Perceptron Inspired Branch Prediction

By David Egolf, for CPTR 350. March 10, 2016.

Presentation Overview

• Motivation
• Transition: saturating counters
• Single perceptron predictor
  • Motivation
  • Derivation
  • Example
• Conceptual overview of refinements
Conditional branching instructions cause problems in pipelines.

- Mispredictions lead to...
  - Wasting work on wrong instructions
  - Stalling while grabbing new instructions

- We want a predictor that is...
  - Accurate
  - Fast (simple)
  - Low power
An attempt: Saturating Counter

- Use context specific counters
- Keep track of $n$ history bits
  - Add one counter per possible histories
  - Problem: $2^n$ counters required
A Big Problem

- We want to use history in predictions
- But, avoid exponential growth of hardware
- What to do?

Single Perceptron Predictor

inputs weights

\[
\begin{align*}
\text{new } w_i &= \text{old } w_i + t \cdot x_i \\
\end{align*}
\]
Example

- Predict user provided pattern
  - [1, -1, -1, 1, 1]

- Using 15 history inputs
  - There are 16 weights

- Error disappears quickly!

Problems with Perceptrons

- Calculating the weighted sum is slow

- Linear separability issues
Refinement: Path Based

- Decrease latency
- Increase precision

Figure 1. Rather than being done all at once (above), computation is staggered (below).

Refinement: Piecewise Linear

Figure 1. The XOR function cannot be learned by a perceptron (a), but can be learned using a piecewise linear decision surface (b).
Why accurate branch prediction?
- Avoid stalling
- Avoid doing wasted work

Why perceptrons?
- Simple, scale well, accurate

Figure 1. The GEHL predictor
Review of Presentation

- Problems with perceptrons?
  - Slow, linear separability issues

- Some ways to improve:
  - Path based prediction
  - Piecewise linear prediction
  - O-GEHL

Questions?