**DESCRIPTION**

The predictor is based on perceptron predictor [1]. There is a perceptron table and there is one bias table. Both these tables have 141 entries. There are 28 weights in each perceptron. Each weight has 8 bits. Hence weights have to be from -128 to 127.

To index into the perceptron tables and the bias table, the predictor uses the modulus operator.

The training algorithm is similar to that used in a normal perceptron. The tables are updated if the prediction is incorrect or if the absolute value of the calculated dot product is under a particular threshold.

I did some experimentation with threshold and also tried dynamic threshold. In dynamic threshold, if there are let’s say x number of continuous mispredicts, the threshold will be changed. But this did not give good results in all cases. Hence I settled for a fixed threshold. For choosing threshold, [1] cites that using (1.93\*h+14) gives good results where h is the length of GHR. The same has been used in the predictor with good results.

Size of Predictor = 141\*8\*(28+1) bits

= 32712 bits

**RESULTS**

On test traces**:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Trace** | **Alpha Predictor** | **Local Predictor** | **My Predictor** |
| DIST-INT-1 | 12.443 | 9.588 | 7.448 |
| DIST-INT-2 | 14.867 | 13.062 | 10.323 |
| DIST-FP-1 | 4.451 | 2.664 | 2.458 |
| DIST-FP-2 | 2.306 | 1.84 | 1.119 |
| DIST-MM-1 | 11.428 | 10.34 | 7.687 |
| DIST-MM-2 | 15.722 | 13.72 | 10.049 |
| DIST-SERV-1 | 17.245 | 11.634 | 7.778 |
| DIST-SERV-2 | 15.167 | 12.093 | 8.317 |
| **Average** | **11.703625** | **9.367625** | **6.897375** |

On SPEC benchmarks ([source](http://taco.cs.utsa.edu/camino/cbp2/cbp2-infrastructure-v2/traces/)):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Trace** | **Alpha Predictor** | **Local Predictor** | **My Predictor** | |
| bzip2.trace | 0.041 | 0.04 | | 0.028 | |
| vortex.trace | 0.662 | 0.543 | | 0.59 | |
| db.trace | 0.337 | 0.287 | | 0.192 | |
| vpr.trace | 0.644 | 0.697 | | 0.6 | |
| parser.trace | 2.144 | 1.93 | | 1.508 | |
| twolf.trace | 1.872 | 1.987 | | 1.524 | |
| eon.trace | 0.347 | 0.287 | | 0.186 | |
| compress.trace | 1.011 | 0.991 | | 0.685 | |
| mpegaudio.trace | 0.428 | 0.424 | | 0.32 | |
| crafty.trace | 1.113 | 1.127 | | 0.895 | |
| mcf.trace | 2.071 | 2.151 | | 1.795 | |
| gcc.trace | 3.824 | 3.728 | | 2.795 | |
| jack.trace | 1.7 | 1.506 | | 1.262 | |
| gzip.trace | 0.676 | 0.697 | | 0.57 | |
| javac.trace | 0.411 | 0.404 | | 0.35 | |
| raytrace.trace | 0.923 | 0.829 | | 0.778 | |
| gap.trace | 1.45 | 1.14 | | 0.774 | |
| mtrt.trace | 0.511 | 0.46 | | 0.426 | |
| jess.trace | 0.435 | 0.337 | | 0.192 | |
| perlbmk.trace | 1.289 | 1.073 | | 0.786 | |
| **Average** | **1.09445** | **1.0319** | | **0.8128** | |

**References**

[1] Daniel A. Jim´enez and Calvin Lin. [Dynamic branch prediction with perceptrons](http://www.cs.utexas.edu/~lin/papers/hpca01.pdf).

In *Proceedings of the 7th Int’l Symposium on High Performance Computer Architecture*, pages 197–206, January 2001.