AIM: Predict the baseline SPECfp score for a given input system configuration.

Result : 87% accuracy achieved (with an allowable error margin on 10%)

Motivation: Andrew Ng’s Deep Learning MOOC on Coursera

Authors: Santhosh Kumar Varma Patchamatla, Hrishikesh Satyavasu Murukkathampoondi

What exactly is SPECfp?

SPECfp is a computer benchmark designed to test the Floating Point (FP) performance of a computer and is managed by the Standard Performance Evaluation Corporation(SPEC) (<https://www.spec.org/>)

SPECfp2000 is a test suite on which the system configurations run and a score is computed. Refer to Wikipedia for an idea of how the overall SPECfp score is obtained based on the results of the testsuite.

Why SPECfp?

The results published in the SPEC website are official results, and companies publish the full system configuration that is used to obtain the SPECfp score. This is necessary so that any other company can take the full system configuration published in the website, and can reproduce the SPECfp score. So, for official documentation and reproducibility purposes, the full system configuration , including the baseline output and the peak output and specified.

Tools Used:

There were several empirical models that were developed to model the various SPEC scores, but this is probably the first time that a machine learning model is being developed to model the SPEC score. As opposed to empirical models which rely on assumptions to model the SPEC score, machine learning uses real historical data to model the SPEC score and hence, it is highly accurate.

There were 3 different prediction models that were developed, and all of them achieved approximately the same SPECfp score.

1. Prediction model based on Octave/MATLAB
2. Prediction model based on basic Python programming
3. Prediction model based on Tensor flow backend in Python

Who can use this model, and how?

This model can be used by the performance modelling teams to model and predict the SPEC scores. By using this model, the various performance teams can easily predict the effect of changing one parameter on the output performance number.

For example, to achieve a performance gain of X%, by how much should the L1Caches be increased? Or, by how much should the core frequency be increased? This model helps analyze this information at a very quick speed, and helps the performance teams in making a right decision based on historically available data, rather than heuristic models.

This model can also be used by the CPU architects to determine if a component should be increased or decreased (and by how much), to obtain a certain increase in the SPEC score. This helps them in determining the correct system configuration for a certain performance number.

Why SPECfp2000?

There are many SPEC benchmarks, SPECfp89, SPECfp92, SPECfp95, SPECfp2000, SPECfp2006, and SPEC2017. Initially, we decided to use the SPECfp2017 data, as it is the latest. But, when we analyzed the data, it turned out that all L1 entries are 32KB/core, and there are only 2 types of L2 entries 1MB/chip and 256KB/core. As this data doesn’t look diverse, and the general intuition makes us believe that the performance is very tightly linked to the cache sizes, we decided to defer creating the model for SPEC2017 and go with SPEC2000 instead ,which contains more diverse data.

Approach:

Data processing:

The first step in any machine learning problem is data processing. The data that is received as input from SPECfp cannot be used as is. It had to be formatted, and new entries had to be created so that the data can be comprehended properly. For ex: There is a field indicating the size of both L1ICache and L1DCache.This field had to be split into two (one for L1ICache, and another for L1DCache).

Another ex: There is a field indicating the DDR component type. Now, though the DDR component type cannot be used as is. We had to dig into google to find out the DDR size and DDR frequency, and add these two fields as input.

Model creation:

Machine learning is a highly iterative process. A preliminary model has to be built first, then the data has to be analysis further and the model needs to be tuned based on the results of the data analysis.

Armed with the above information, we created a basic linear model (give formula here), and we were able to achieve an accuracy of 35%.

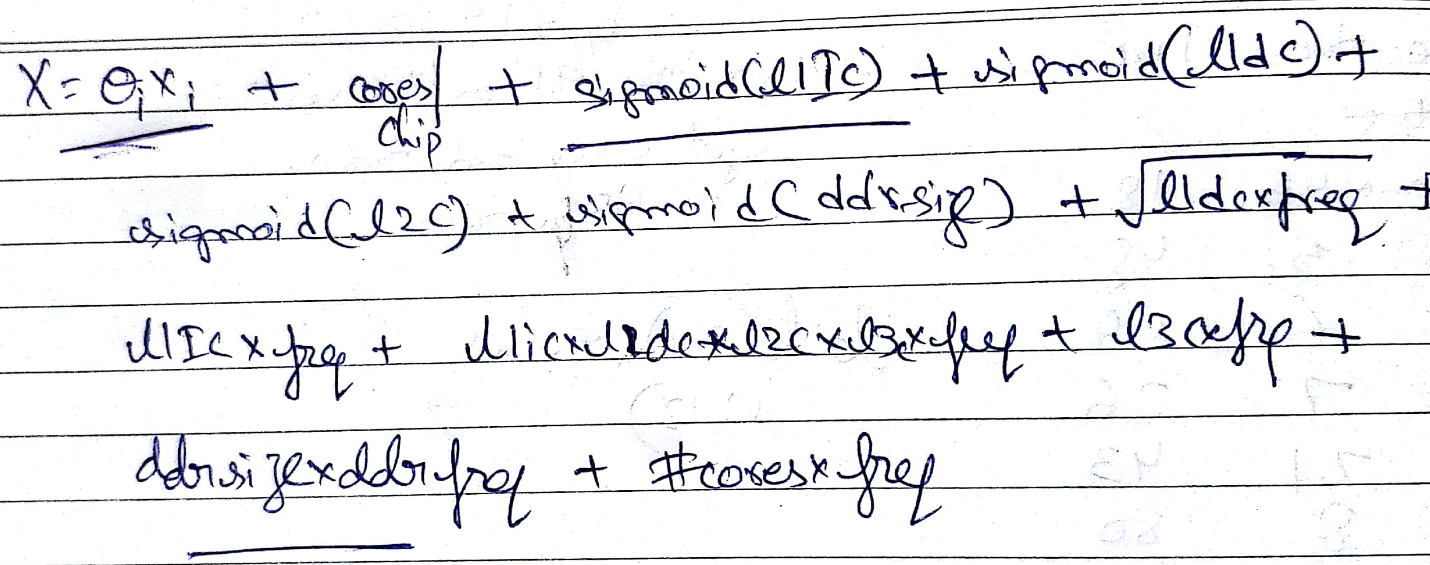
Now, we decided to plot each of the input features w.r.t the output spec number. The plots are drawn in the attached document.

There are 2 steps before us to ensure our model converges to an accuracy of close to 90%:

1. Based on the plots drawn, add more polynomial terms to the model and see which of these increases the accuracy of the model.
2. Analyze the mispredicted indices and see if we can infer something.

Both the above steps were performed iteratively till the model converged.

We were able to arrive at the below equation(equation is in data\_model.m in the attached code), which gave an accuracy of 77%.



When we further analyzed the data (and the mispredicted indices), there were several entries with all the input features being the same yet the output SpecFP number varied by a huge margin (20%-30%). When we analyzed the data further closely, these input features were differing in the compiler version. Some of them had Intel Fortran Compiler with various versions, which others had AMD compiler. Based on experience, we do know that a compiler can optimize the code to attain a higher SpecFP number, but this cannot be quantified. So, we had to split the entries into multiple sets of input data (one block per compiler version), and use the same equation as above to determine the accuracy.

The same equation as above, when applied to each input set (one input set per compiler version), achieves an accuracy of 87%.

This solves our problem.

################################# END ################################################

Reference Links:

1. Input SPEC data:

<http://spec.org/cgi-bin/osgresults?conf=cpu2000>

1. DDR types and specification:

<http://www.hardwaresecrets.com/everything-you-need-to-know-about-ddr-ddr2-and-ddr3-memories/>

<https://en.wikipedia.org/wiki/RDRAM>