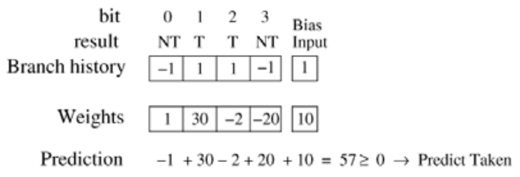
Proposal on designing a ML based branch predictor taking branch offset into account

Prior work (ML based branch predictor I found on IEEE):

1997: Calder et al – pre-defined static decision tree

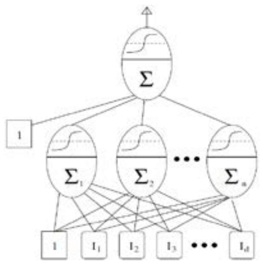
2001: Jimenez et al – weighted sum of past taken / not taken sequence. (least mean square), run-time training



bit	0	1	2	3	Bias
result	NT	T	T	NT	Input
Branch history	−1	1	1	−1	1
Weights	1	30	−2	−20	10

Prediction $-1 + 30 - 2 + 20 + 10 = 57 \geq 0 \rightarrow$ Predict Taken

2004: Smith et al - multi-layer of 1D data sequence, tested on different networks.



2005: Wang et al - vector – based “1.5D” training model.

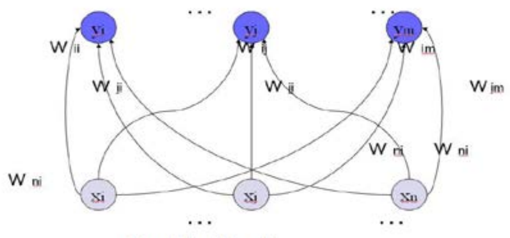
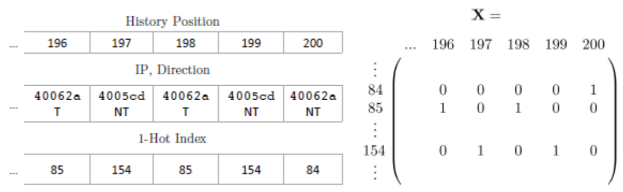


Figure-2 The LVQ model

2017 Mao et al - 6-layer 1D neural network

2019: Tarsa et al – convert past branch predictions into a one-hot matrix, and use 2-layer CNN, runtime.



	History Position				
...	196	197	198	199	200
	IP, Direction				
...	40062a T	4005cd NT	40062a T	4005cd NT	40062a NT
	1-Hot Index				
...	85	154	85	154	84

$$\mathbf{X} = \begin{array}{c|ccccc} & \dots & 196 & 197 & 198 & 199 & 200 \\ \hline \vdots & & & & & & \\ 84 & & 0 & 0 & 0 & 0 & 1 \\ 85 & & 1 & 0 & 1 & 0 & 0 \\ \vdots & & & & & & \\ 154 & & 0 & 1 & 0 & 1 & 0 \\ \vdots & & & & & & \end{array}$$

Fig. 2: A CNN is fed H2P-1’s global history as a matrix of 1-hot columns with 1’s in indices $((IP \ll 1) + Dir)\&(2^p - 1)$.

2020: Mao et al – use DNN, deeper layer, but 1D (best performed have deepest network, 18 layers), pre-compiled.

Mao showed that the DNN based predictor performs excellent in media – related workload (uniform repetition, many fixed-sized or parameterized loops)

What I propose:

All prior on machine-learning based branch predictors are based on branch history, which is a sequence of 1s and 0s. This is totally understandable that branch predictor's training should be computed in run time. And it's also been addressed by Mao et al that such branch predictor performs excellent in media-related workloads. However, in my opinion, except from the past branch taken / not taken sequence, as well as the program-pointer contains information for predictor training, I take the assumption that the "step" parameter, which is the branch offset should also be taken into account during training. I find this assumption rather intuitive that in media-related workload, the loops in the program often have fixed iteration and the branch offset are static rather than dynamic, and such iteration pattern may occur multiple time in a data-stream type workload.

So, based on the assumption that the branch offset contains valuable information to training a branch predictor and propose the following training model:

The input is a sparse, one-hot 3D matrix with three axes: PC/taken, step size, and time;

The PC/taken axis design is referred from Tarsa et al (2019), that the value of PC/taken =

LSBs(PC of Branch Instruction) << 1 + taken/not taken(1/0)

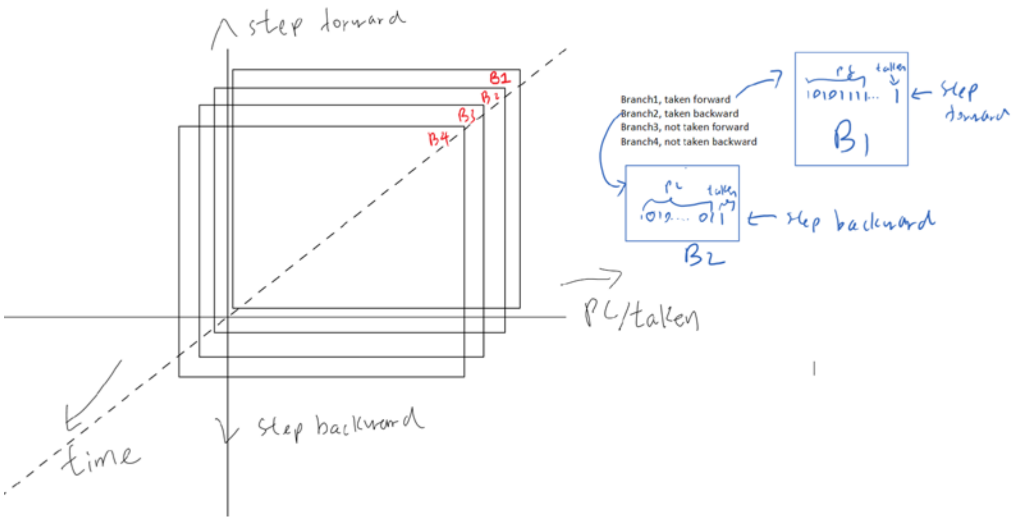
For example, Branch @ 0x0000_1101 is taken, then the value of this axis is 0001_1011

Branch @ 0x0000_0101 is not taken, then that value of this axis is 0000_1010

The step size axis is some_activation_function(branch imm) + offset,

The time axis is basically the branch pattern history

To visualize the 3D branch history information matrix, is shown:



As the data presented above, we can setup some pre-defined or come up with a 3D CNN model and evaluate whether including "step size" parameter into account would yield a nice result on certain workloads.