特別研究報告書

ニューラルネットを用いた
分岐予測器の改良

指導教員　　富田 眞治 教授

京都大学工学部情報学科

高橋 俊和

平成 17 年 2 月 10 日

ニューラルネットを用いた
分岐予測器の改良

高橋 俊和

内容梗概

近代的なプロセッサにとって，分岐予測 (branch prediction) は不可欠な技術となっている．近年のプロセッサではパイプラインが深化する傾向にあり，分岐予測ミス・ペナルティを低く抑えるためには，分岐予測ヒット率の更なる向上が必要となる．

現在までに提案されてきた多くの分岐予測器は g-share に代表されるような，飽和型カウンタとグローバル分岐履歴を組み合わせた方式であった．

しかし，分岐予測テーブルのインデクスの一部にグローバル分岐履歴を用いる方式では，同一の分岐に対して複数のエントリを更新するため，破壊的競合 (destractive conflict) の発生確率が上昇してしまう．この確率を一定に保つためには，グローバル分岐履歴長の指数に比例するメモリ量が必要となる．

それに対して Jiménez らは，パーセプトロンを用いた分岐予測器を提案している．Jiménez らの方式では，必要なメモリ量は，グローバル分岐履歴長に比例する．この方式は，パーセプトロンの重みセット (weight set) をいくつか用意し，複数の分岐命令が同一の重みセットを共有する．評価の結果では，重みセットの数を増やして，共有の度合いが下がると，性能が向上しており，共有は性能に悪影響を与えていることを示している．

本稿では，重みセットの共有による性能への悪影響を低減させる方法を提案する．一つは重みセットを一切共有させない方法，一つは分岐傾向の似た分岐命令同士で重みセットを共有するという方法である．

結果，SPEC CINT95 ベンチマークの 6 つのプログラムで，1 つめの方式ではメモリ量 1〜6KB の領域で Jiménez らの方式よりも高い予測ヒット率 (最大約 0.2%) を得ることができたが，2 つめの方式では，perl を用いた評価で Jiménez らの方式を上回る予測ヒット率を得ることはできなかった．

2 番目の方式で性能の向上を得られなかった理由として，この方式で用いたニューラル・ネットワークに対して提案した学習則が，分岐傾向の似た分岐命令同士で重みセットを共有するという目的を果たすような学習を行えなかったことが挙げられる．

Improvement of Neural Branch Predictors

Toshikazu TAKAHASHI

Abstract

A Branch prediction is indispensable technology for modern microprocessors. In recent years, the deeper pipeline gets, the higher branch miss prediction penalty reaches. Thus, in order to reduce this penalty, branch predictors must predict more acculately. Many branch predictors have been investigated but most of them use saturating counters and a global branch history (GBH). Gshare is a typical one.

But methods that use GBH to decide an index of pattern history table renew plural entries for a same branch. As a result, a probability of destractive conflict rises. To keep it constant, they need memory capacity that is proportional to an exponentiation of GBH length.

In contrast with them, Jiménezet al. proposed branch predictors with perceptrons. A memory they need is proportional to GBH length.

Their predictors have several weight sets of perceptrons and plural branches share the same one. An evaluation shows that an increase of weight sets improves their prediction accuracy. It means that sharing weight sets gives a bad influence.

In this paper, we propose 2 methods to reduce the bad influence for prediction accuracy.

- Each branch uses exclusive weight set to predict.
- Sharing each weight set by branches that have a same tendency.

Our exprerimental results using SPEC CINT95 show:

- Using the first method, prediction accuracy is, in the best case, increased 0.2% from 1 to 6KB memory.
- Using the second method, prediction accuracy doesn't exceed the one Jiménez's predictors gain.

On second method, we proposed learning rules of neural network in that predictor. But they could not let neural network work as we hope.

ニューラルネットを用いた
分岐予測器の改良

目次

第 1 章　はじめに　1

第 2 章　ニューラル・ネットワークの基礎　2

2.1　ニューロンの基礎 2

2.1.1　ニューロン 2

2.1.2　バイアス 3

2.2　パーセプトロン 4

2.2.1　パーセプトロンの学習 4

2.2.2　パーセプトロンの性能 5

2.2.3　デルタ則 5

2.3　階層型ネットワークと誤差逆伝播法 (BP) 7

2.3.1　誤差逆伝播法 7

2.3.2　対称性の破壊 9

第 3 章　既存の分岐予測器　9

3.1　グローバル分岐履歴と飽和型カウンタを組み合わせる方式 10

3.2　Filter 機構 10

3.3　Jiménez らの方式 11

3.3.1　Jiménez らの方式におけるパーセプトロン 11

3.3.2　Jiménez らの方式 12

3.3.3　性能 13

第 4 章　提案方式　13

4.1　分岐予測におけるニューラル・ネットワーク 14

4.1.1　高速性 14

4.1.2　入力 15

4.1.3　精度 16

4.1.4　学習速度 16

4.2　タグ付き重みセットを用いた方式 16

4.2.1　構成 16

4.2.2 改善策 17
4.3 重みセット選択ニューロンを用いた方式 18
4.3.1 構成 18
4.3.2 出力関数 18
4.3.3 出力計算と学習 20
第 5 章 評価 **21**
5.1 評価モデル 21
5.2 評価結果 22
5.2.1 タグ付き重みセットを用いた方式 22
5.2.2 重みセット選択ニューロンを用いた方式 23
第 6 章 おわりに **24**
謝辞 **24**
参考文献 **25**

第1章　はじめに

近代的なプロセッサにとって，**分岐予測** (branch prediction) は不可欠な技術となっている．分岐予測ミス時のペナルティは，命令パイプラインの命令フェッチ・ステージから実行ステージまでのサイクル数で与えられる．ところが，近年のプロセッサでは，パイプラインはますます深化する傾向にある．そのため，分岐予測ミス・ペナルティを低く抑えるためには，分岐予測ヒット率の更なる向上が必要となる．

現在までに様々な分岐予測器が提案されてきた．その多くは，飽和型カウンタを用いた方式であった．通常，2b のカウンタをエントリとする，**PHT** (pattern history table) と呼ばれる表が用いられる．最も単純な方式では，PHT のインデクスに分岐命令のアドレス（の一部）を用いる．

最も代表的な **g-share** 分岐予測器では，PHT のエントリを指し示すインデクスの一部に**グローバル分岐履歴** (global branch history) を加えることで，ヒット率の向上を狙っている．

しかし，インデクスの一部にグローバル分岐履歴を用いる方式では，同一の分岐に対しても，出現するグローバル分岐履歴のパタンの数だけエントリを更新することになる．その結果，**破壊的競合** (destractive conflict) の発生確率が上昇してしまう．破壊的競合の発生確率を一定に保つためには，グローバル分岐履歴長の指数に比例するメモリ量が必要となる．

Jiménez らはそれに対して， パーセプトロンを用いた分岐予測器を提案している [1, 2]．パーセプトロンを用いた分岐予測器では，必要なメモリ量は，グローバル分岐履歴長に比例する．結果，同メモリ量で比較的長い履歴を用いることができ，既存の方式よりも高い予測ヒット率を示している．

Jiménez らの方式では，パーセプトロンの**重みセット** (weight set) をいくつか用意し，複数の分岐命令が同一の重みセットを共有する．評価の結果では，重みセットの数を増やして，共有の度合いが下がると，性能が向上しており，共有は性能に悪影響を与えていることを示している．

そこで本稿では，重みセットの共有による性能への悪影響を低減させる方法を提案する．一つは重みセットを一切共有させない方法，一つは分岐傾向の似た分岐命令同士で重みセットを共有するという方法である．

以下，第 2 章ではニューラル・ネットワークの基礎について，第 3 章では既

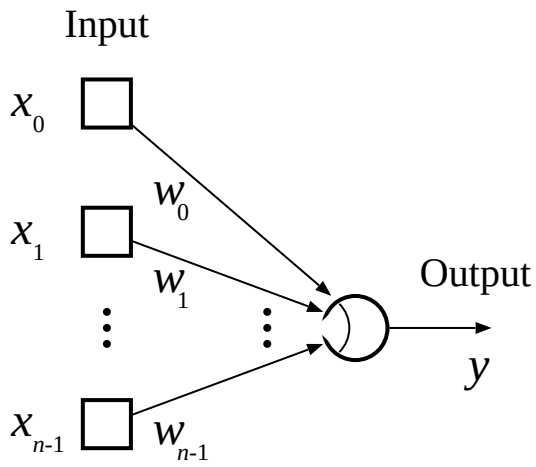

図 1: ニューロン

存の分岐予測器と Jiménez らの方式について述べ，第 4 章で Jiménez らの方式の性能を向上させる案について述べ，第 5 章では第 4 章で実装した分岐予測器の性能を評価する．

第 2 章　ニューラル・ネットワークの基礎

次章以降での議論に集中するため，本章では，本稿で取り上げる分岐予測器で用いられるニューラル・ネットワークについてまとめておく．

以下，2.1 節では，基本的な構成要素であるニューロンについて，2.2 節では，パーセプトロンについて，2.3 節では，提案方式で用いられる階層型ニューラル・ネットワークと誤差逆伝播法について説明する．

2.1　ニューロンの基礎

2.1.1　ニューロン

図 1 に，**ニューロン**(neuron) のモデル図を示す．図中，ニューロンは ◯ で表されている．

1 つのニューロンは，通常複数の**入力** (input) を持つ．同図では $n-1$ 個の入力が ロで表されている．入力枝には，それぞれ**重み**(weight) が付けられている．重みとは，その枝に入力が与えられたときにニューロンの**膜電位** (membrane potential) がどれほど上昇するかを表す．この膜電位がある**閾値**を超えるとニューロンが興奮すると考えられる．

i 番目の入力を x_i としよう．入力の値 x_0〜x_{n-1} は，「ある」か「ない」かであるか，それぞれ，0 または 1 である．したがって，i 番目の入力枝の重みを w_i とすると，膜電位 u は，重みベクトル $\boldsymbol{w} = (w_0, w_1, \cdots, w_{n-1})$ と，入力ベクトル $\boldsymbol{x} = (x_0, x_1, \cdots, x_{n-1})$ との内積として得られる．すなわち u は，次式で表される：

$$u = \boldsymbol{w} \cdot \boldsymbol{x} = \sum_{i=0}^{n-1} w_i x_i \,. \tag{1}$$

ここで，以下のような**単位ステップ関数** (unit step function) を考える：

$$f(u) = \begin{cases} 1 & (u \geq 0) \\ 0 & (u < 0) \,. \end{cases} \tag{2}$$

単位ステップ関数を用いれば，ニューロンの出力 y は次式のようにモデル化される：

$$y = f(u - \theta) = f\left(\sum_{i=0}^{n-1} w_i x_i - \theta \right) . \tag{3}$$

式 (3)，および，(2) から，ニューロンの出力 y は，膜電位 u が閾値 θ 以上なら 1，より小さければ 0 となる．

関数 f には，上述の単位ステップ関数以外も用いることができる．そこでこの f のことを一般に，**出力関数** (output function) と呼ぶ．

2.1.2 バイアス

式 (3) は，閾値 θ の分だけやや煩雑である．そこで，以下のような変形を行うと，閾値 θ を重みの 1 つとして統一的に扱うことができる．すなわち，仮想的に常に 1 であるような n 番目の入力 $x_n = 1$ を加え，かつ，その重み w_n を $w_n = -\theta$ とするのである．そうすると，式 (3) は，以下のように簡単化される：

$$u - \theta = \sum_{i=0}^{n-1} w_i x_i + w_n = \sum_{i=0}^{n-1} w_i x_i + w_n x_n = \sum_{i=0}^{n} w_i x_i \,.$$

$$y = f(u - \theta) = f\left(\sum_{i=0}^{n} w_i x_i \right) = f(\boldsymbol{w} \cdot \boldsymbol{x}) \,. \tag{4}$$

なお上式では，$(w_0, \cdots, w_n)$，および，$(x_0, \cdots, x_n)$ を，それぞれ，$\boldsymbol{w}$，$\boldsymbol{x}$ と置き直している．この表現式において，閾値を表す重み w_n を，特に**バイアス** (bias) という．

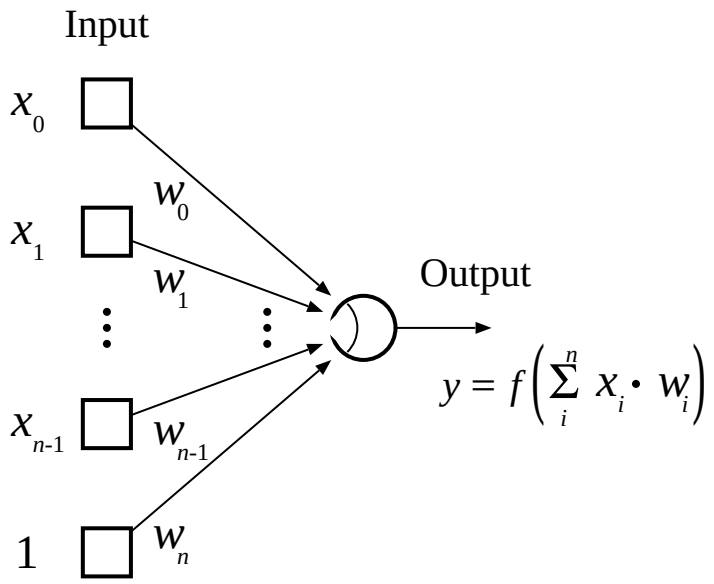

図 2: ニューロン（バイアスあり）

2.2 パーセプトロン

ニューラル・ネットワークの内，以下の条件を満たすものを，特に**パーセプトロン** (perceptron) という：

- 出力関数が単位ステップ関数（式 (2)）である．
- フィードバック・ループを持たない．

パーセプトロンは，ニューラル・ネットワークの内で最も基本的なものの 1 つである．

ニューロンが 1 つのみの，つまり前節の図 2 で表されるようなパーセプトロンについて考える．これが最も単純なパーセプトロンであり，後述する Jiménez らの方式のパーセプトロンと類似の構成である．

2.2.1 パーセプトロンの学習

パーセプトロンの学習は普通，**教師付き学習**(supervised learning) によって行われる．教師付き学習とは，ある入力に対して望まれる出力が**教師信号**(teacher signal) として与えられるものである．

教師付き学習では，基本的には，出力が教師信号の値に近づくように重みを変化させればよい．例えば，1 と出力すべきところを 0 と出力してしまった場合には，1 である入力の枝の重みを増加させればよい．すると次回同一の入力が与えられた場合には，増加させた分だけ膜電位 u の値は大きくなる．このよう

な過程を繰り返せば，やがてニューロンは正しい値を出力するようになる．逆に，0 と出力すべきところを 1 と出力してしまった場合には，1 である入力の枝の重みを減少させればよい．

具体的には，各 w_i に対して加えられる修正量 Δw_i は，

$$\Delta w_i = \eta(t - y)x_i \tag{5}$$

で表される．ここで η は**学習率**(learning rate) と呼ばれる定数であり，通常正の小さな値（1 以下）とする．

後述するように，問題が線形分離可能であれば学習は有限回で収束し，正しい出力を行えるようになる．

2.2.2 パーセプトロンの性能

線形分離　入力ベクトル $\boldsymbol{x_p}$ に対し，1 を出力すべきものの集合を X^+，0 を出力すべきものの集合を X^- とする．このとき，

$$\begin{cases} x_p \in X^+ & (\boldsymbol{w} \cdot \boldsymbol{x_p} > 0) \\ x_p \in X^- & (\boldsymbol{w} \cdot \boldsymbol{x_p} < 0) \end{cases} \tag{6}$$

となるような一次式

$$\boldsymbol{w} \cdot \boldsymbol{x} = w_0 \cdot x_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n \tag{7}$$

が存在するとき，X^+ と X^- は**線形分離可能**(linearly separable) であるという．つまり全ての入力ベクトルに対して，出力の集合が線形分離可能であれば，正しい出力を行えるパーセプトロンの重みが存在するということである．

パーセプトロンの収束定理　パーセプトロンの収束定理により，線形分離可能なら学習が有限回で完了する．

パーセプトロンの限界　線形分離可能でないものは学習できない．線形分離可能でない問題の典型的なものとしては，排他的論理和がある．

後に述べる，誤差逆伝播法を用いた階層型ニューラル・ネットワークは，この問題を解決している．

2.2.3 デルタ則

線形分離可能でない問題においても，教師信号と出力との 2 乗誤差

$$E = \frac{1}{2}(t - y)^2 \tag{8}$$

が最小になるように重みが収束することが望まれる．しかしパーセプトロンにはそのような性質はなく，その欠点の克服のために，2 値を出力するニューロンの代わりに連続値を出力するニューロンを採用するという試みがなされ，**デルタ則**(delta rule) と呼ばれる学習則が提案されてきた．

このモデルでは入力 x_i，出力 y，教師信号 t が $(0, 1)$ の連続値をとる．さらに出力関数は線形 (すなわち $f(u) = u$) であり，$y = \boldsymbol{w} \cdot \boldsymbol{x}$ で与えられる．入力パターン p と教師信号 t_p が与えられた場合のデルタ則は，

$$\begin{cases} \Delta_p w_i = \eta \delta x_{pi} \\ \delta = t_p - y_p \end{cases} \tag{9}$$

で表され，学習は，全ての入力に対する E が設定値以下になると終了する．

デルタ則が教師信号と出力との 2 乗誤差を最小にするような学習であることを以下に示す．

入力が P 組あったとして，その p 番目の入力パターン $\boldsymbol{x_p}$ に対する出力 y_p と教師信号 t_p との 2 乗誤差

$$E_p = \frac{1}{2}(t_p - y_p)^2 \tag{10}$$

を定義すると，全てのパターンに対する 2 乗誤差の総和は

$$E = \sum_{p=1}^{P} E_p \tag{11}$$

となる．この場合のデルタ則は

$$\Delta_p w_i = \eta \delta x_{pi} \tag{12}$$

$$\delta = t - y \tag{13}$$

で表される．各 E_p に対して

$$-\frac{\partial E_p}{\partial w_i} = \delta x_{pi} \tag{14}$$

となることを示す．合成微分の公式により，この式の左辺は

$$\frac{\partial E_p}{\partial w_i} = \frac{\partial E_p}{\partial y_p} \frac{\partial y_p}{\partial w_i} \tag{15}$$

となり，さらにこの右辺を計算すると式 (10) より

$$\frac{\partial E_p}{\partial y_p} = -(t_p - y_p) = -\delta \tag{16}$$

そして $y = \boldsymbol{w} \cdot \boldsymbol{x}$ なので

$$\frac{\partial y_p}{\partial w_i} = x_{pi} \tag{17}$$

以上より

$$\frac{\partial E_p}{\partial w_i} = -\delta x_{pi} \tag{18}$$

ここで

$$\frac{\partial E}{\partial w_i} = \sum_p \frac{\partial E_p}{\partial w_i} \tag{19}$$

なので，全ての入力パターンを一度ずつ入力した後の重みの変化の総和は $\frac{\partial E}{\partial w_i}$ に比例することが分かる．これが厳密に成立するのは，全ての入力パターンを用いた後で一括して重みの修正を行う場合であるが (はじめの方で現れた入力パターンによる修正が後の方で現れる入力パターンに対する修正に影響を与えるため)，学習率 η を十分小さくすることでそのずれを無視することができる．

標準デルタ則は 2 乗誤差に対する最急降下法のよい近似となっているといえる．

2.3 階層型ネットワークと誤差逆伝播法 (BP)

今までは単層のパーセプトロンについて説明してきた．以下では多層から構成される階層型ネットワークとその学習法について述べる．

2.3.1 誤差逆伝播法

誤差逆伝播法(back-propagation : BP) は，フィードフォワードの階層型ニューラル・ネットワークに対する教師付き学習法の一つで，出力層から入力層に向かって重みを修正するものである．標準デルタ則を階層型ネットワークに対して一般化した学習則とみなすことができるため，**一般化デルタ則**(generalized delta rule) とも呼ばれる．

この学習法は，出力層から入力層に向かって，出力層以外の上位層の重みの修正量が出力から求まるというものである．

最急降下法に従って，各ニューロン $j(j = 1 \cdots m)$ に対し $w_{ji}(i = 0 \cdots n)$ を $\frac{\partial E_p}{\partial w_{ji}}$ に比例して修正するのであるが，ニューラル・ネットワークが多層となると E の微分を求めるのは容易ではなくなる．そこで，**シグモイド関数**(sigmoid function) といったような関数が，BP においては各ニューロンの出力関数としてしばしば使われる．シグモイド関数は以下のように表される:

$$f(u) = \frac{1}{1 + e^{-\beta u}} \tag{20}$$

この関数は，$-\infty < u < +\infty$ で $(0, 1)$ の範囲の連続値をとり，微分可能な単調増加の連続関数となる．β は関数の変化する急激さを定める定数で，$\beta \to \infty$ の極限で式 (2) の単位ステップ関数となる．

以下で一般化デルタ則の定式化を行う．

ニューロン j に対して，入力パターン p とその出力 y_{pj}，教師信号 t_{pj}，出力と教師信号との 2 乗誤差 $E_p = \frac{1}{2}\sum_j (t_{pj} - y_{pj})^2$ を定義する．この時，2 乗誤差の総和 $E = \sum_p E_p$ に対して，

$$\Delta_p w_{ji} \propto -\frac{\partial E_p}{\partial w_{ji}} \tag{21}$$

が成立するような重みの修正法について考察する．右辺を合成関数の微分公式を用いて変形すると

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial u_{pj}} \frac{\partial u_{pj}}{\partial w_{ji}} \tag{22}$$

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} y_{pk} \tag{23}$$

という関係式が得られる．ここで誤差の変化を $\delta_{pj} = -\frac{\partial E_p}{\partial u_p}$ とおいた．以上から，

$$\Delta_p w_{ji} = \eta \delta_{pj} y_{pk} \tag{24}$$

と重みを修正すればよい．各ニューロンに対する δ_{pj} は，出力層から遡って計算することで再帰的に得ることができる．

$$-\frac{\partial E_p}{\partial u_{pj}} = -\frac{\partial E_p}{\partial y_{pj}} \frac{\partial y_{pj}}{\partial u_{pj}} = -\frac{\partial E_p}{\partial y_{pj}} f'(u_{pj}) \tag{25}$$

更に最右辺第一項は，出力層では標準デルタ則同様 $-(t_{pj} - y_{pj})$ となるので，出力層における δ_{pj} は

$$\delta_{pj} = -(t_{pj} - y_{pj}) f'(u_{pj}) \tag{26}$$

他の層では更に合成関数の微分の公式から

$$\frac{\partial E_p}{\partial y_{pj}} = \sum_k \frac{\partial E_p}{\partial u_{pk}} \frac{\partial u_{pk}}{\partial y_{pj}} = \sum_k \frac{\partial E_p}{\partial u_{pk}} \frac{\partial}{\partial y_{pj}} \sum_i w_{ki} y_{pi}$$
$$= \sum_k \frac{\partial E_p}{u_{pk}} w_{kj} = -\sum_k \delta_{pk} w_{kj} \tag{27}$$

となるので，他の層における δ_{pj} は

$$\delta_{pj} = f'(u_{pj}) \sum_k \delta_{pk} w_{kj} \tag{28}$$

で得られる．ここで $\sum_k$ は，ニューロン j の出力を入力としている次層のニューロンの全てに対する総和を表す.

一般化デルタ則での重みの修正量 $\Delta_p w_{ji}$ は，その結合の終点となる出力層のニューロンに関する誤差信号 δ_{pj} と，結合の始点となる入力層のニューロンの出力 y_{pi} の積に比例したもの，すなわち式 (9) で表された標準デルタ則の式と同じ形になっている.

2.3.2 対称性の破壊

多層からなるニューラル・ネットワークでは，もし全ての重みの初期値が同じであると、全ての重みの修正量が同じになってしまい非対称な解を得ることはできなくなる。重みの初期値を小さな乱数に設定しておくことでこの問題は解決される。

第3章 既存の分岐予測器

現在，分岐予測の方式としては，グローバル分岐履歴と飽和型カウンタを組み合わせた方式が一般的になっている．例えば，最も有名な g-share 分岐予測器では，PHT のインデクスにグローバル分岐履歴を用いることによって，グローバル分岐履歴を考慮したきめ細かい予測が行われる．しかし，グローバル分岐履歴のどの部分も同じ重要度で用いているため，本来の予測には無関係な履歴が混ざると学習が分散してしまう.

1 章でも述べたように，Jiménez らは，g-share などのカウンタを用いる方式に対して，パーセプトロンを用いた分岐予測の方式を提案している．パーセプトロンを用いると，グローバル分岐履歴のどの部分が重要か，重要でないかを認識できるため，それにより予測の精度を向上させることができる.

以下，3.1 節ではまず，グローバル分岐履歴を用いた方式について述べ，3.2 節では予測精度を上げる仕組みの 1 つ Filter 機構，3.3 節で Jiménez らの方式について述べる．

3.1 グローバル分岐履歴と飽和型カウンタを組み合わせる方式

グローバル分岐履歴を用いた方式の一つ，g-share について説明する．g-share は，グローバル分岐履歴による分岐の傾向，分岐命令アドレスによる分岐の傾向，その両方を考慮した予測を行う分岐予測器である．

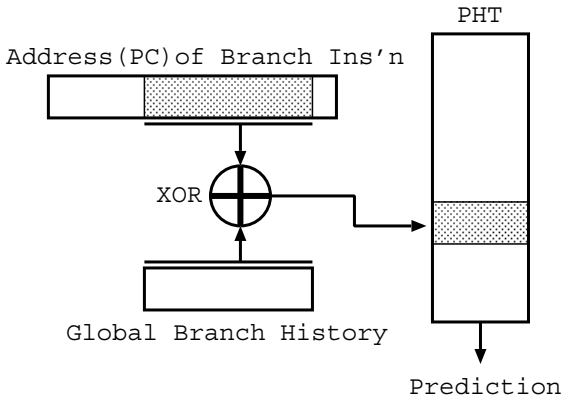

図 3: g-share 予測器

g-share の PHT は多数の 2 ビット飽和型カウンタで構成されている．分岐命令アドレスの下位数ビットと グローバル分岐履歴の 排他的論理和で PHT の インデクスを決定する．カウンタの値が 2 以上ならば taken、1 以下ならば not taken と予測する．使用されたカウンタは，分岐結果が taken ならば値が+1，not taken ならば-1 される．

3.2 Filter 機構

分岐予測器は，無数の分岐命令を有限の PHT を用いて予測するため，PHT の各エントリが異なる分岐命令によって共有されることとなる．このエントリを共有する分岐命令同士の分岐の傾向が同じならば効率よく予測ができるが，大きく異なっていると学習がいつまでたっても進まない．

Filter 機構 [3] は，この破壊的競合による予測精度の低下を防ぐため，taken, not taken のどちらかに強く偏向している分岐命令を分岐予測器による予測の対

象から外す手法である．

この手法では，BTB の各エントリにバイアスビットと確信度カウンタが追加される．分岐結果とバイアスビットが同じ場合にはカウンタの値が+1 され、異なる場合にはバイアスビットを反転してカウンタを 0 に戻す。確信度カウンタが全ビット 1 になると、当該分岐命令を偏った分岐命令と判断して、バイアスビットの値を予測値として利用する。この場合、分岐予測器の更新は行わない。

3.3 Jiménez らの方式

Jiménez らの方式における パーセプトロンは，一般的なものとは多少異なる．以下，3.3.1 節でそのパーセプトロンの差異について，3.3.2 節では Jiménez らの方式の構成について，3.3.3 節ではその性能について述べる．

3.3.1 Jiménez らの方式におけるパーセプトロン

Jiménez らの方式におけるパーセプトロンは，2.2 節でとりあげたものと同じく，ニューロンが一つの単純なパーセプトロンである．

入出力　入力は 1/-1 で与えられる．出力もまた，1/-1 の 2 値である．

重み　8 ビット符号付き整数型で表される．

出力関数　入力と重みの積和

$$u = \sum_{i=0}^{n} x_i w_i$$

に対し，

$$f(u) = \begin{cases} 1 & (u \geq 0) \\ -1 & (u < 0) \end{cases} \tag{29}$$

となるステップ関数を用いて出力 $y = f(u)$ とする．

教師信号と学習　教師信号も 1/-1 で与えられる．

学習は，パーセプトロンの判断が誤った場合だけでなく，正しい判断を行っても積和の絶対値 $|u|$ がある閾値 θ 以下であった場合にも行う．

各 $i(i = 0 \ldots n)$ に対し，$x_i = t$ なら w_i を $+1$，$x_i \neq t$ なら w_i を -1 する．

これにより，判断が正しければより強くそう判断する (同じ入力ベクトルを入れると積和の絶対値がより大きくなる) ように，誤れば逆の判断をする (同じ入力ベクトルに対して積和の符号が逆になる) ように学習が行われることとなる．

原則は同じであるが，正しい判断が行われた場合にも学習を行う，学習時には全ての重みが変化する (一般のパーセプトロンでは，入力が 0 である重みは変

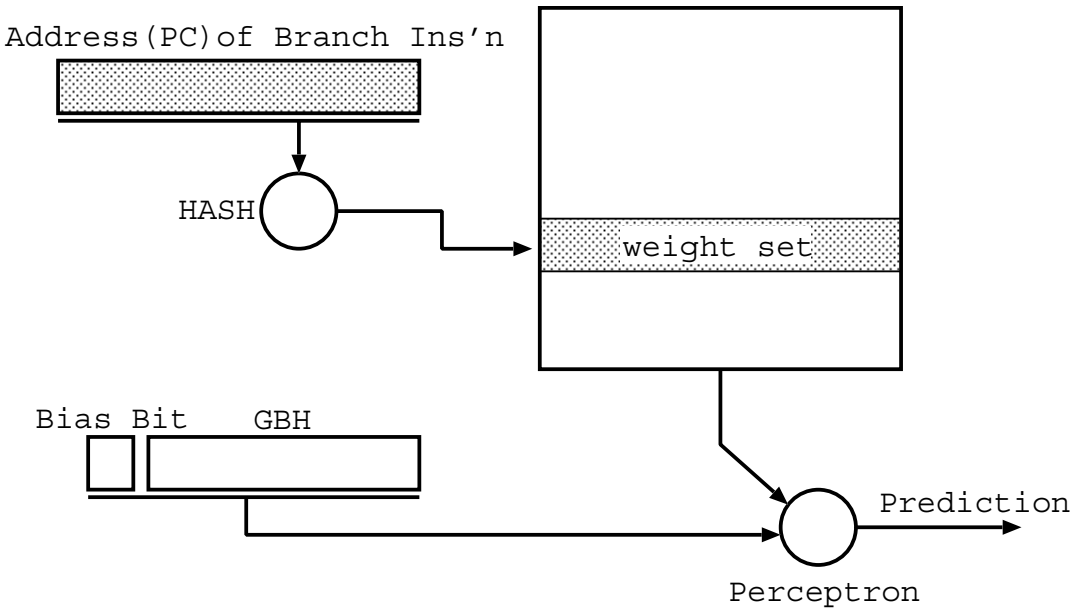

図 4: Jiménez らの方式

化しない) という部分が異なる.

この学習則により、結果に強く相関がある入力は対応する重みの絶対値が増大し、相関が小さい (あるいは無相関の) 入力は、対応する重みがゼロに近い値となっていく。重みは相関の強さを表すといえる.

3.3.2 Jiménez らの方式

図 4 に，Jiménez らによる分岐予測器の構成を示す．中心はパーセプトロンである．パーセプトロンにグローバル分岐履歴を入力して分岐予測を行う．ただし，複数の**重みセット** (weight set) を用いる．重みセットは，分岐命令アドレスをハッシングしたものをインデクスとするテーブルに格納されている．予測に際しては，前節で述べた通常の分岐予測器などと同様，分岐命令のアドレスを用いてテーブルにアクセスする．ただし，g-share などとは異なり，インデクスの生成にはグローバル分岐履歴を用いない．分岐命令に対して 1 つの重みセットが選ばれることになる．この重みセットをパーセプトロンの重みとし，グローバル分岐履歴を入力として，予測を行うのである．重みセットの数は，数十から数百程度を想定している.

入力　入力となるグローバル分岐履歴の各ビットは，taken に対して 1，not taken に対しては –1 を与える.

出力と予測　$y = 1$ で taken，$y = -1$ で not taken と予測する.

結果 (教師信号)　taken では 1，not taken では-1 が与えられる．

なお，学習の閾値 θ は Jiménez らによると，⌊1.93×(入力数)+14⌋ にした場合に最も高い予測精度をあげるとされている．

3.3.3　性能

同メモリ量 (4KB) の g-share 方式に比べて，SPEC2000int ベンチマークにおいて予測ヒット率が平均 1.6%向上するという結果が出ている．

また，Jiménez らは，入力に グローバル分岐履歴だけでなくローカル分岐履歴も用いれば簡単にローカル/ グローバル分岐履歴併用の分岐方向予測器にできるとしている。Alpha21264 と同程度のメモリ量における比較で，ローカル/ グローバル分岐履歴を入力とし，適切な閾値や履歴長に設定した場合，Alpha21264 の予測器と比べて予測ヒット率が平均 0.7%向上するという結果が出ている．

なお パーセプトロンのよく知られた性質として，前述したように線形分離でない問題を学習できないというものがあるが，同じハードウェア量で扱える履歴の長さが大きいため，線形分離でない分岐もより正確に予測することができ，十分カバーできているとしている．実際に，扱える履歴長が短くなってしまう低ハードウェア量の場合には g-share より予測ミス率が大きくなるケースもある．

第 4 章　提案方式

第 3 章で述べたとおり，Jiménez らの方式は，既存の分岐予測器を上回る予測精度をあげている．しかし，分岐命令アドレスをハッシングして用いる重みを決めるため，全く異なる分岐傾向を持つ分岐命令同士で同じ重みを共有する可能性があり，そこに改良を施すことでさらに予測精度を上げることができると考えられる．

以下，4.1 節では改良を施す上で，分岐予測におけるニューラル・ネットワークについて考慮しなければならない点について考察し，以降その改良案について，4.2 節では異なる分岐命令間で重みを一切共有させない方式について述べ，4.3 節では分岐命令がその分岐傾向に応じてどのニューロンを使うか学習していく方式について述べる．

4.1　分岐予測におけるニューラル・ネットワーク

分岐予測器などに用いるためのニューラル・ネットワークは，通常のニューラル・ネットワークと比べて，以下のような違いがある：

高速性　できれば 1 サイクル，そうでなくても 2 サイクル程度以内に予測を行う必要がある．ニューラル・ネットワークで普通用いられているような浮動小数点数を用いることは極めて困難である．整数（もしくは，固定少数点数）を用いる必要がある．多層化も計算量の増大を招くため，その意味で困難になる．

リアル・タイム性　第 2 章で見てきたようなニューラル・ネットワークは，全ての入力パターンによる学習を何度でも行い，”最終的に”正しい判断を行うようになればよいという目標を持つものであった．

それに対し，分岐予測器などに用いる場合，学習後に使用するのではなく，使用と学習を同時に並行して行ういわゆる**リアル・タイム学習**(real-time learning) となる．

プログラムの実行にしたがって，トレーニング・セットは次々変化していく．マルチ・プログラミング環境においては，別のプログラムに実行が移ることさえ考慮する必要がある．そのため，最小の実行回数で局所解，できれば最適解まで学習が行われなければならない．

以下で，より詳しく述べる．

4.1.1　高速性

1〜2 サイクル程度以内に結果を出す，という条件下では，様々な制限がなされる．

浮動小数点数加減算では，小数点の位置を一致させるためのシフト操作，結果の正規化のための最初の 1 位置の発見 (detect) とシフトが必要なため，これを用いることはできない．固定小数点数と考えても同じことである．

乗除算，平方根，指数関数などの超越関数，シグモイド関数も使えない．

ただし，もちろん，$\times 1$，$\times -1$ は可で，また，$\times 2^n$ (n は整数) もシフト演算で実現できる．

Jiménez らの方式では，予測時には，それぞれ 8b の重み 4〜50 個程度を加算，結果の符号によって taken/not taken を予測する．carry-save adder を用いてトゥリーを組むことにより，2 サイクル以内に実行可能としている．

4.1.2 入力

通常入力は 0/1 であるが，Jiménez らの方式では –1/1 としている．

–1/1 である入力を x_i' とすると，

$$\begin{aligned} x_i' &= 2x_i - 1 \\ x_i &= \frac{x_i' + 1}{2} \end{aligned} \tag{30}$$

である．これを式 (1) に代入して，

$$\begin{aligned} u &= \sum_{i=0}^{n} w_i x_i \\ &= \sum_{i=0}^{n} w_i \frac{x_i' + 1}{2} \\ &= \sum_{i=0}^{n} \left(\frac{w_i}{2} x_i' + \frac{w_i}{2} \right) \end{aligned}$$

ここで

$$w_i' = \frac{w_i}{2} \quad (i = 0, 1, \cdots, n-1) \tag{31}$$

とすると，

$$\begin{aligned} u &= \sum_{i=0}^{n-1} (w_i' x_i' + w_i') + w_n \\ &= \sum_{i=0}^{n-1} w_i' x_i' + \sum_{i=0}^{n-1} w_i' + w_n \end{aligned}$$

さらに，

$$w_n' = \sum_{i=0}^{n-1} w_i' + w_n \tag{32}$$

とすると，式 (4) と同様に，

$$\begin{aligned} u &= \sum_{i=0}^{n} w_i' x_i' = \boldsymbol{w'} \cdot \boldsymbol{x'} \\ y &= f(\boldsymbol{w'} \cdot \boldsymbol{x'}) \end{aligned} \tag{33}$$

となる．

したがって，式 (31) と式 (32) のような $\boldsymbol{w'}$ を選べば，式 (4) と式 (33) は等価となる．

4.1.3 精度

出力は積和の符号によって決まるので，出力を求める際に積和の正負さえ分かれば，予測時に積和の正確な値自体は不要である．

4.1.4 学習速度

分岐予測におけるニューラル・ネットワークは，より少ない学習で正しい判断を行うようにならなければならない．一般のニューラル・ネットワークは出力と教師信号の誤差によって学習が行われるが，学習を速めるため Jiménez らの方式では，正しい判断を行った場合にもさらに強くそう判断するよう，強化の学習が行われる．

4.2 タグ付き重みセットを用いた方式

個々の分岐命令に対し，専用の重みセットを用意する方式である．複数の分岐命令の学習が混ざることで予測精度が下がるのを防ぐ．しかし保存できる重みセットの数が有限であるため，新たな分岐命令のため重みセットを設ける際には，別の分岐命令の重みセットを破棄し置き換えるという作業が行われる．そのため，エントリの入れ替えが激しく起こる場合には競合よりも学習の破棄が予測精度に大きく影響を及ぼす可能性がある．

以下 4.2.1 節ではその構成を述べ，4.2.2 節 でそのデメリットを緩和する手法について述べる．

4.2.1 構成

Jiménez らの方式をベースとする．重みテーブルはセットアソシアティブ方式とし，各エントリに，分岐命令アドレスの上位ビットを記録する領域を設ける(以下これをタグと呼ぶ)．図 5 に重みテーブルが 4way の場合の構成を示す．

重みテーブルの インデクスを 分岐命令アドレスの下位数ビットによって決定し，対応したセットの中から， 分岐命令アドレスの上位ビットとタグが一致するものがあればその重みセットをパーセプトロンの重みとし，グローバル分岐履歴を入力として予測を行い，無かった場合は無条件で not taken と予測する．

重みのエントリが無かった場合に予測を誤ると，その分岐命令に対応した重みのエントリと LRU(Least Recently Used) のエントリとを置き換え初期化 (重みをゼロに) し，not taken と予測して誤ったものとして学習する．

学習則，学習の閾値 θ は Jiménez らの方式と同様である．

異なる分岐命令が同じ重みのエントリを選んだとしても，その 分岐命令アド

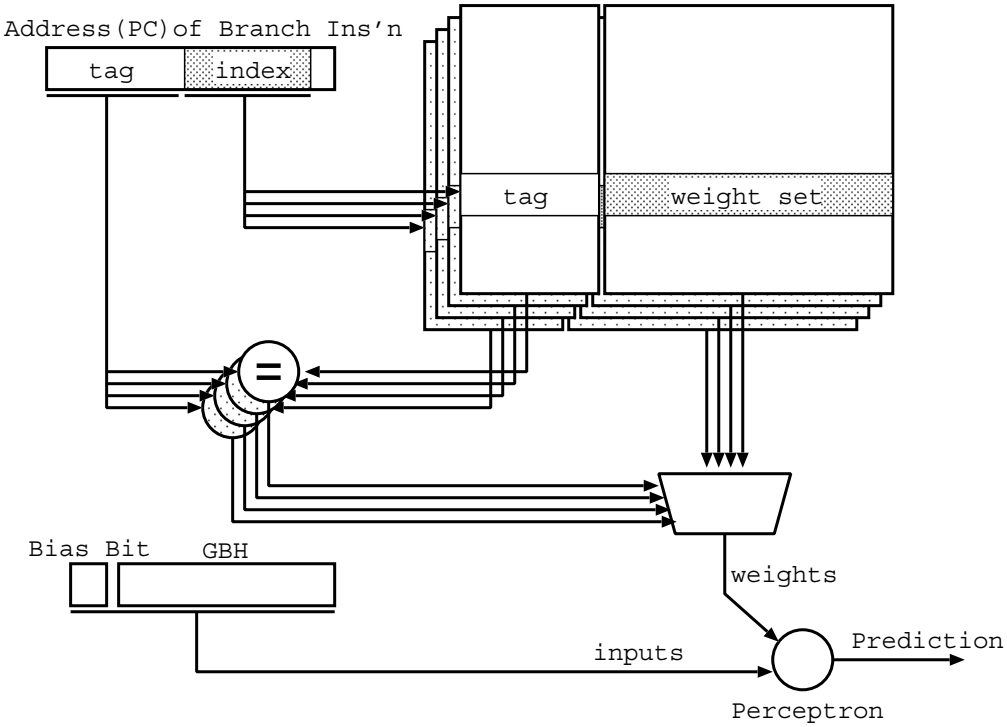

図 5: タグ付き重みセットを用いた方式 (重みテーブル 4way)

レスの上位ビットとタグは一致しないため，用いることはできない．これによって異なる分岐命令同士の学習が混ざるのを防ぐ．

重みセットの置き換えが起きた場合には，学習の破棄が起こる．その回数が大きいものとなると，多くの分岐命令に対して パーセプトロンの学習が進まなくなってしまう．

4.2.2 改善策

Filter 機構 タグ付き重みセットを用いた方式に Filter 機構を利用すると，強偏向の分岐命令が重みのエントリを持つ期間が減り，それによってエントリの置き換えの回数が緩和されると考えられる．他の分岐予測器でも同じように，競合を緩和できると考えられるが，この方式では更に，重みのエントリが無かった場合，BTB にエントリがあればバイアスビットによって予測をさせるという利用法を取り入れる．エントリが無い場合に常に not taken と予測するのに比べ，合理的な予測が行われると考えられる．

重みテーブルのキャッシュ化 重みエントリの置き換え時に，取り除かれた重みを破棄するのではなく，ある領域に退避させる．そして待避先に，新しく重みテーブルに置かれる分岐命令の過去の重みが保存してあれば，それを持って

くる．予測を行ってから分岐予測器の更新を行うまでの間に対応する重みセットを探せばよいので，この作業自体は分岐予測の速度に影響を及ぼさない．ゼロから学習しなおすのではなくなるため，これによって学習の破棄による予測精度への影響が緩和されると考えられる．

4.3 重みセット選択ニューロンを用いた方式

タグ付き重みセットを用いた方式は，重みエントリにタグをつけることで学習の混合を防ぎ，個々の分岐命令の予測を専用の パーセプトロンで行おうというものであった．

重みセット選択ニューロンを用いた方式は，分岐の傾向が似た分岐命令同士で重みセットを共有させることで，無作為に重みセットを共有していた Jiménez らの方式の予測ヒット率を向上させようというものである．

以下 4.3.1 節でその構成を，4.3.2 節で利用する パーセプトロンの詳細について述べる．

4.3.1 構成

2 層構造のニューラルネットワークである．グローバル分岐履歴を入力とする n 個の パーセプトロン (以下 GBH Perc.)，分岐命令アドレスを入力とする n 個の パーセプトロン (以下 ADDR Perc.)，さらにそれらの出力同士の積和をとる パーセプトロン (以下 SEC Perc.) から構成される．

Jiménez らの方式の方式では分岐命令アドレスでどの重みセットを用いるか決めていたが，この方式では ADDR Perc. の出力によってどの重みセットを用いるか決めている．ADDR Perc. にいわゆるセレクタの役割を負わせようというものである．

入力　GBH Perc. は Jiménez らの方式などと同様である．ADDR Perc. は分岐命令アドレスの各ビット 1/0 それぞれに対し 1/-1 を与える．

重み　GBH Perc.，ADDR Perc. 共に Jiménez らの方式などと同様である．

4.3.2 出力関数

一般の多層のニューラル・ネットワークではシグモイド関数のような値域が (0, 1) のものが用いられるが，前述したように分岐予測器ではそのような複雑な関数は使えない．そこで，それを近似するような，分岐予測器でも使えそうなものとして数種類の出力関数を実装した．各ニューロンで別々の出力関数を用いることも可能である．以下，実装した出力関数を記す．

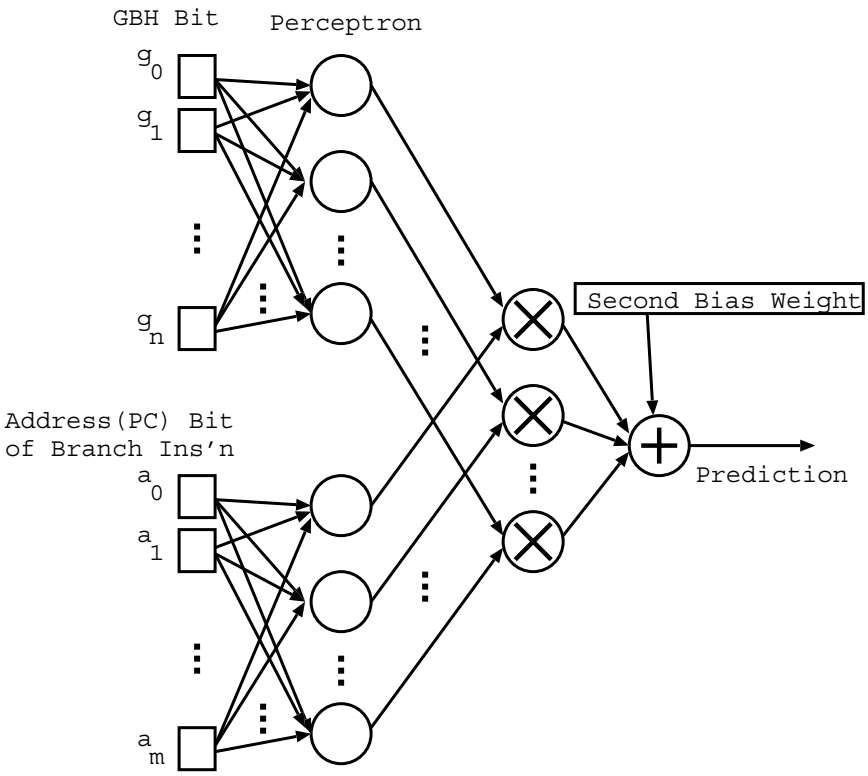

図 6: 重みセット選択ニューロンを用いた方式

linear　$f(u) = u$ とそのまま出力される.

bi-step　単位ステップ関数を 2 つ組み合わせた形をしており，正の閾値 θ に対し

$$f(u) = \begin{cases} 1 & (u \geq \theta) \\ 0 & (\theta > u \geq -\theta) \\ -1 & (u < -\theta) \end{cases} \tag{34}$$

$\theta = 0$ とすると，Jiménez らの方式におけるパーセプトロンの出力関数 (式 (29)) と同じになる．以下その関数を sign と呼ぶ.

quad-step　さらに bi-step を 2 つ組み合わせた形をしており，正の閾値 $\theta_1, \theta_2 (\theta_1 > \theta_2)$ に対して

$$f(u) = \begin{cases} 2 & (u \geq \theta_1) \\ 1 & (\theta_1 > u \geq \theta_2) \\ 0 & (\theta_2 > u \geq -\theta_2) \\ -1 & (-\theta_2 > u \geq -\theta_1) \\ -2 & (-\theta_1 > u) \end{cases} \tag{35}$$

bi-step よりもとる値の幅が大きく，積和の絶対値が大きいものは強く発言で

きることになる.

select-one　ADDR Perc.，GBH Perc. などの各ニューロンの積和のうち，絶対値が最大のものに対し sign を用い，他のものは出力を 0 とする．例えば ADDR Perc. にこれを使うと，GBH Perc. の 1 つの出力のみが予測に用いられることになる (他はゼロになるので予測に影響を与えなくなる).

scale　それなりに積和の絶対値が大きいものには発言させる．selrct-one の条件を緩めた形となる．ADDR Perc.，GBH Perc. などの各ニューロンの積和のうち，絶対値が最大のものを u_{max} として，bi-step の閾値を

$$\theta = \frac{|u_{max}|}{2^x}, (x \in \mathbf{N}) \tag{36}$$

としたものである.

4.3.3　出力計算と学習

一般の多層のニューラル・ネットワークでは，誤差信号は各層の出力から得るが，本方式では出力時に用いたものとは異なる出力関数を積和に用い，そこから誤差信号を得ることも考えられる．例えば，出力時は sign を，学習時は select-one を用いるといった具合である．以下，GBH Perc.，ADDR Perc. それぞれの出力時の出力関数を f_g, f_a，学習時の出力関数を b_g, b_a とし，区別する.

なお，予測の仕方は Jiménez らの方式に則り，用いる重みセットを ADDR Perc. に選択させるという方針から，f_g は linear，出力層の出力関数 f_s は sign とする.

出力計算　まず ADDR Perc.，GBH Perc. それぞれで積和 $u_a^{(i)}, u_g^{(i)} (i = 1 \cdots n)$ を求める:

$$u_g^{(i)} = \sum_j g_j w_{ij}, u_a^{(i)} \sum_j a_j v_{ij} \tag{37}$$

それをそれぞれの出力関数 f_a, f_g を通して出力 (次段への入力) $o_a^{(i)}, o_g^{(i)}$ を得る:

$$o_a^{(i)} = f_a(u_a^{(i)}), o_g^{(i)} = f_g(u_g^{(i)}) \tag{38}$$

SEC Perc. でその積和を求め，f_s に通して最終的な出力 y を得る.

$$u_s = \sum_{i=0}^{n} o_a^{(i)} o_g^{(i)} \tag{39}$$

$$y = f_s(u_s) \tag{40}$$

バイアスが加えられていることに注意されたい.

学習　Jiménez らの方式の学習則に BP を組み合わせたものになる．ADDR Perc., GBH Perc. のそれぞれに対して，予測を誤るか，積和 u_a, u_g の絶対値が閾値以下であった場合に行われる．ADDR Perc.，GBH Perc. それぞれの重みの修正量は，対となったニューロンの出力をあたかも重みであるかのように捉え，計算する．以下は ADDR Perc. の場合を示すが，GBH Perc. の場合も同様である．出力層の誤差信号 δ は教師信号 t と等しいので，番号 $i(i = 1 \cdots n)$ のニューロンの j 番目の重み $w_i j$ の修正量は

$$\Delta w_{ij} = \eta t a_i b_g(u_g^{(i)}) \tag{41}$$

となる．SEC Perc. のバイアス重みは，u_s が閾値以下であった場合に t 変化させる．いずれも閾値は Jiménez らの方式に準ずる．

第5章　評価

SimpleScaler ツールセット (ver.3.0) の sim-bpred シミュレータに対して，これまでに上げた分岐予測器を実装し，SPEC CINT95 ベンチマークを用いて予測精度の評価を行った．コンパイラは gcc(ver2.7.2.3) を用いた．最適化オプションは，-O6 -funroll-loops である．

以下 5.1 節で評価モデルについて述べ，5.2 節で評価結果について述べる．

5.1　評価モデル

sim-bpred シミュレータでは命令は逐次実行され，分岐命令が現れると実際の分岐結果が分かるまで次の命令はフェッチされない．

対応する分岐命令のエントリが BTB に存在しない場合は not taken と予測するようにソースコードを変更してある．表 1 に，各種パラメタを示す．

BTB	512set 4way
グローバル分岐履歴長	12
Filter 機構	確信度カウンタ 2bit

表 1: 各種パラメタ

なお，BTB のパラメタ 512set 4way というのは，挙げたベンチマークにおいて BTB ミスヒットの割合が平均 0.1% という非常に小さいものとなる程度のも

ベンチマーク	入力
099.go	9 9
126.gcc	genrecog.i
129.compress	100 q 2131
132.ijpeg	vigo.ppm -GO
134.perl	primes.pl primes.in
147.vortex	vortex.in

表 2: ベンチマーク

のである.

グローバル分岐履歴長 12 というのは Alpha21264 の分岐予測器のものと同じである.

タグ付き重みセットを用いた方式の重みテーブルは 4way 固定とした.

5.2 評価結果

5.2.1 タグ付き重みセットを用いた方式

図 7 に，各ベンチマークの予測精度の平均をとったものを表す．重みセットの数は左の点からそれぞれ $8, 16, 32, \cdots$ と倍々になっている.

Jiménez らの方式では，同程度のメモリ量で比較すると Filter 機構を用いない方が予測精度が高いという結果が出た．そのため図 7 では煩雑になるのを避けるため Filter 機構を用いた場合の Jiménez らの方式の予測精度は省いた.

Filter 機構，重みテーブルのキャッシュ化を併用した タグ付き重みセットを用いた方式 (図中ロ) のみ，メモリ量 1〜6KB 程度の領域で Jiménez らの方式 (図中米印) よりも最大 0.2%程度予測精度が高くなった．以下，Jiménez らの方式と比べ予測ヒット率が上回らなかった部分について述べる.

高メモリ量の領域　高メモリ量の領域では Jiménez らの方式では競合が，タグ付き重みセットを用いた方式ではエントリの置き換えが少なくなるため両者の差が少なくなり，Filter 機構の効果も薄くなっていく．その上，Filter 機構のバイアスビットよりも予測器による予測の方が正確なので，Filter 機構がない方が予測精度が高くなっていく.

Filter 機構により，(バイアスビット + 確信度カウンタ)×(エントリ数)b つまり

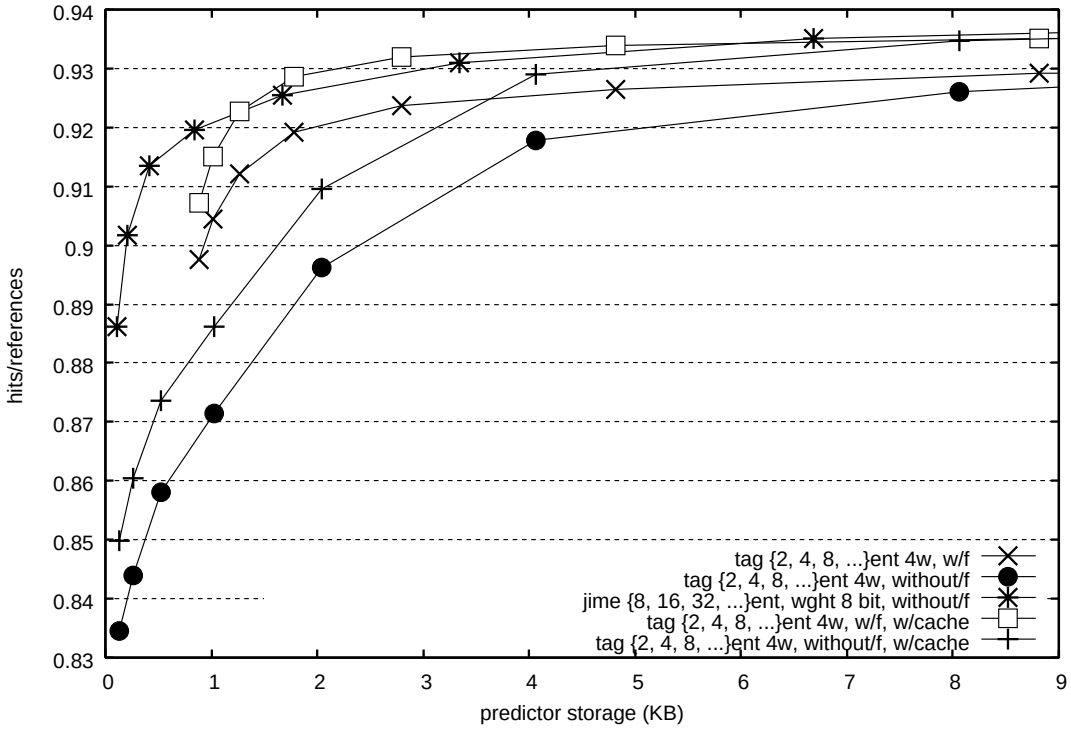

図 7: 評価結果

この場合は $(1 + 2) \times (512 \times 4) = 6$Kb，更にはタグの分のメモリ量の差がつき，同メモリ量で比較すると Jiménez らの方式の方が高い予測ヒット率を挙げることとなる．

低メモリ量の領域　Jiménez らの方式も，重みセットの数を増やす方が高い予測ヒット率を出している．このことは重みセットの共有が予測ヒット率に悪影響を及ぼしていることを示すが，Filter 機構もキャッシュもない純粋なタグ付き重みセットを用いた方式はどの領域でも Jiménez らの方式に勝る予測ヒット率を出すことはできなかった．Filter 機構を用いることで予測ヒット率は Jiménez らの方式と比べ改善されるが，その分の容量の増加で改善分が相殺されている．競合時に，学習が混ざった重みセットによる予測を行う方が，無条件で not taken と予測するより予測ヒット率が下がらないこととなる．競合によって学習が混ざることよりも，学習の破棄の方が予測ヒット率に影響を与えるという結果となった．

5.2.2　重みセット選択ニューロンを用いた方式

重みの初期値は、8b 符号付き整数の最大/最小値の約 10%である +12 から −12 までの任意の乱数で与えた．

Jiménez らの方式同様の学習になるよう b_g:sign とした．f_a, b_a:bi-step あるいは quad-step としてそれぞれの閾値を変化させながら perl で予測ヒット率を測ったところ，Jiménez らの方式の重みセット数 8 の場合の予測ヒット率 0.9555 に対し，GBH Perc.8 + ADDR Perc.8 で予測ヒット率 0.9 を超える事ができなかった (0.8680 程度).

ADDR Perc. の出力　多少学習時にぶれがあっても，学習が済めば適切な GBH Perc. を選択するようになっているはずである．そうなれば似た分岐傾向の分岐命令は似た累積の分布を持つと考えられる．そこで，ADDR Perc. が適切な GBH Perc. を選択しているかを確かめるために，各分岐命令ごとの ADDR Perc. の積和の正と負のそれぞれの累積を求めた.

結果その分岐命令の傾向に関係なく累積の分布は似た形となり，似た分岐傾向の分岐命令同士で同じ重みセットを用いるという働きがなされていないことが分かった.

適切な学習への課題　似た分岐傾向の分岐命令同士で同じ重みセットを用いるようになる学習が GBH Perc.，ADDR Perc. 双方に必要である.

第 6 章　おわりに

本稿では，複数の分岐命令で重みセットを共有する Jiménez らの方式を改良する提案を 2 つ示した.

各分岐命令に専用の重みセットを用意するタグ付き重みセットを用いた方式では，メモリ量 1〜6KB 程度の領域で Filter 機構とキャッシュ化を併用した場合に Jiménez らの方式よりも最大 0.2%程度高い予測ヒット率を得られた.

分岐傾向の似た分岐命令同士で同じ重みセットを共有させようとした重みセット選択ニューロンを用いた方式では perl での評価で同程度のニューロン数の Jiménez らの方式に勝る予測ヒット率を得られなかった．適当な重みセットを共有させるために，一般の BP とは違った学習則が必要だと考えられる.

謝辞

本研究の機会を与えて頂いた富田眞治教授に深く感謝の意を表します．また，本研究に関して日頃より技術的，精神的な面から多大なる御指導を頂いた五島正裕助手に心より感謝いたします.

さらに，数多くの技術的，精神的な面での助言をいただいた京都大学情報学研究科通信情報システム専攻富田研究室の皆様に心より感謝いたします.

参考文献

[1] Jiménez, D. and Lin., C.: Neural Methods for Dynamic Branch Prediction, *ACM Trans. Comp. Sys.*, Vol. 20, pp. 369–397 (2002).

[2] Jiménez, D. and Lin., C.: Dynamic Branch Prediction with Perceptrons, *Proc. 7th Int'l Symp. on High Performance Computer Architecture (HPCA7)*, pp. 197–206 (2001).

[3] Chang, P. Y., Evers, M. and Patt., Y. N.: Improving Branch Prediction Accuracy by Reducing Pattern History Table Interference, *Int'l Conf. of Parallel Architectures and Compilation Techniques (PACT96)* (1996).

[4] 斎藤史子, 北村健志, 山名早人: 投機的実行に関する最新技術動向, 情報処理学会研究報告 2001–ARC–145, pp. 67–72 (2001).

[5] 斎藤史子, 北村健志, 山名早人: ハイブリッド分岐方向予測機構の性能比較, 情報処理学会研究報告 2002–ARC–150, pp. 89–94 (2002).

[6] 斎藤史子, 山名早人: BTB のエントリ有無を参照した分岐予測器, 先進的計算基盤システムシンポジウム SACSIS 2004, pp. 261–268 (2004).

[7] 坂和正俊, 田中雅博: ニューロコンピューティング入門, 森北出版 (1997).

[8] 富田眞治: 第 2 版コンピュータアーキテクチャ —基礎から超高速化技術まで—, 丸善 (2000).