-SPEC2K6-12	10.84	10.864	10.689
LONG-SPEC2K6-13	3.908	3.897	3.584
LONG-SPEC2K6-14	0.001	0.001	0.001
LONG-SPEC2K6-15	0.18	0.18	0.162
LONG-SPEC2K6-16	2.55	2.55	2.427
LONG-SPEC2K6-17	1.66	1.643	1.557
LONG-SPEC2K6-18	0.003	0.003	0.003
LONG-SPEC2K6-19	0.797	0.8	0.77
AMEAN	**2.007**	**2.003**	**1.919**

V. CONCLUSION

Bootstrap aggregating (bagging) is a statistical method to improve the accuracy of predictors by reducing variance and overfitting. It is applicable to any unstable learning algorithm. In this work, we applied bagging to branch prediction. Our submitted predictor forms an ensemble of slightly different predictors each of which is updated with slightly different data.

Our results show that bagging shows promise as a future research direction. Although online bagging method used in this work provides a way to apply bagging to branch prediction, it assumes independent samples, which is not the case for branch history. Different online bagging methods may prove better and are subject to future research. Finally, our analysis was done by only using TAGE as our base predictor. It is possible to use multiple different predictors that use different methods for prediction.

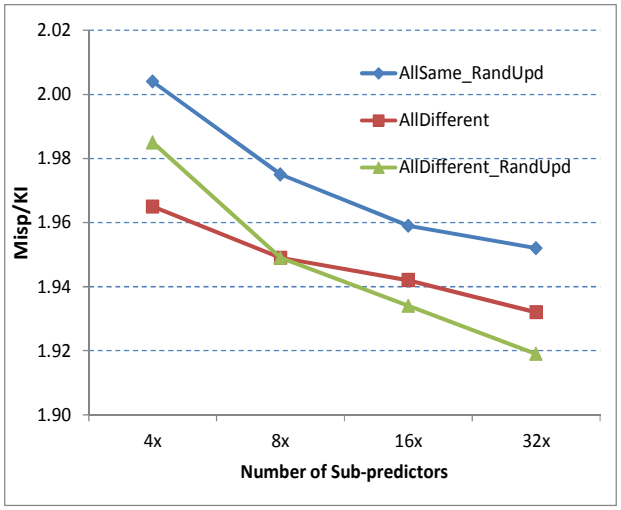

Figure 3: Comparison of different T-BAG versions.

REFERENCES

[1] Andre Seznec and Pierre Michaud, "A case for (partially)-tagged geometric history length predictors," Journal of Instruction Level Parallelism (http://www.jilp.org/vol7), April 2006.

[2] L. Breiman, "Bias, variance and arcing classifiers," Technical Report 460, Department of Statistics, University of California, Berkeley, 1996.

[3] E. Bauer and Ron Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting and variants," Machine Learning, 36:105-139, Sep. 1999.

[4] N. Oza and S. Russell, "Online Bagging and Boosting," Artificial Intelligence and Statistics, 2001, pp. 105-112.

[5] W. Hardle, J. Horowitz, J-C Kreiss, "Bootstrap Methods for Time Series," International Statistical Review, Vol. 71. No 2. Aug. 2003, pages: 435-459.

[6] A. Seznec, "A 64 Kbytes ISL-TAGE branch predictor," Workshop on Championship Branch Prediction Competition, CBP-3, 2011. http://www.jilp.org/jwac-2/program/JWAC-2-program.htm.

[7] A. Seznec, "A new case for the TAGE branch predictor," Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, Pages 117-127, 2011.