Neural Branch Predictors

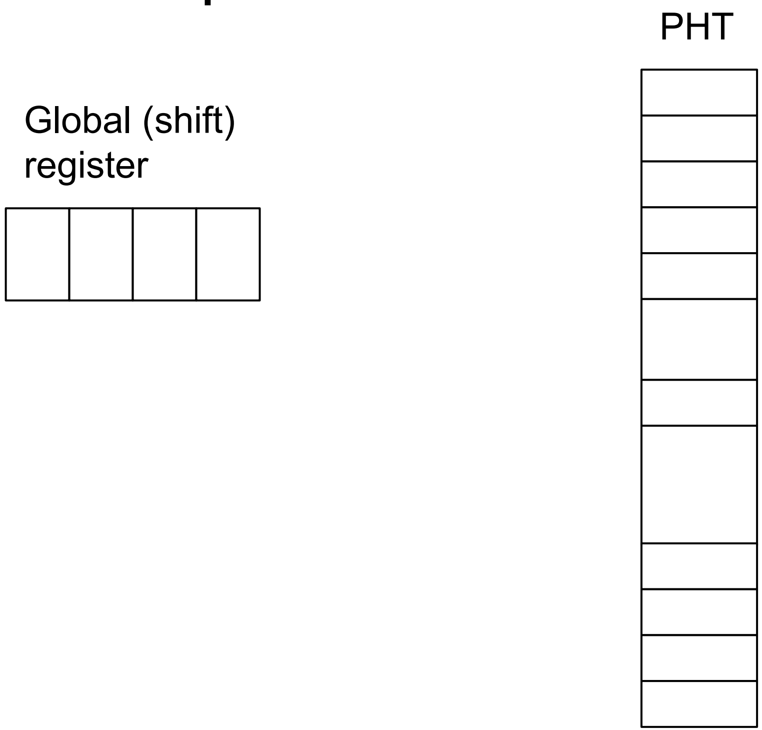
Quinn Pham

You are at the LRT station chatting with a fellow student. She asks, what is the big deal with neural branch prediction? Why did Quinn spend a lecture talking about it? Your train is coming, you can only say a few sentences. What would you tell her?

Two-level branch predictors

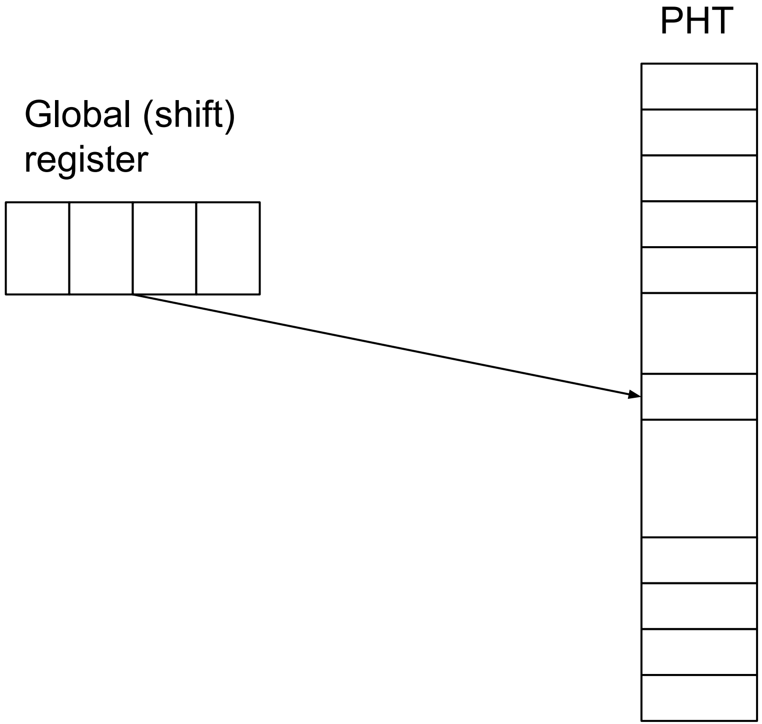
Baer p.139

Two-level branch predictors



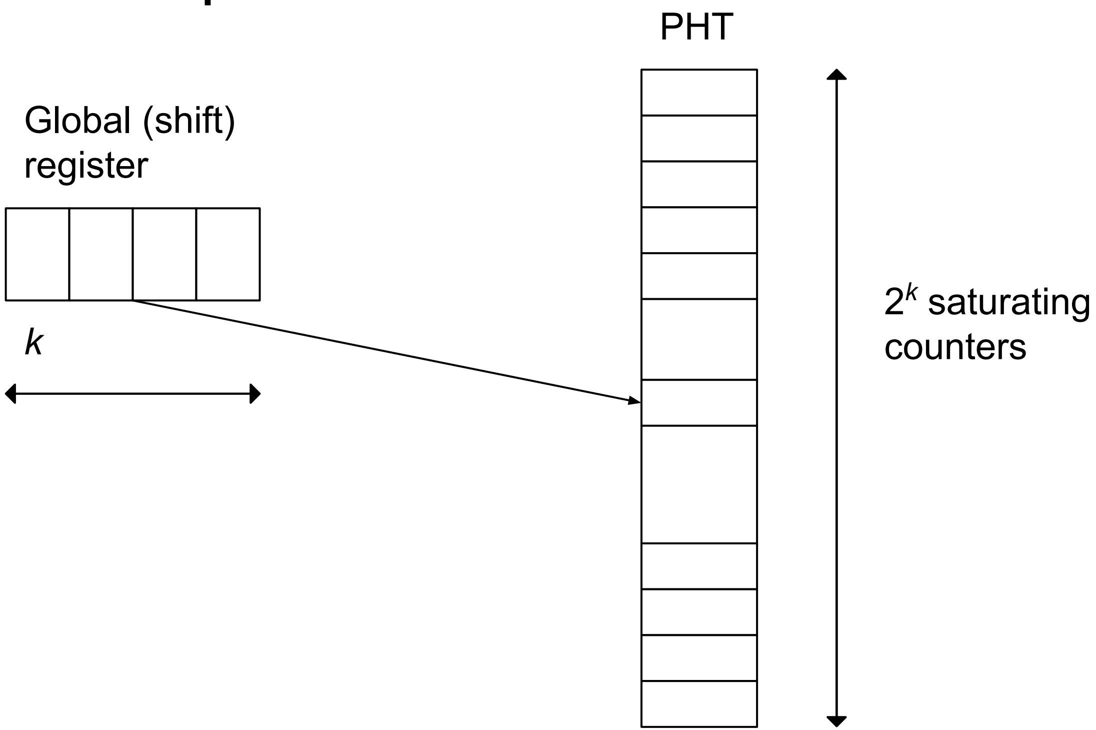
Baer p.139

Two-level branch predictors



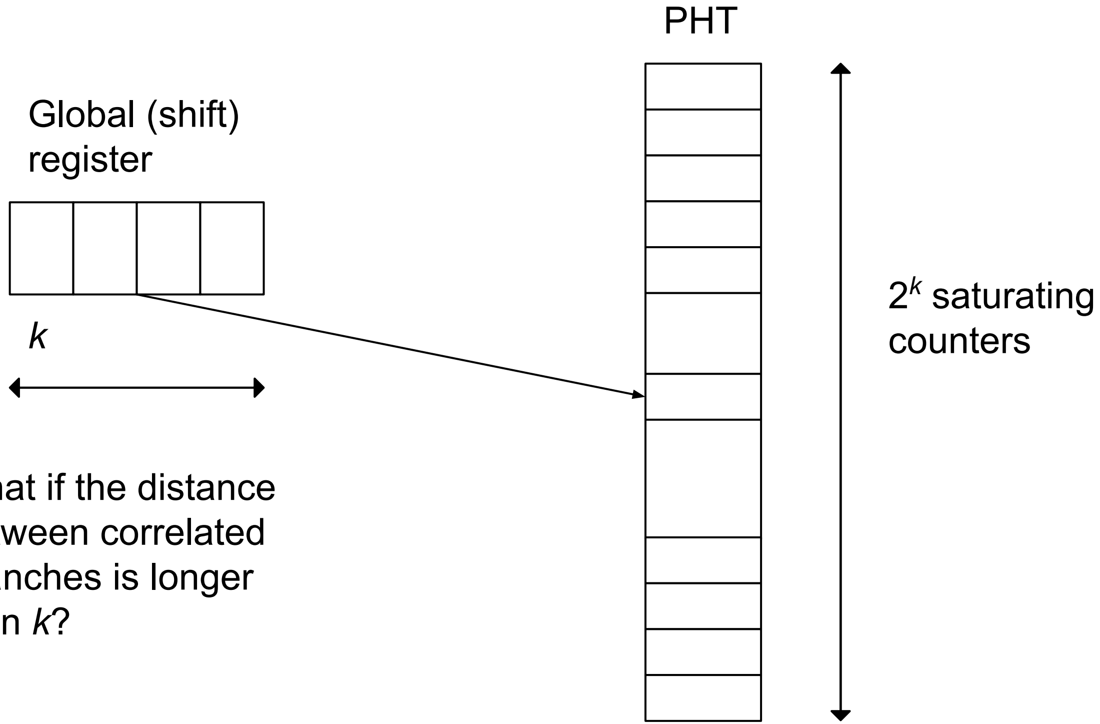
Baer p.139

Two-level branch predictors



Baer p.139

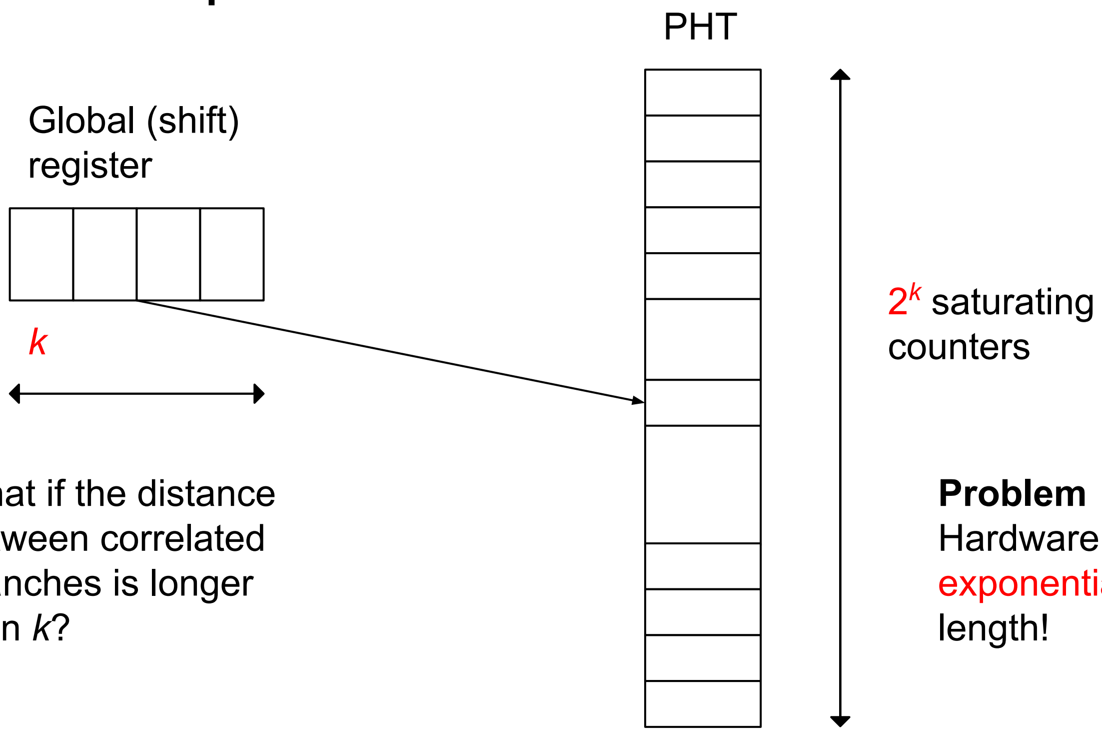
Two-level branch predictors



What if the distance between correlated branches is longer than k?

Baer p.139

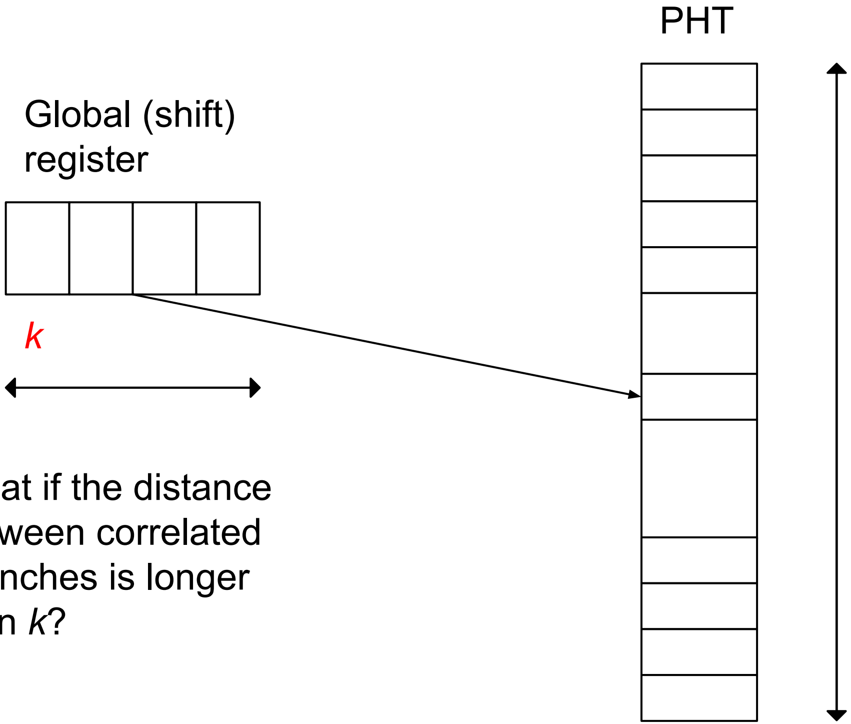
Two-level branch predictors



What if the distance between correlated branches is longer than k?

Problem
Hardware resources scale exponentially with the history length!

Two-level branch predictors



2^k saturating counters

What if the distance between correlated branches is longer than k?

Problem
Hardware resources scale exponentially with the history length!

Cannot consider long history lengths

Baer p.139

Dynamic Branch Prediction with Perceptrons

Daniel A. Jiménez Calvin Lin
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
{djimenez,lin}@cs.utexas.edu

Jiménez 2001

Dynamic Branch Prediction with Perceptrons

Daniel A. Jiménez Calvin Lin
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
{djimenez,lin}@cs.utexas.edu

Replace the pattern history table (PHT) in the two-level predictor with a table of perceptrons

Jiménez 2001

Dynamic Branch Prediction with Perceptrons

Daniel A. Jiménez Calvin Lin
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
{djimenez,lin}@cs.utexas.edu

Replace the pattern history table (PHT) in the two-level predictor with a table of perceptrons

Jiménez 2001

Perceptrons

Jiménez 2002

Perceptrons

- Neural networks learn to compute a function using example inputs and outputs

Jiménez 2002

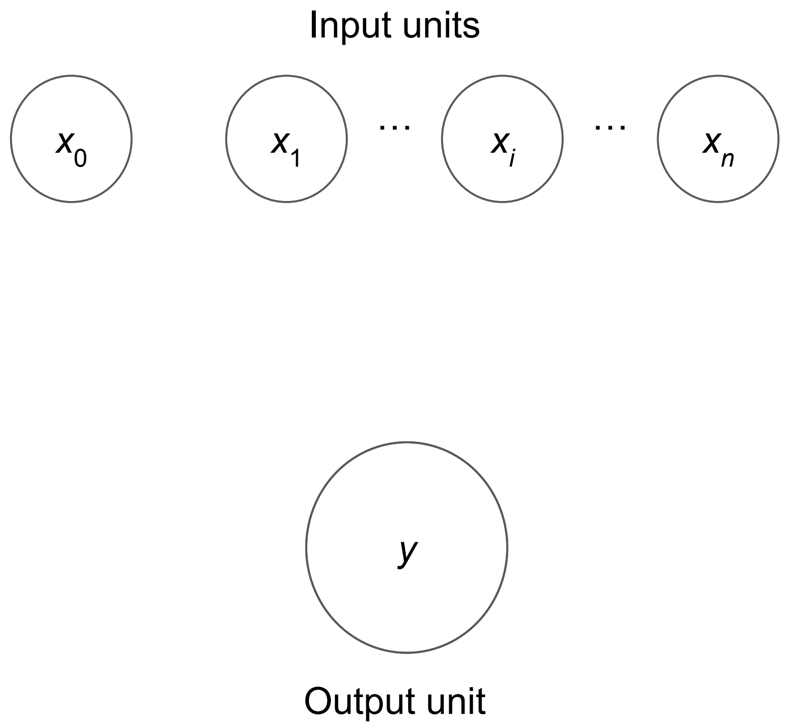
Perceptrons

- Neural networks learn to compute a function using example inputs and outputs
- Single-layer neural network

Jiménez 2002

Perceptrons

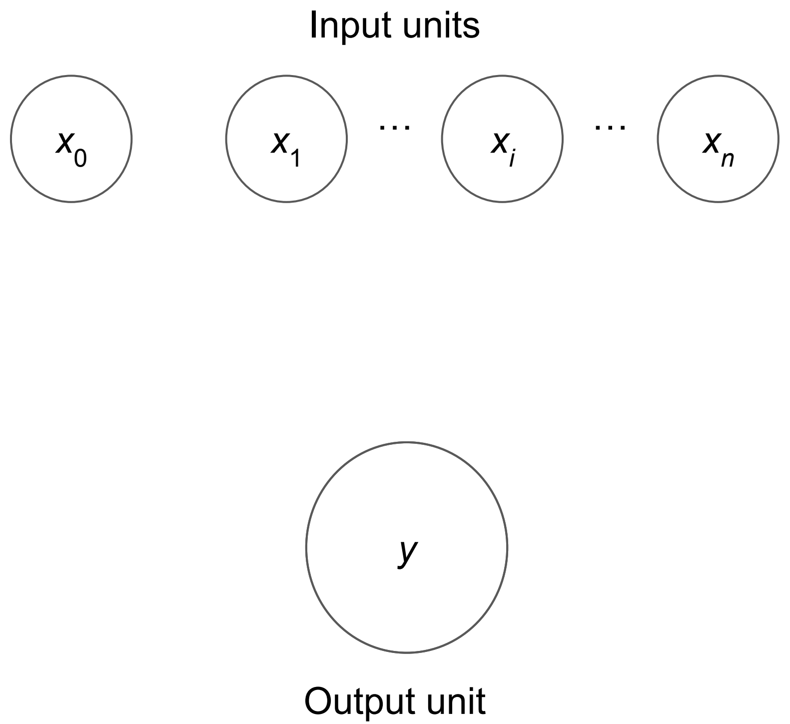
- Neural networks learn to compute a function using example inputs and outputs
- Single-layer neural network



Jiménez 2002

Perceptrons

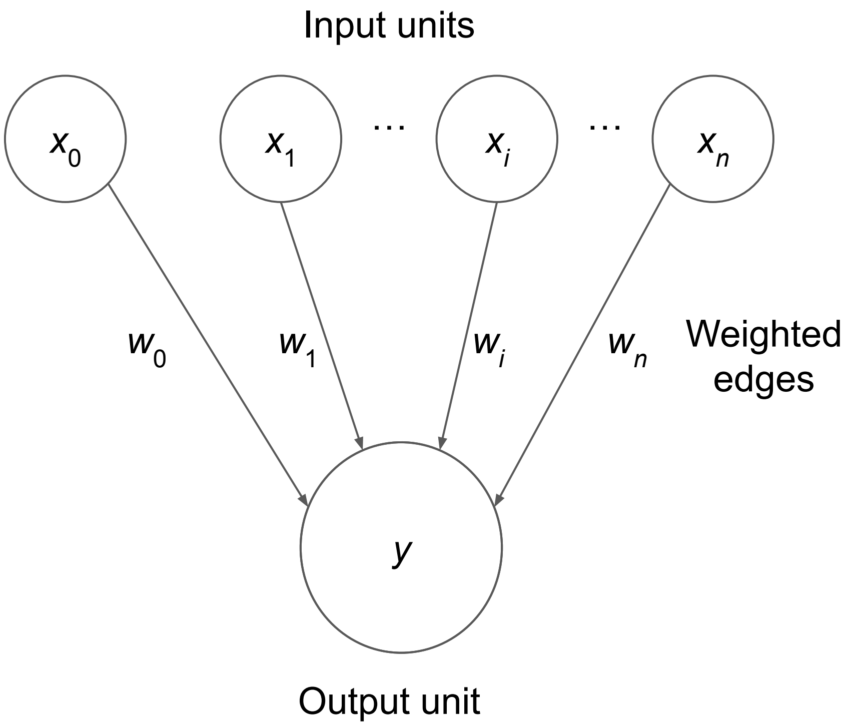
- Neural networks learn to compute a function using example inputs and outputs
- Single-layer neural network
- Data is fed into input unit units and propagated through the network to the output unit



Jiménez 2002

Perceptrons

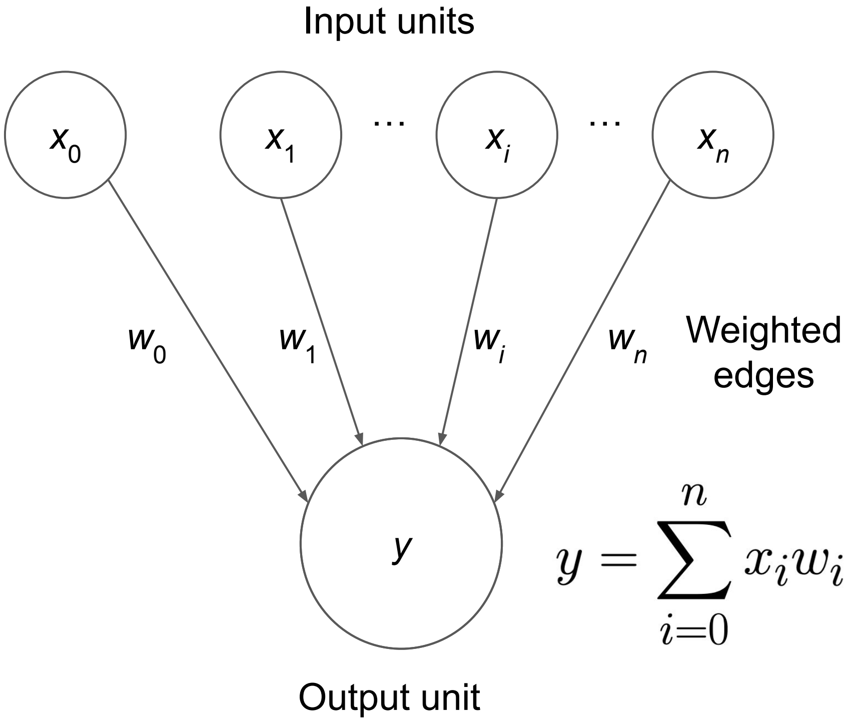
- Neural networks learn to compute a function using example inputs and outputs
- Single-layer neural network
- Data is fed into input unit units and propagated through the network to the output unit



Jiménez 2002

Perceptrons

- Neural networks learn to compute a function using example inputs and outputs
- Single-layer neural network
- Data is fed into input unit units and propagated through the network to the output unit

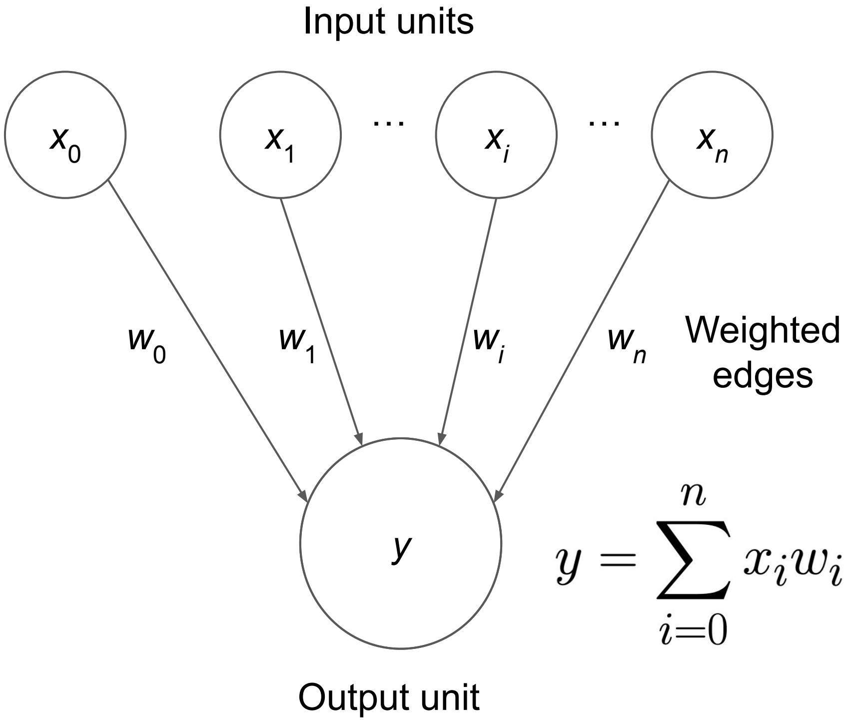


$$y = \sum_{i=0}^{n} x_i w_i$$

Jiménez 2002

Perceptrons

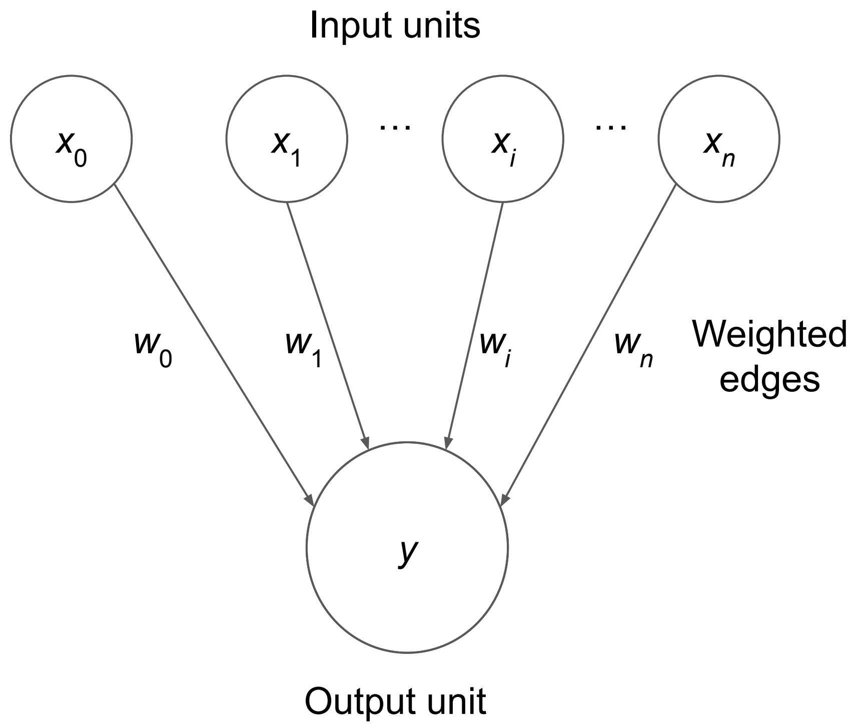
- Neural networks learn to compute a function using example inputs and outputs
- Single-layer neural network
- Data is fed into input unit units and propagated through the network to the output unit
- A training algorithm strengthens or weakens the connections between the input and the output by modifying the weights



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

Jiménez 2002

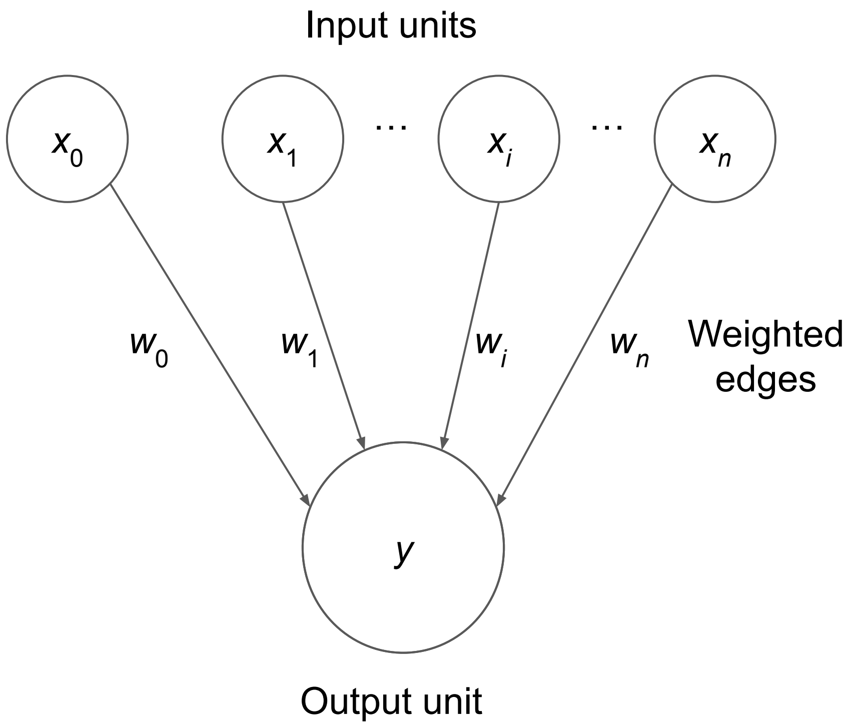
Predicting a Branch



Jiménez 2001

Predicting a Branch

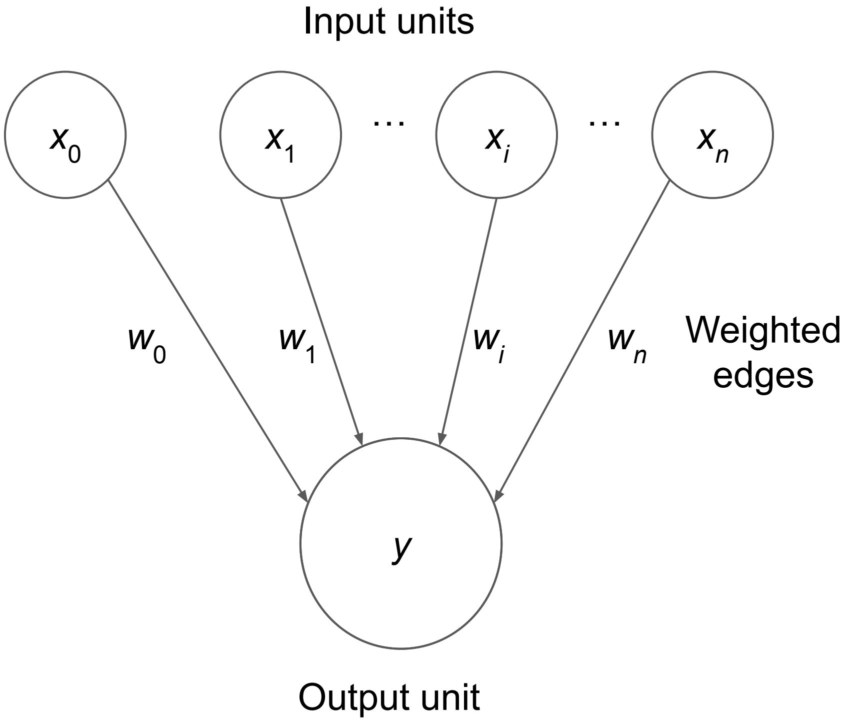
- Input units (x_i)



Jiménez 2001

Predicting a Branch

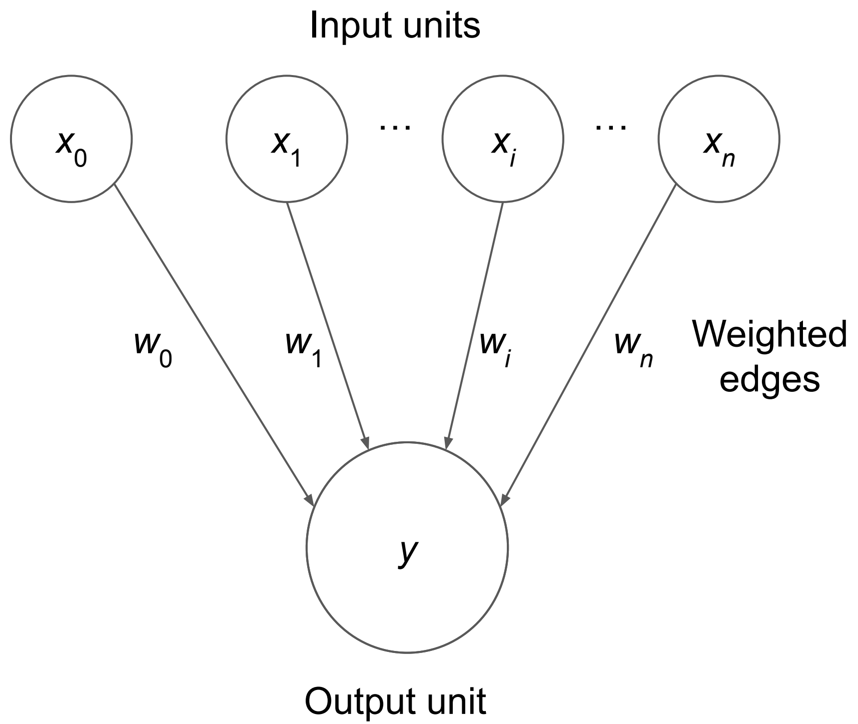
- Input units (x_i)
  - Use bits of global register



Jiménez 2001

Predicting a Branch

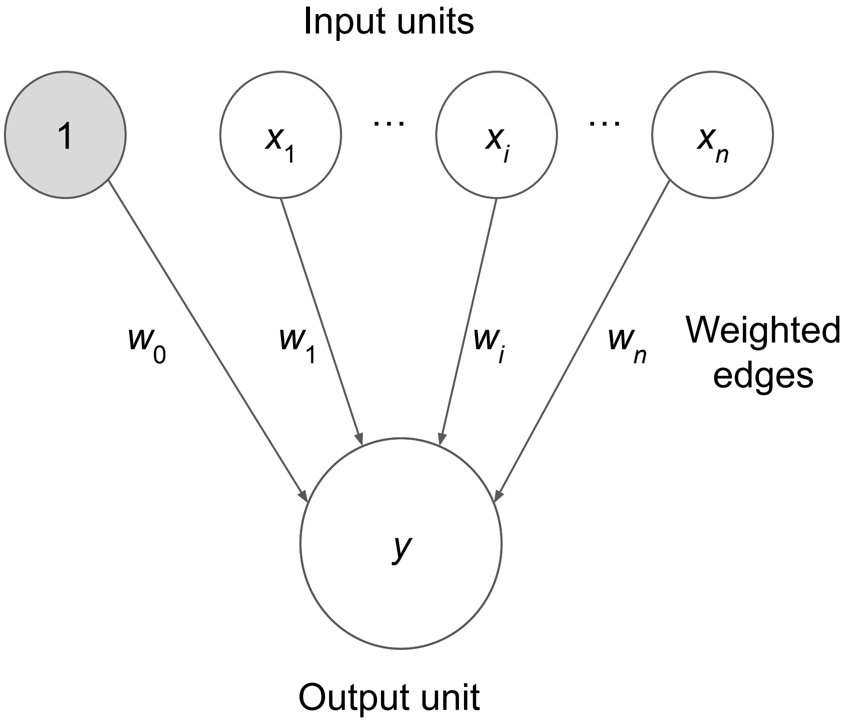
- Input units (x_i)
    - Use bits of global register
        - taken: $x_i = 1$
        - not taken: $x_i = -1$



Jiménez 2001

Predicting a Branch

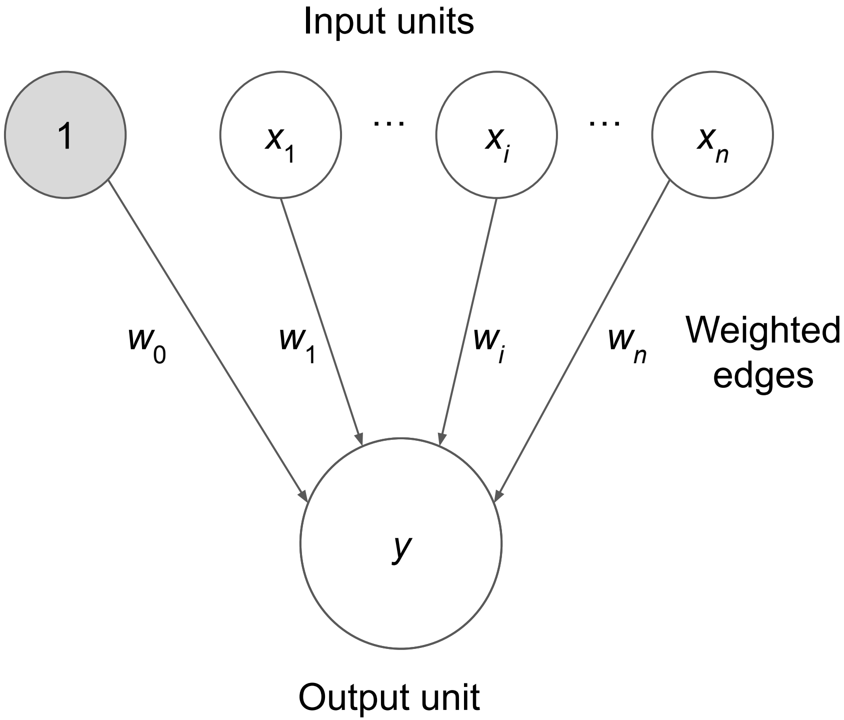
- Input units (x_i)
    - Use bits of global register
        - taken: $x_i = 1$
        - not taken: $x_i = -1$
    - x_0 is always set to 1 as a bias input



Jiménez 2001

Predicting a Branch

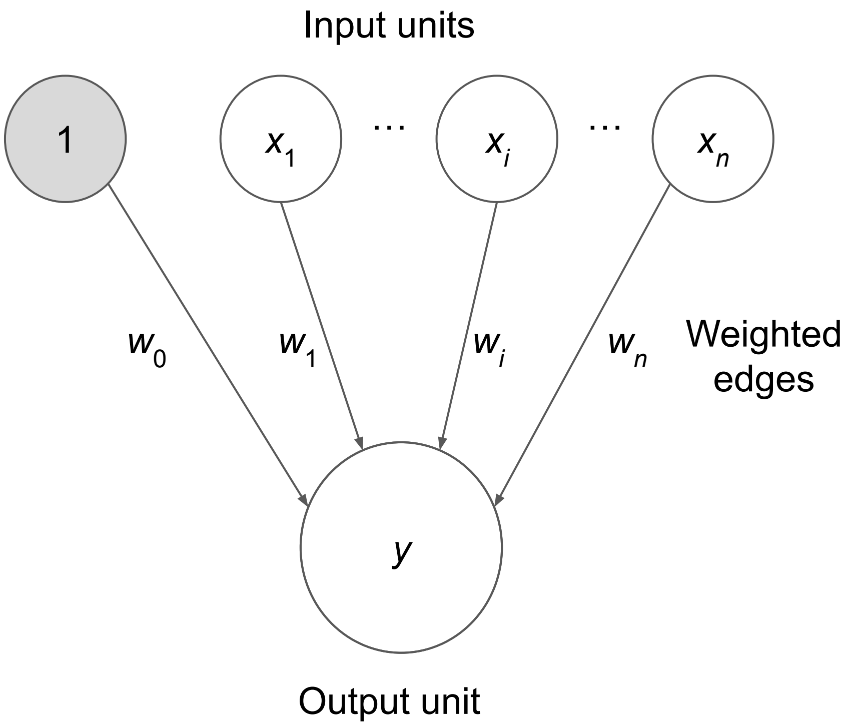
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)



Jiménez 2001

Predicting a Branch

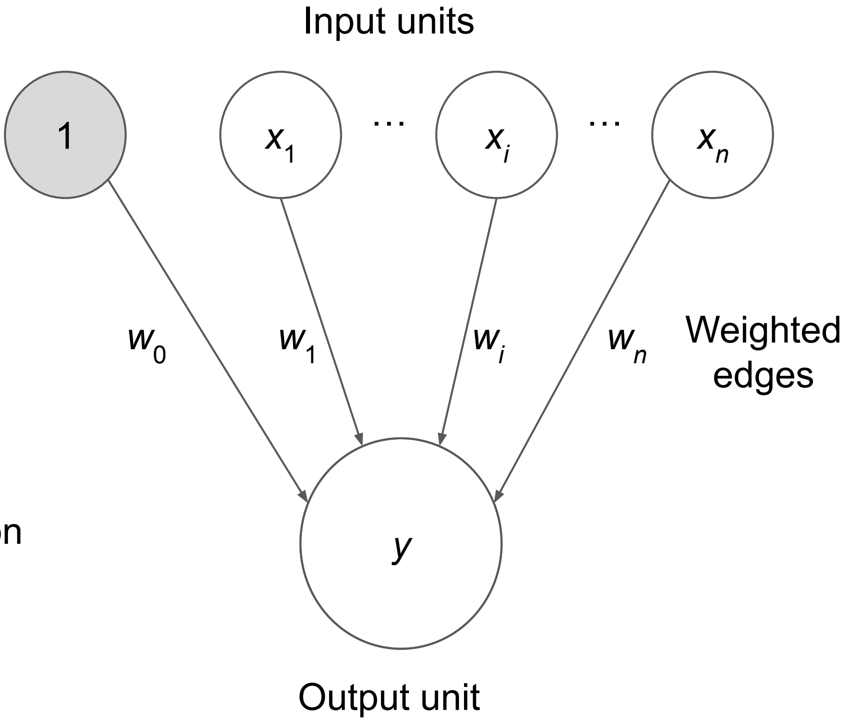
- Input units (x_i)
    - Use bits of global register
        - taken: $x_i = 1$
        - not taken: $x_i = -1$
    - x_0 is always set to 1 as a bias input
- Weights (w_i)
    - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch



Jiménez 2001

Predicting a Branch

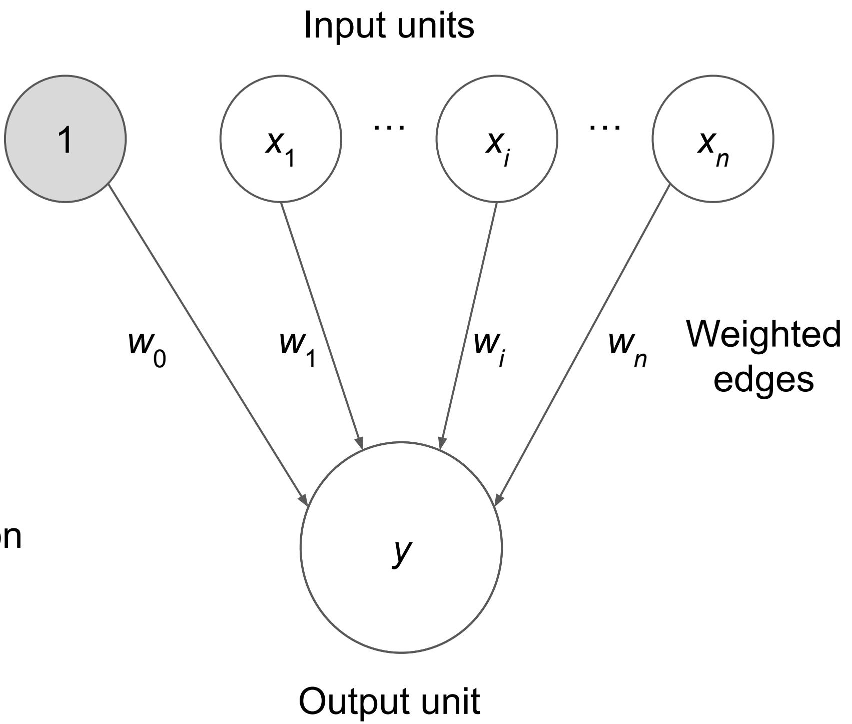
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation



Jiménez 2001

Predicting a Branch

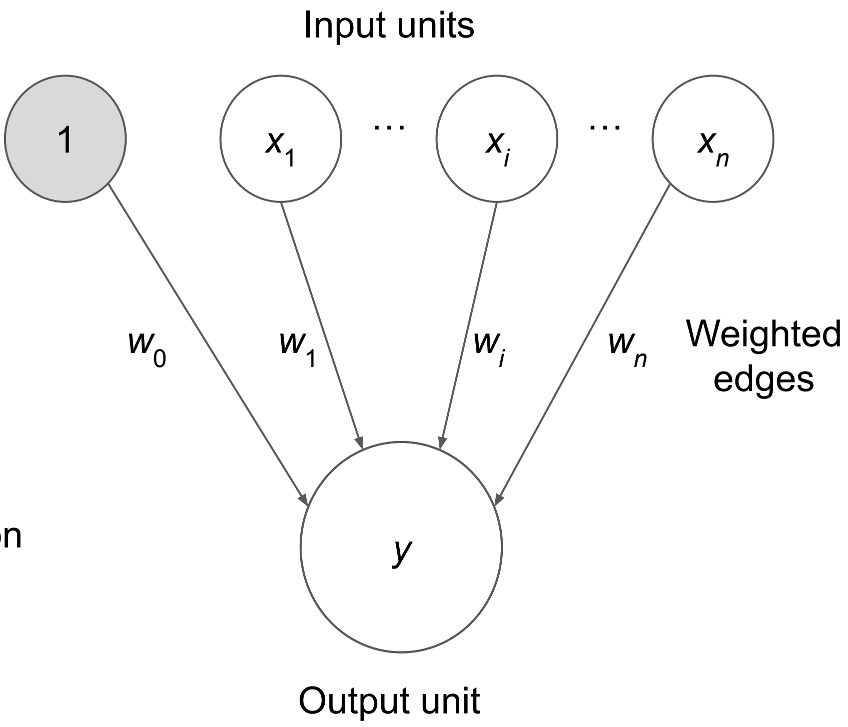
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)



Jiménez 2001

Predicting a Branch

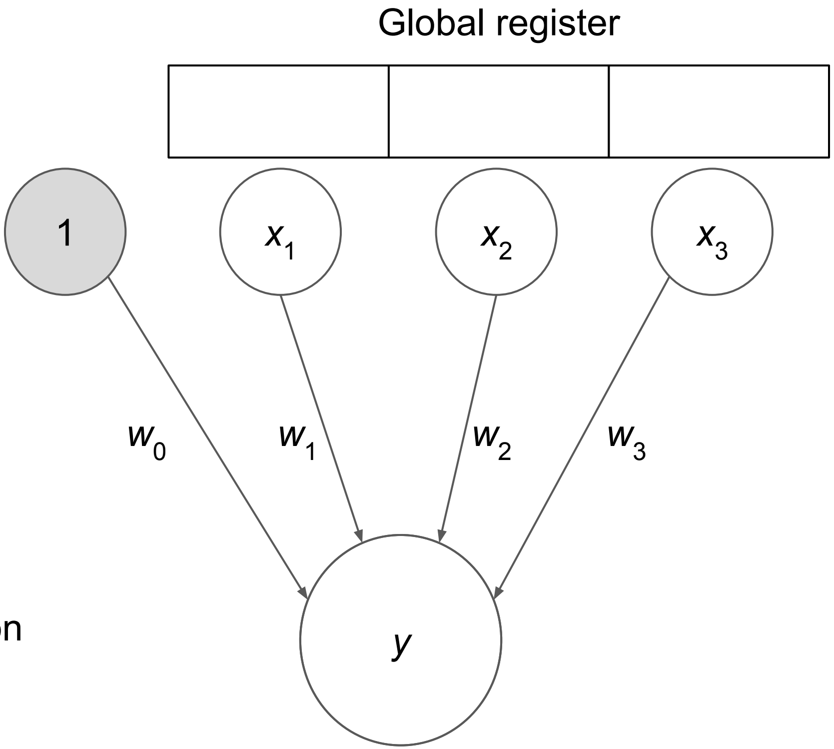
- Input units (x_i)
    - Use bits of global register
        - taken: $x_i = 1$
        - not taken: $x_i = -1$
    - x_0 is always set to 1 as a bias input
- Weights (w_i)
    - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
        - very negative: strong negative correlation
        - very positive: strong positive correlation
        - close to 0: weak correlation
- Output (y)
    - $y < 0$: predict not taken
    - $y \geq 0$: predict taken



Jiménez 2001

Example

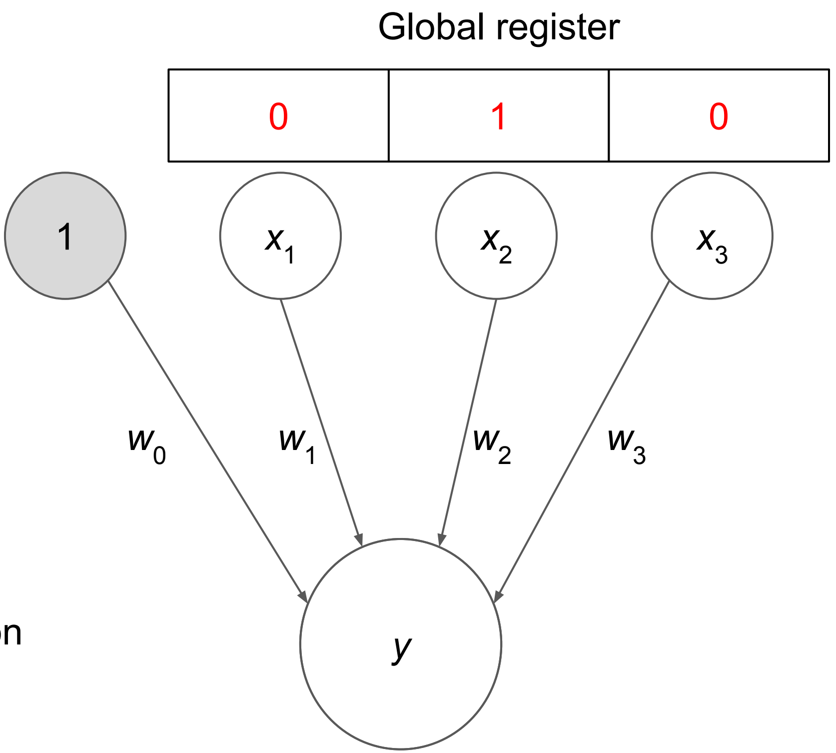
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken



Jiménez 2001

Example

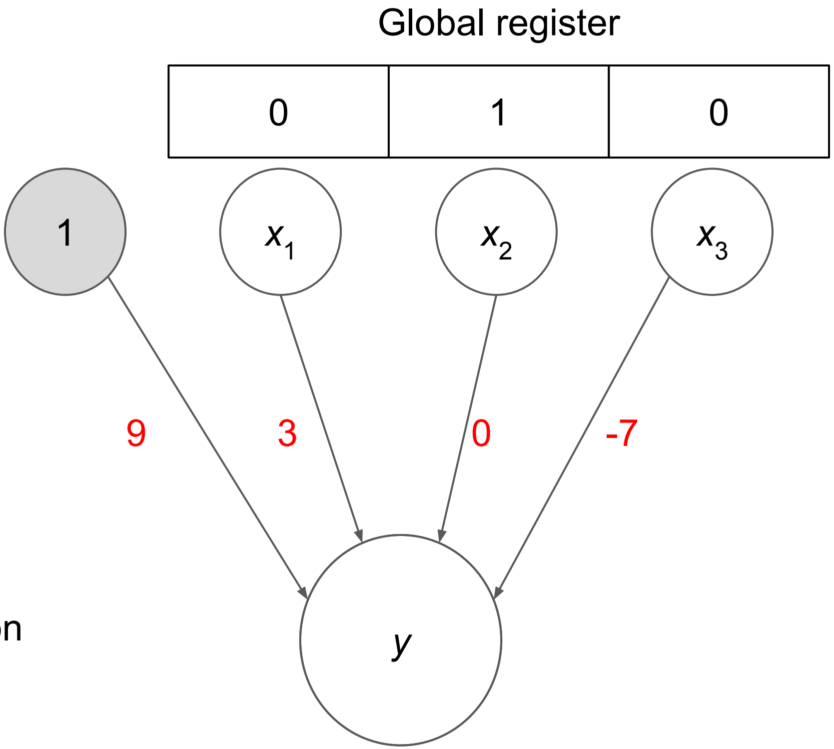
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken



Jiménez 2001

Example

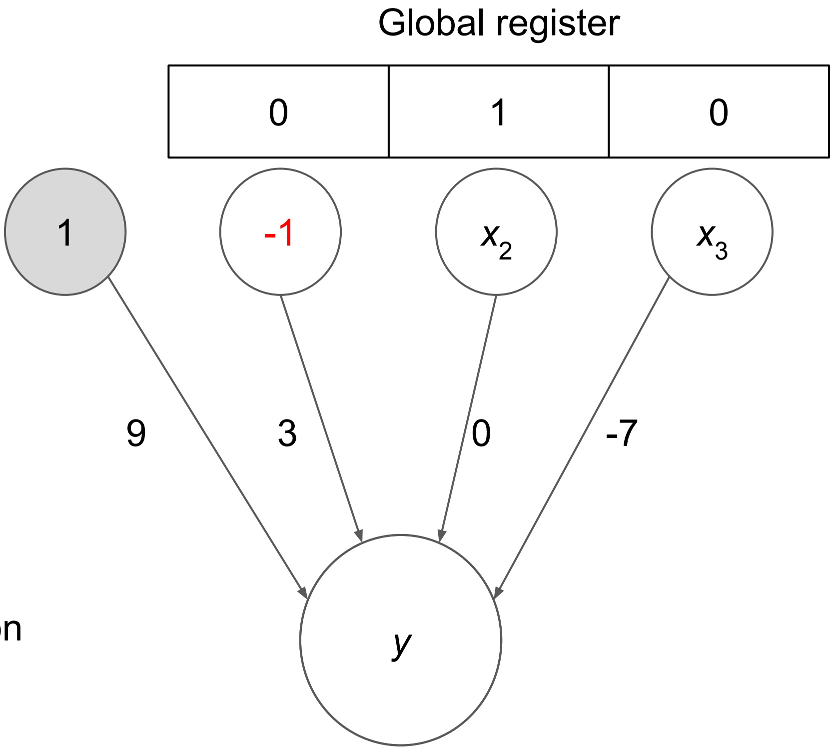
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken



Jiménez 2001

Example

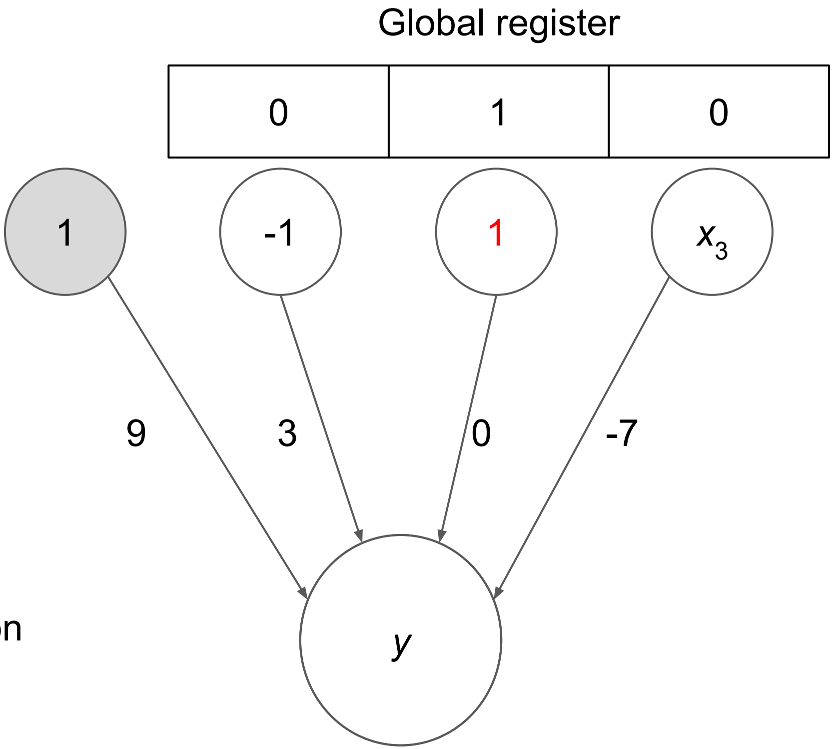
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken



Jiménez 2001

Example

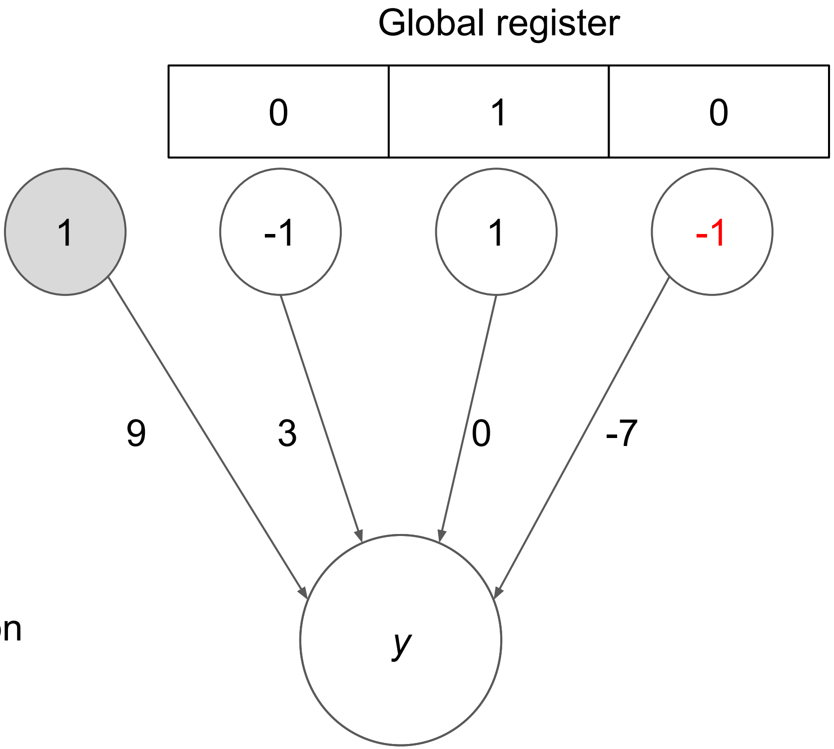
- Input units (x_i)
    - Use bits of global register
        - taken: $x_i = 1$
        - not taken: $x_i = -1$
    - x_0 is always set to 1 as a bias input
- Weights (w_i)
    - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
        - very negative: strong negative correlation
        - very positive: strong positive correlation
        - close to 0: weak correlation
- Output (y)
    - $y < 0$: predict not taken
    - $y \geq 0$: predict taken



Jiménez 2001

Example

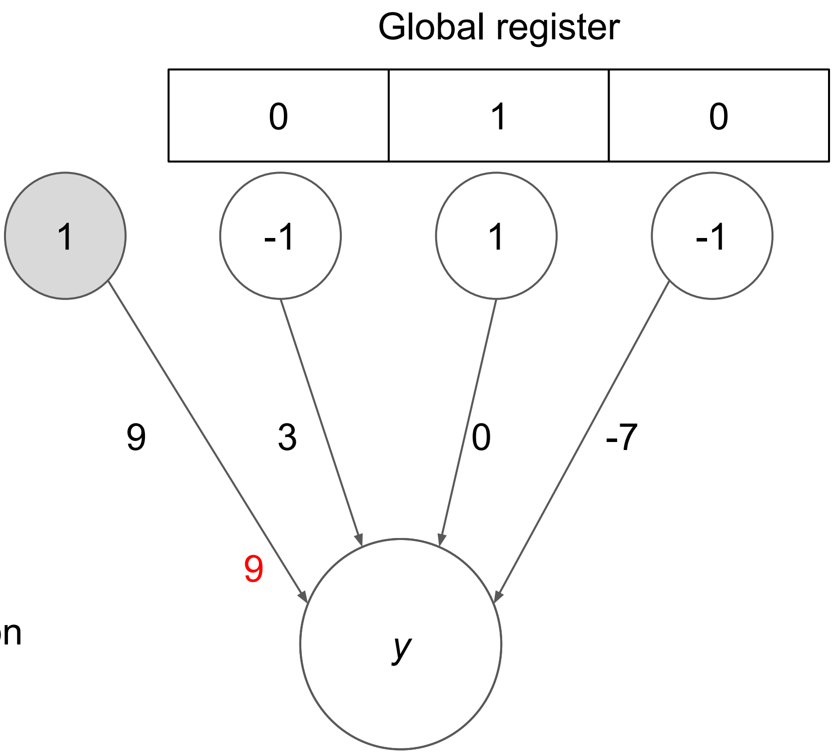
- Input units (x_i)
    - Use bits of global register
        - taken: $x_i = 1$
        - not taken: $x_i = -1$
    - x_0 is always set to 1 as a bias input
- Weights (w_i)
    - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
        - very negative: strong negative correlation
        - very positive: strong positive correlation
        - close to 0: weak correlation
- Output (y)
    - $y < 0$: predict not taken
    - $y \geq 0$: predict taken



Jiménez 2001

Example

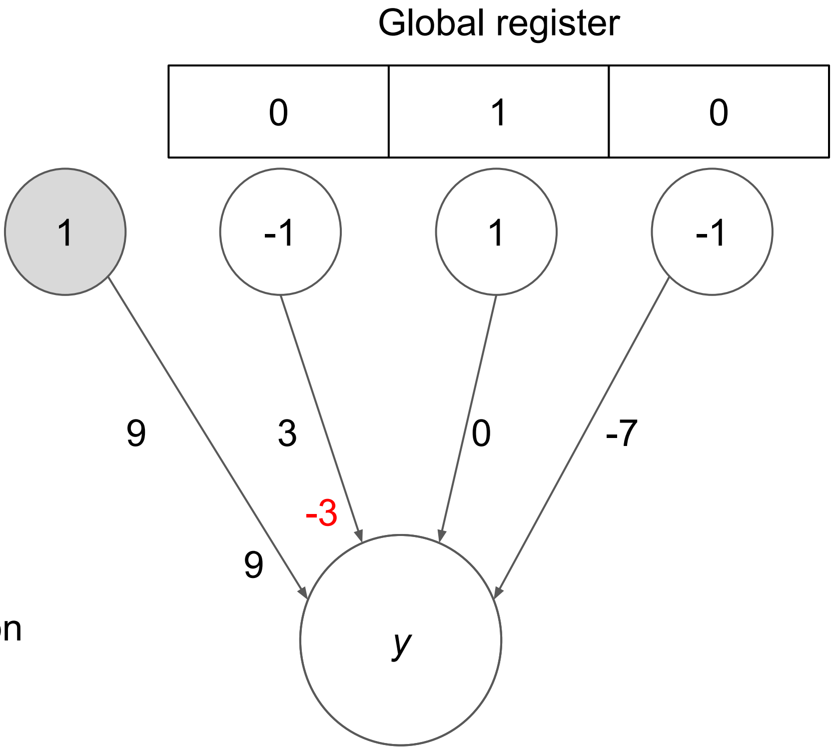
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken



Jiménez 2001

Example

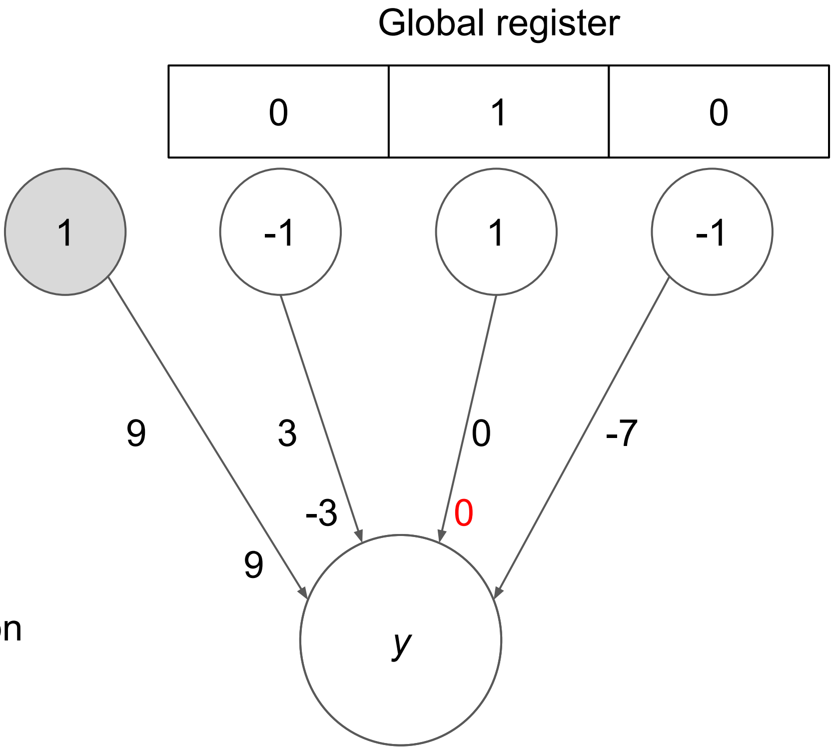
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the i-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken



Jiménez 2001

Example

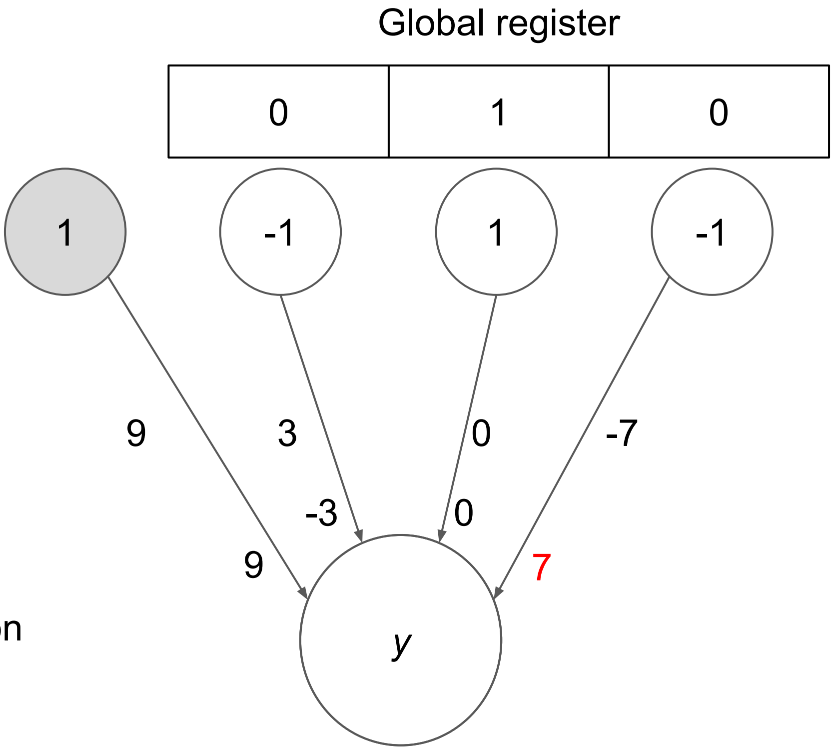
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken



Jiménez 2001

Example

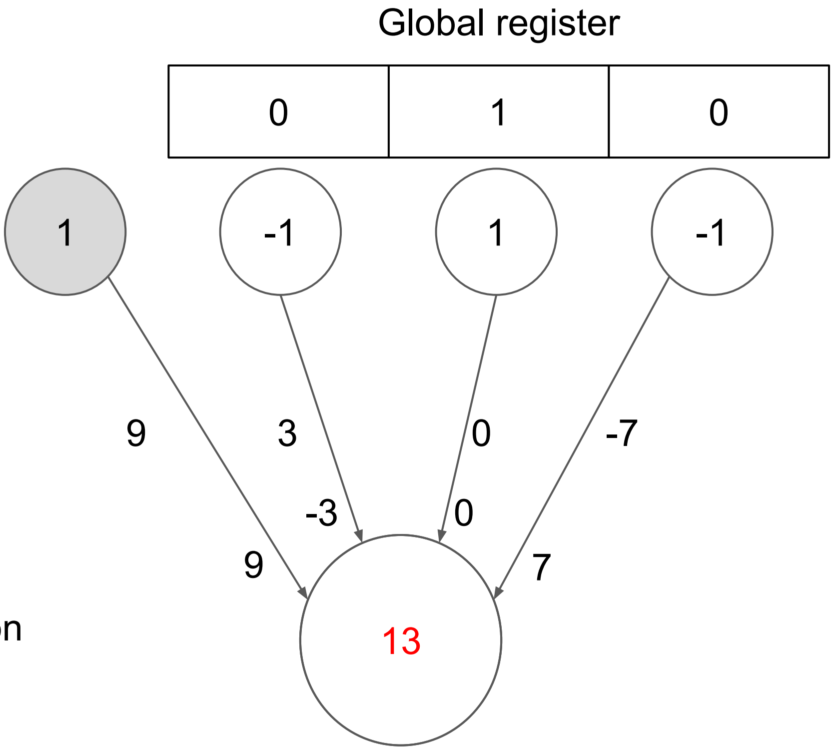
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken



Jiménez 2001

Example

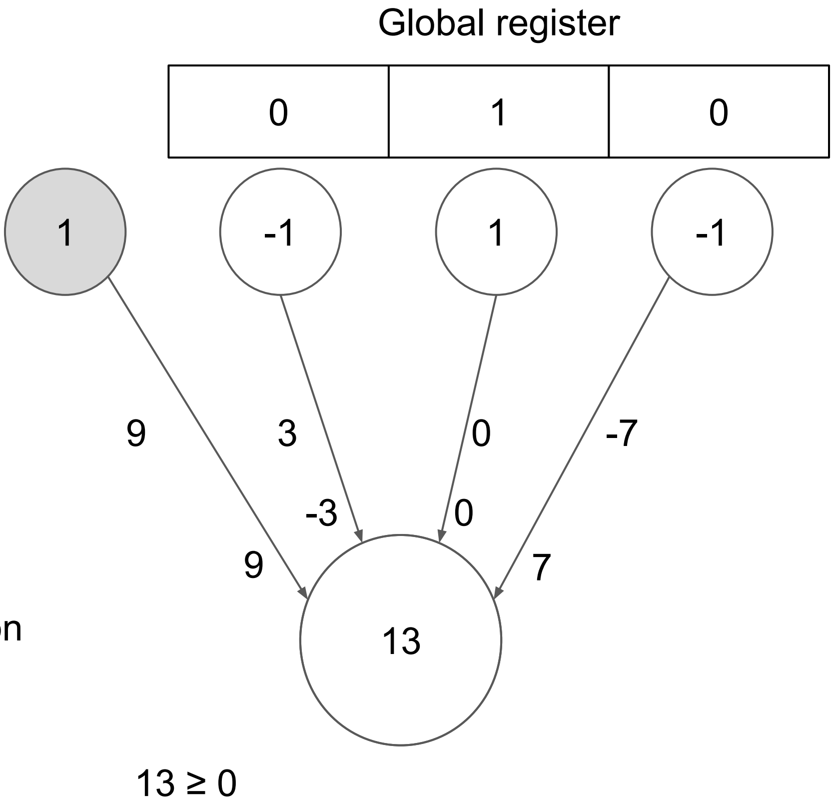
- Input units (x_i)
    - Use bits of global register
        - taken: $x_i = 1$
        - not taken: $x_i = -1$
    - x_0 is always set to 1 as a bias input
- Weights (w_i)
    - Signed integers that represent the correlation between the current branch and the i-th bit of the global register or the bias of the branch
        - very negative: strong negative correlation
        - very positive: strong positive correlation
        - close to 0: weak correlation
- Output (y)
    - $y < 0$: predict not taken
    - $y \geq 0$: predict taken



Jiménez 2001

Example

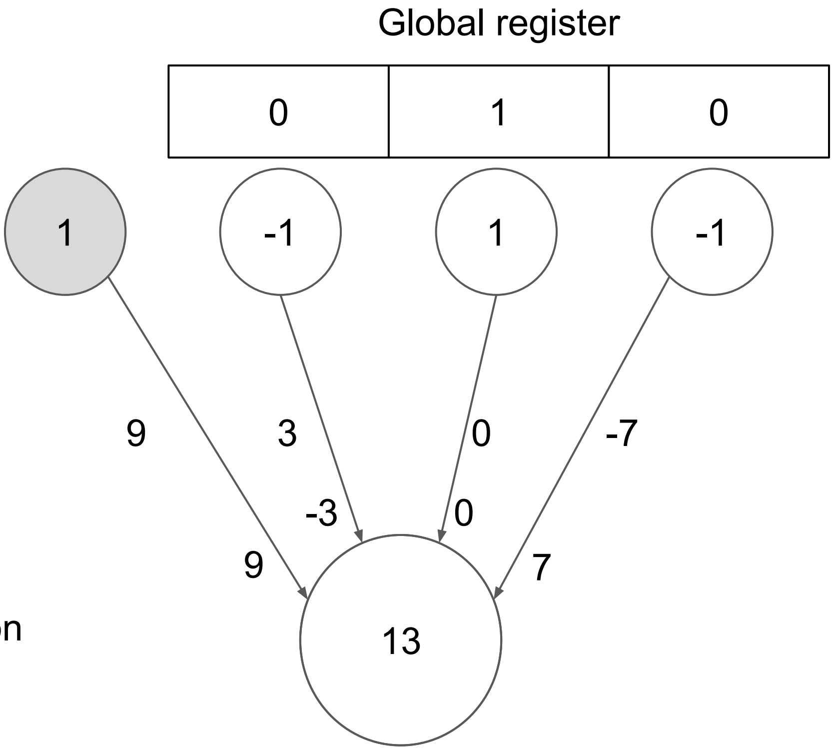
- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken



Jiménez 2001

Example

- Input units (x_i)
    - Use bits of global register
        - taken: $x_i = 1$
        - not taken: $x_i = -1$
    - x_0 is always set to 1 as a bias input
- Weights (w_i)
    - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
        - very negative: strong negative correlation
        - very positive: strong positive correlation
        - close to 0: weak correlation
- Output (y)
    - $y < 0$: predict not taken
    - $y \geq 0$: predict taken

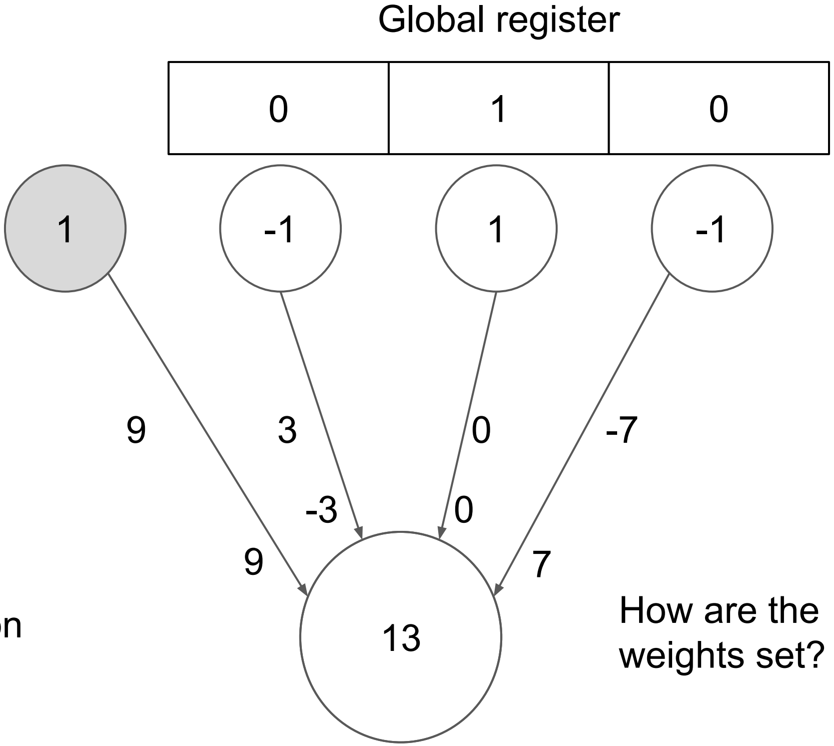


$13 \geq 0 \Rightarrow$ predict taken

Jiménez 2001

Example

- Input units (x_i)
  - Use bits of global register
    - taken: $x_i = 1$
    - not taken: $x_i = -1$
  - x_0 is always set to 1 as a bias input
- Weights (w_i)
  - Signed integers that represent the correlation between the current branch and the *i*-th bit of the global register or the bias of the branch
    - very negative: strong negative correlation
    - very positive: strong positive correlation
    - close to 0: weak correlation
- Output (y)
  - $y < 0$: predict not taken
  - $y \geq 0$: predict taken

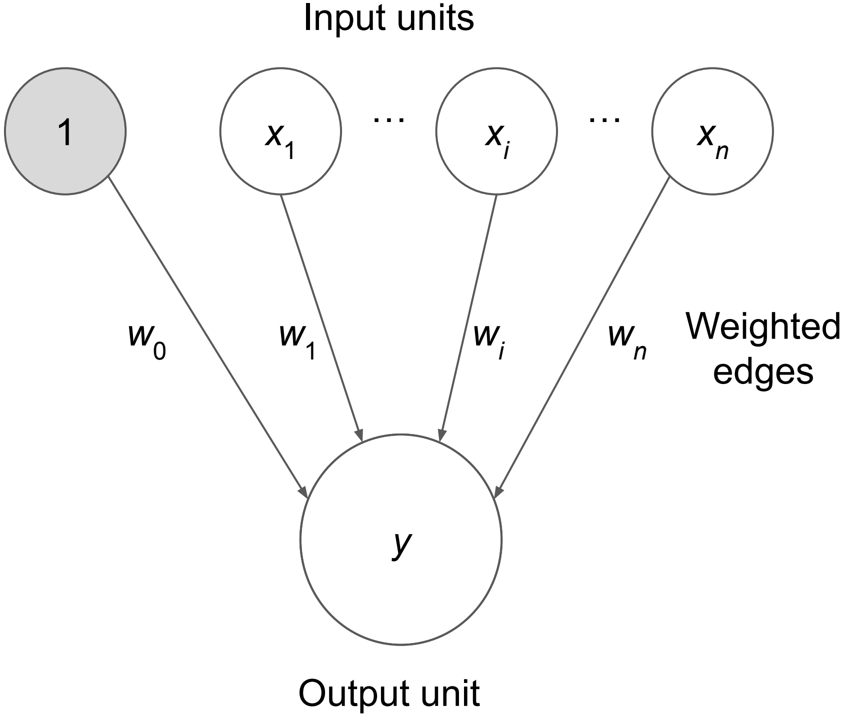


$13 \geq 0 \Rightarrow$ predict taken

How are the weights set?

Jiménez 2001

Training

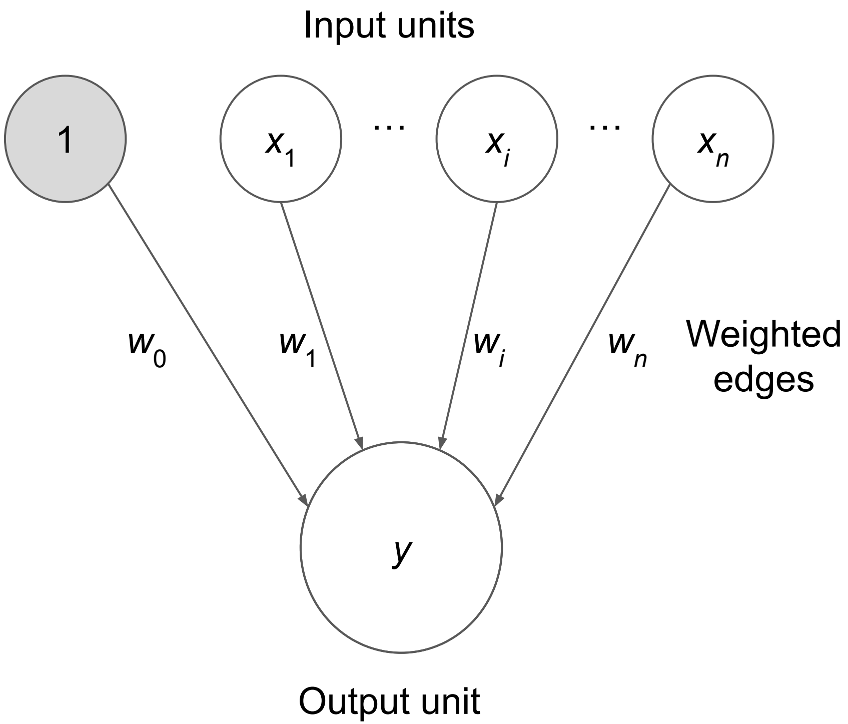


Jiménez 2001

Training

if **then**

end if



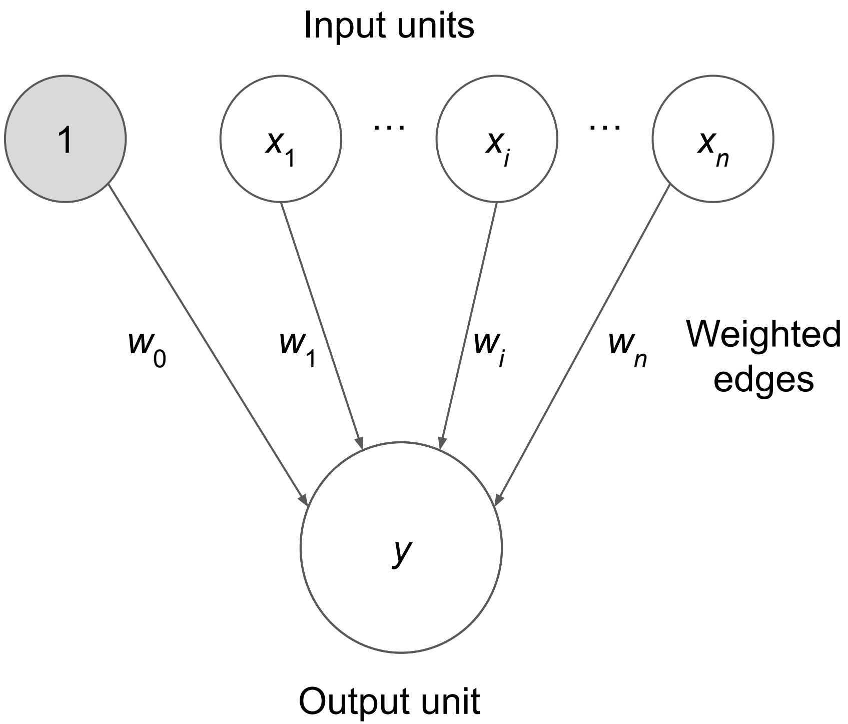
Jiménez 2001

Training

t is 1 if the branch was taken and
-1 otherwise

if $\text{sign}(y) \neq t$ **then**

end if



Jiménez 2001

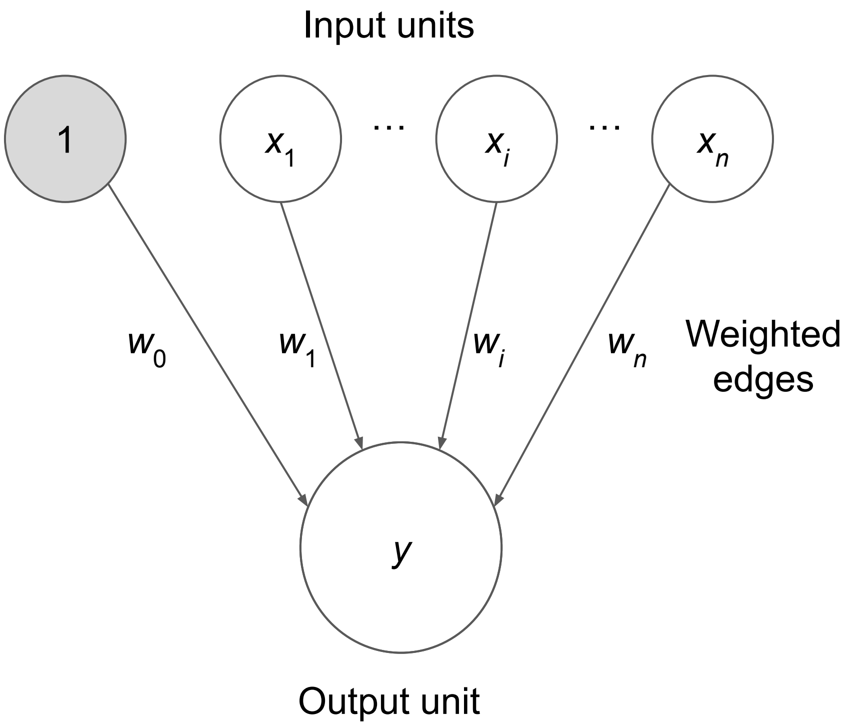
Training

t is 1 if the branch was taken and
-1 otherwise

θ is the threshold used to decide when
enough training has been done

if $\mathrm{sign}(y) \neq t$ or $|y| \leq \theta$ **then**

end if



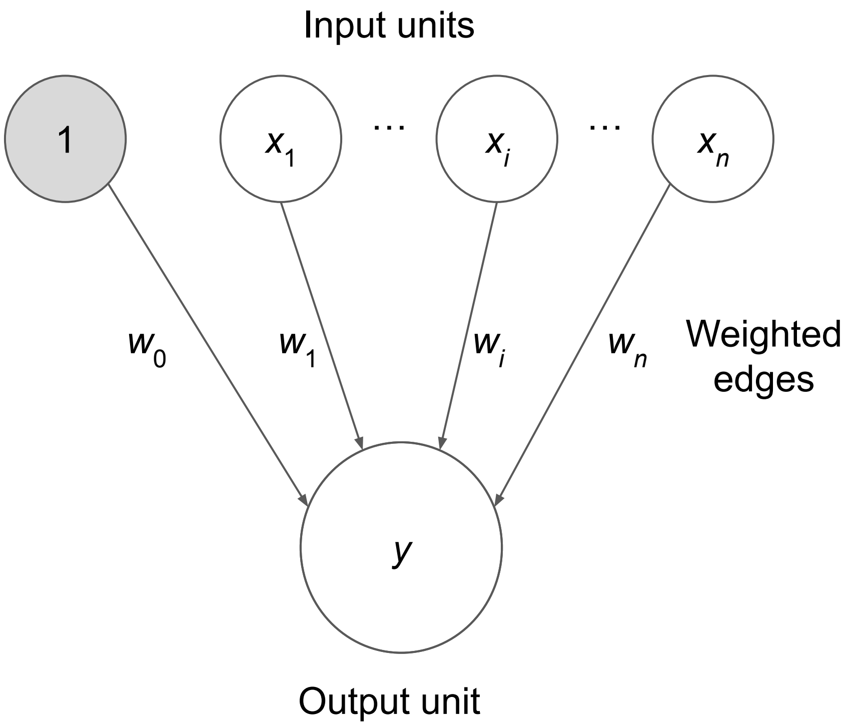
Jiménez 2001

Training

t is 1 if the branch was taken and -1 otherwise

Θ is the threshold used to decide when enough training has been done

$$
\begin{array}{l}
\textbf{if } \mathrm{sign}(y) \neq t \text{ or } |y| \leq \theta \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



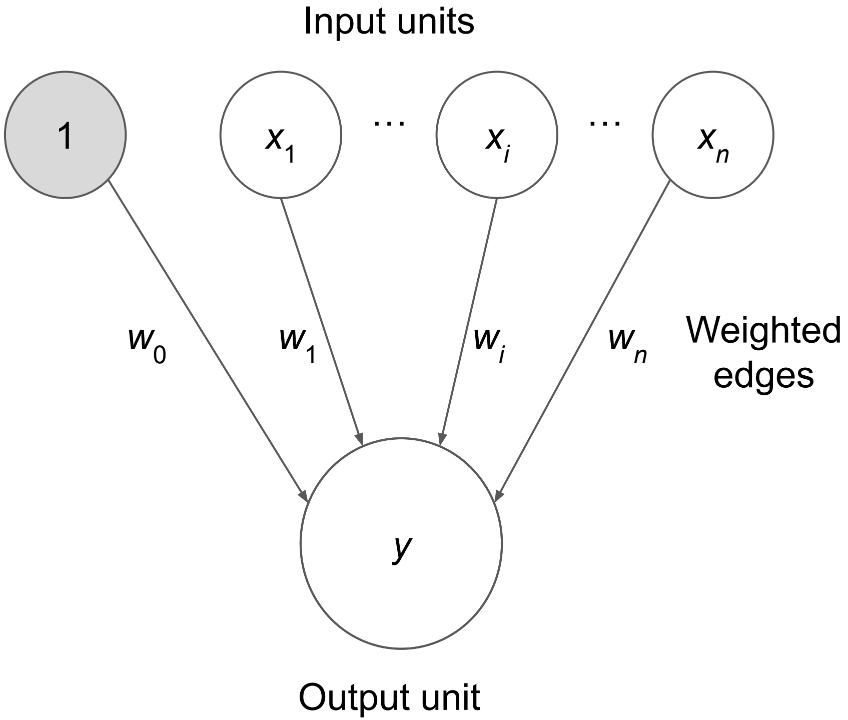
Jiménez 2001

Training

t is 1 if the branch was taken and -1 otherwise

Ɵ is the threshold used to decide when enough training has been done

$$
\begin{array}{l}
\textbf{if } \operatorname{sign}(y) \neq t \text{ or } |y| \leq \theta \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



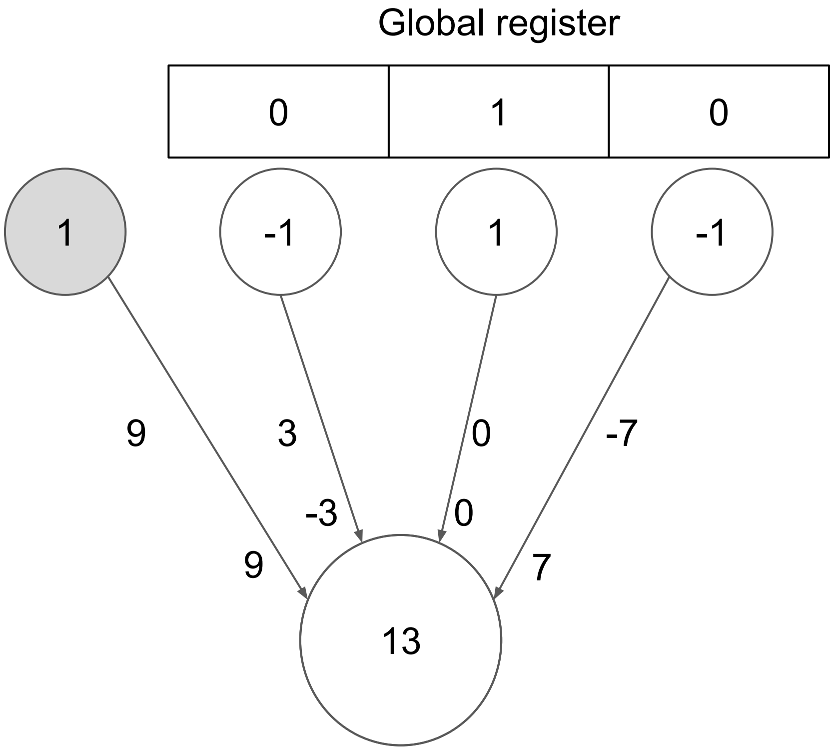
Jiménez 2001

Example

t is 1 if the branch was taken and
-1 otherwise

ϴ is the threshold used to decide when enough training has been done

$$
\begin{aligned}
&\textbf{if } \operatorname{sign}(y) \neq t \text{ or } |y| \leq \theta \textbf{ then}\\
&\quad \textbf{for } i := 0 \text{ to } n \textbf{ do}\\
&\qquad w_i := w_i + t x_i\\
&\quad \textbf{end for}\\
&\textbf{end if}
\end{aligned}
$$



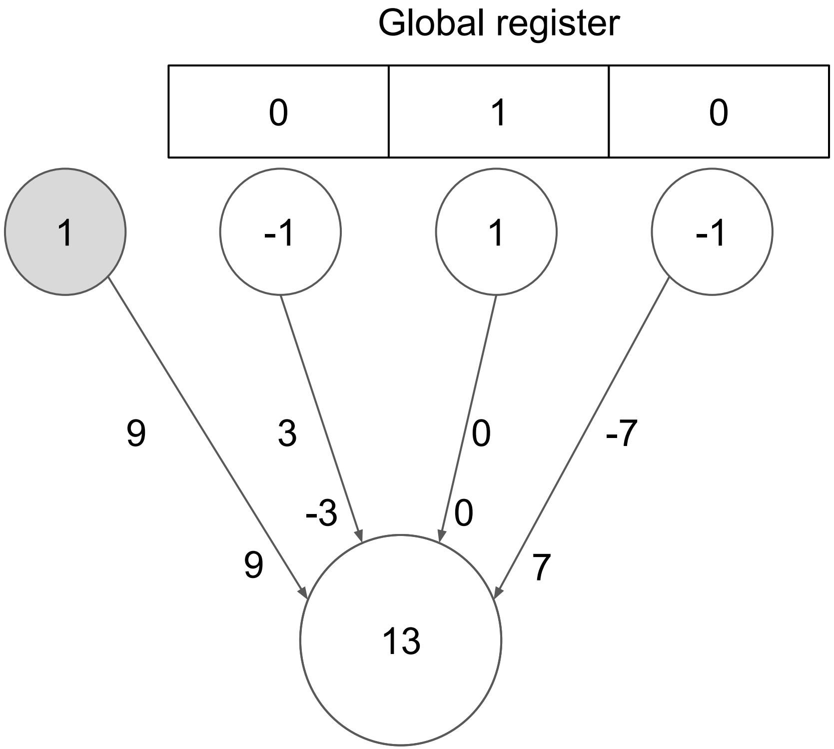
Jiménez 2001

Example

t is 1 if the branch was taken and
-1 otherwise

Ө is the threshold used to decide when
enough training has been done

$$
\begin{array}{l}
\textbf{if } \operatorname{sign}(y) \neq t \text{ or } |y| \leq \theta \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



Let *Ө* = 15 and *t* = -1 (the branch was not taken)

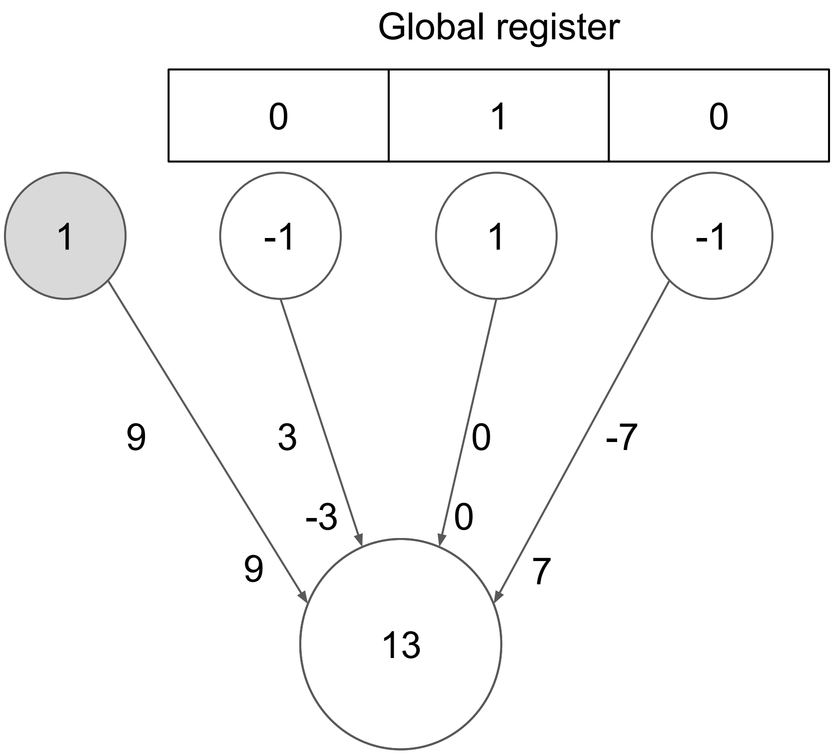
Jiménez 2001

Example

t is 1 if the branch was taken and -1 otherwise

Θ is the threshold used to decide when enough training has been done

$$\begin{array}{l}
\textbf{if } \underline{\text{sign}(y) \neq t} \text{ or } \underline{|y| \leq \theta} \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}$$



Let Θ = 15 and t = -1 (the branch was not taken)

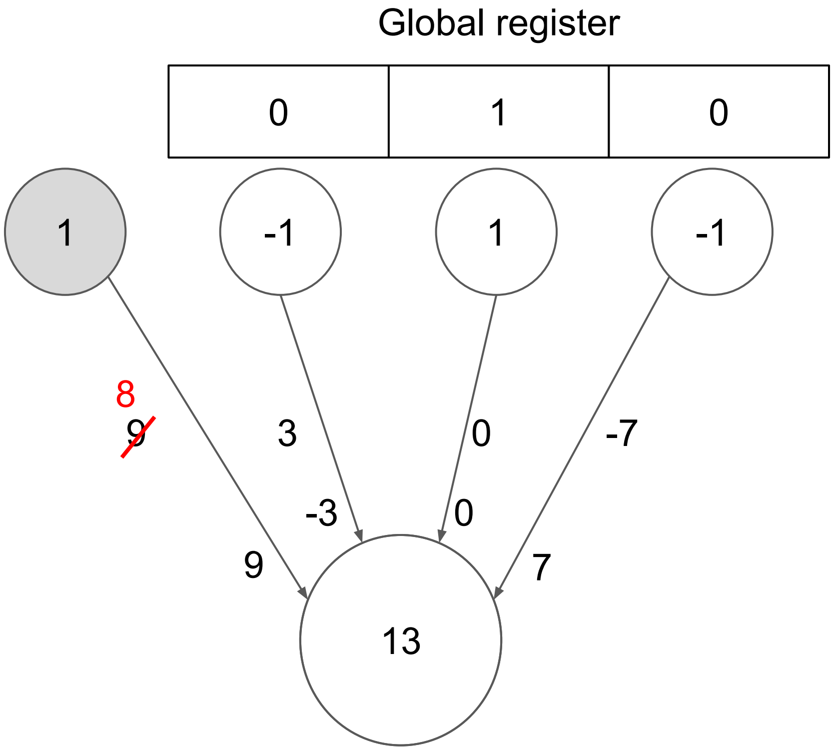
Jiménez 2001

Example

t is 1 if the branch was taken and
-1 otherwise

Θ is the threshold used to decide when
enough training has been done

$$
\begin{aligned}
&\textbf{if } \underline{\mathrm{sign}(y) \neq t} \text{ or } \underline{|y| \leq \theta} \textbf{ then} \\
&\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
&\qquad w_i := w_i + t x_i \\
&\quad \textbf{end for} \\
&\textbf{end if}
\end{aligned}
$$



Let Θ = 15 and t = -1 (the branch was not taken)

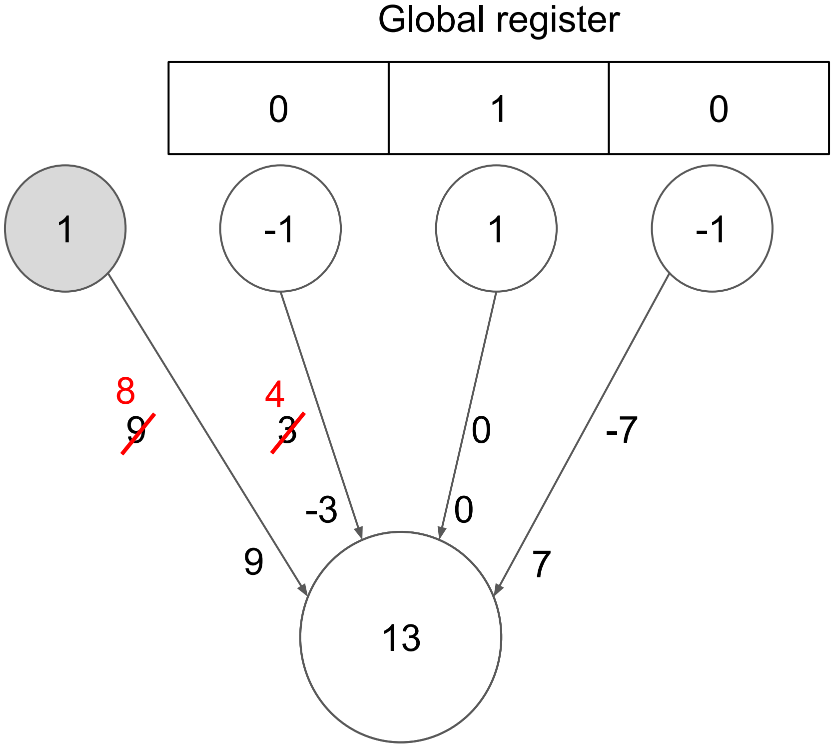
Jiménez 2001

Example

t is 1 if the branch was taken and
-1 otherwise

θ is the threshold used to decide when
enough training has been done

$$
\begin{aligned}
&\textbf{if } \underline{\mathrm{sign}(y) \neq t} \text{ or } \underline{|y| \leq \theta} \textbf{ then}\\
&\quad \textbf{for } i := 0 \text{ to } n \textbf{ do}\\
&\qquad w_i := w_i + t x_i\\
&\quad \textbf{end for}\\
&\textbf{end if}
\end{aligned}
$$



Let θ = 15 and t = -1 (the branch was not taken)

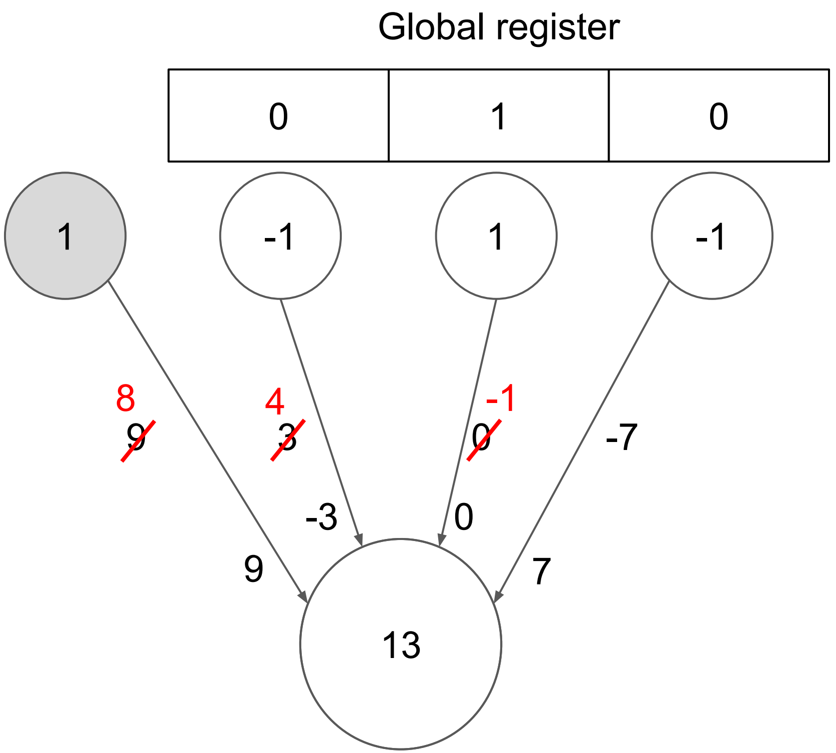
Jiménez 2001

Example

t is 1 if the branch was taken and
-1 otherwise

Θ is the threshold used to decide when
enough training has been done

if $\text{sign}(y) \neq t$ or $|y| \leq \theta$ **then**
  for $i := 0$ to n **do**
    $w_i := w_i + t x_i$
  end for
end if



Let Θ = 15 and t = -1 (the branch was not taken)

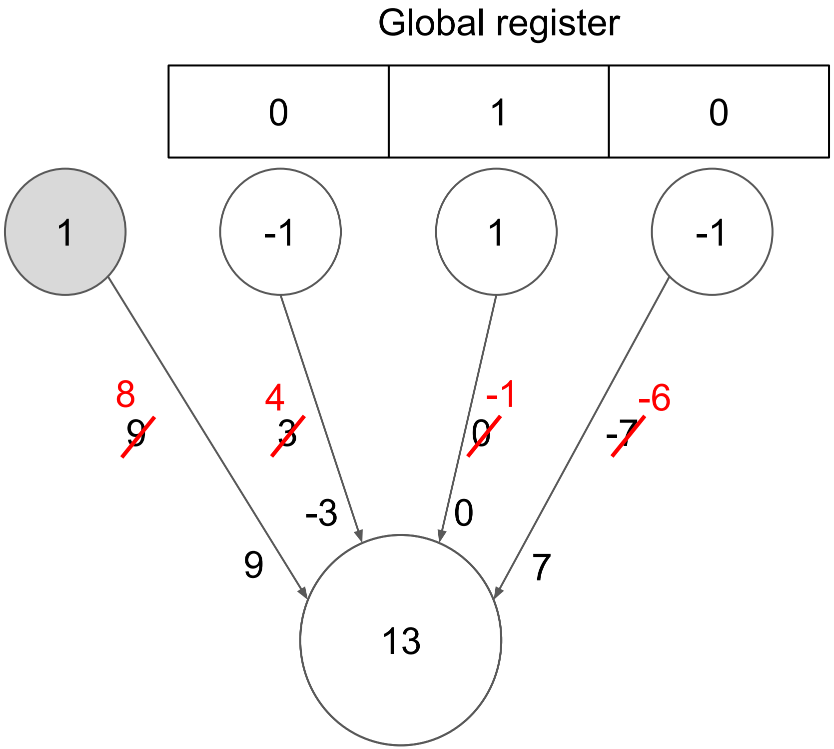
Jiménez 2001

Example

t is 1 if the branch was taken and
-1 otherwise

Ө is the threshold used to decide when
enough training has been done

$$
\begin{array}{l}
\textbf{if } \mathrm{sign}(y) \neq t \text{ or } |y| \leq \theta \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



Let Ө = 15 and *t* = -1 (the branch was not taken)

Jiménez 2001

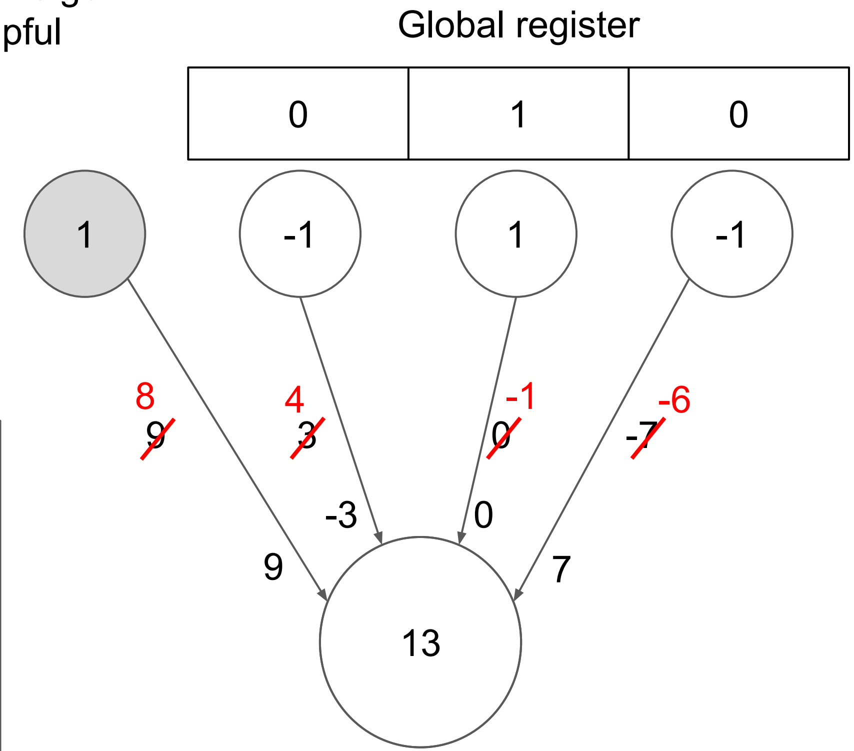
Example

When training, helpful weights get further from zero and unhelpful weights get closer to zero.

t is 1 if the branch was taken and -1 otherwise

Ө is the threshold used to decide when enough training has been done

$$
\begin{array}{l}
\textbf{if } \underline{\mathrm{sign}(y) \neq t} \text{ or } \underline{|y| \leq \theta} \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\qquad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



Let *Ө* = 15 and *t* = -1 (the branch was not taken)

Jiménez 2001

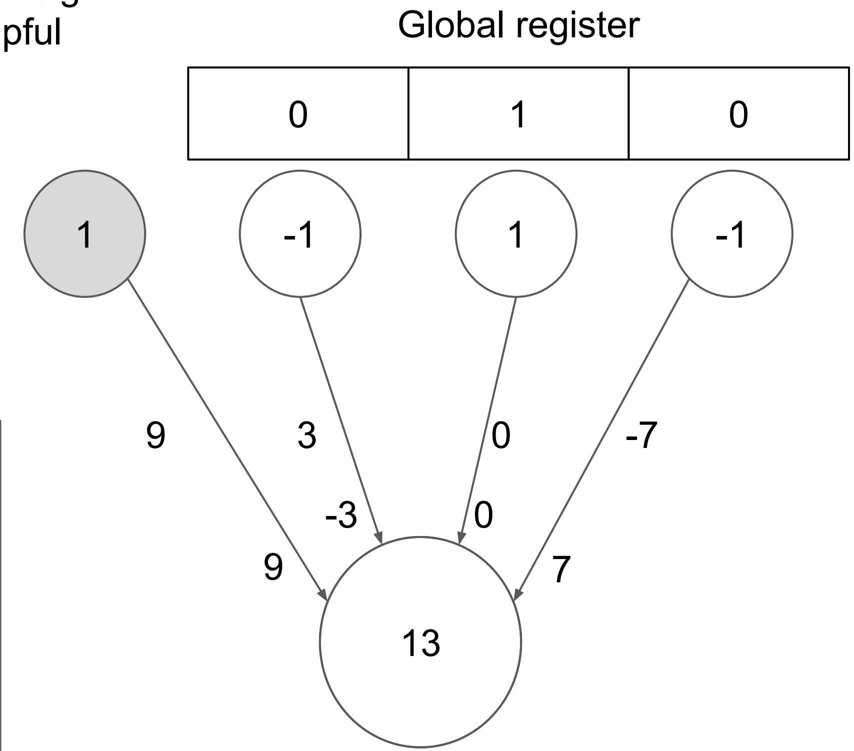
Example

When training, helpful weights get further from zero and unhelpful weights get closer to zero.

t is 1 if the branch was taken and -1 otherwise

ϴ is the threshold used to decide when enough training has been done

if $\mathrm{sign}(y) \neq t$ or $|y| \leq \theta$ **then**
  for $i := 0$ to n **do**
    $w_i := w_i + tx_i$
  end for
end if

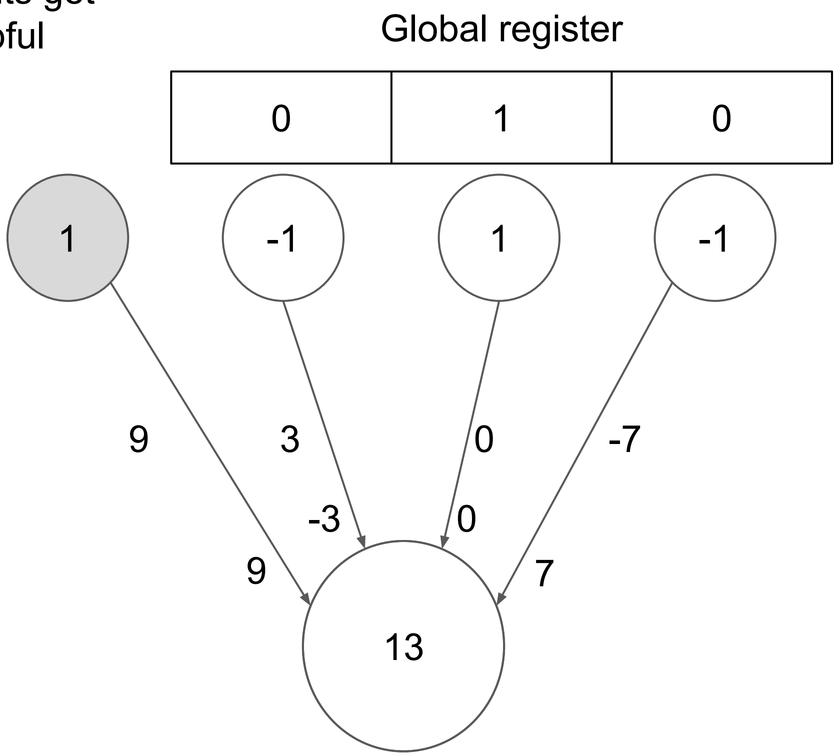


Let *ϴ* = 15 and *t* = 1 (the branch was taken)

Jiménez 2001

Example

When training, helpful weights get further from zero and unhelpful weights get closer to zero.



t is 1 if the branch was taken and -1 otherwise

ϴ is the threshold used to decide when enough training has been done

$$\textbf{if } \operatorname{sign}(y) \neq t \text{ or } \underline{|y| \leq \theta} \textbf{ then}$$
$$\quad \textbf{for } i := 0 \text{ to } n \textbf{ do}$$
$$\qquad w_i := w_i + t x_i$$
$$\quad \textbf{end for}$$
$$\textbf{end if}$$

Let *ϴ* = 15 and *t* = 1 (the branch was taken)

Jiménez 2001

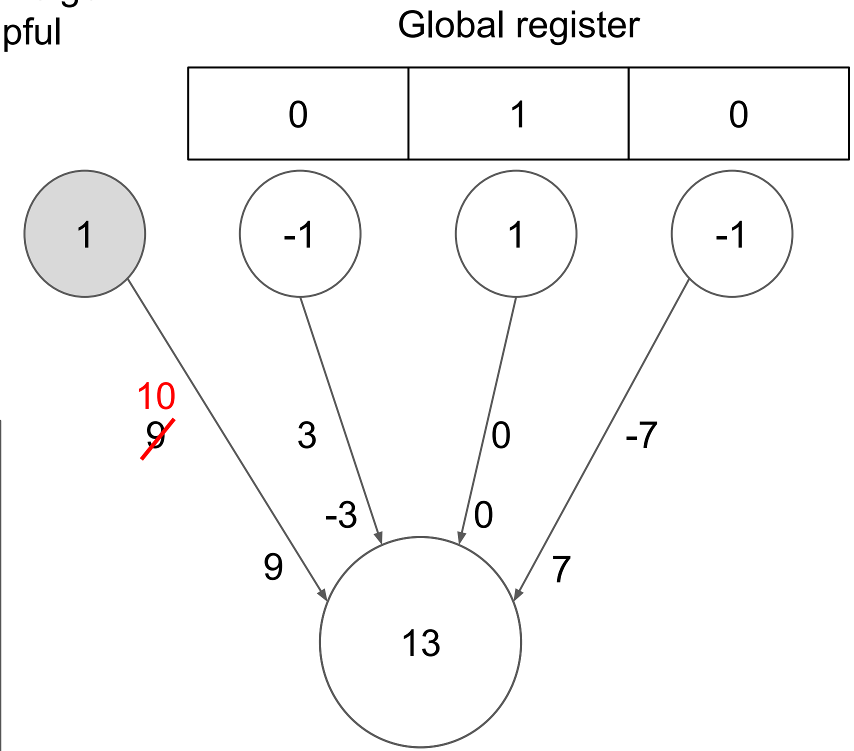
Example

When training, helpful weights get further from zero and unhelpful weights get closer to zero.

t is 1 if the branch was taken and -1 otherwise

Θ is the threshold used to decide when enough training has been done

$$
\begin{array}{l}
\textbf{if } \operatorname{sign}(y) \neq t \text{ or } \underline{|y| \leq \theta} \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



Let Θ = 15 and t = 1 (the branch was taken)

Jiménez 2001

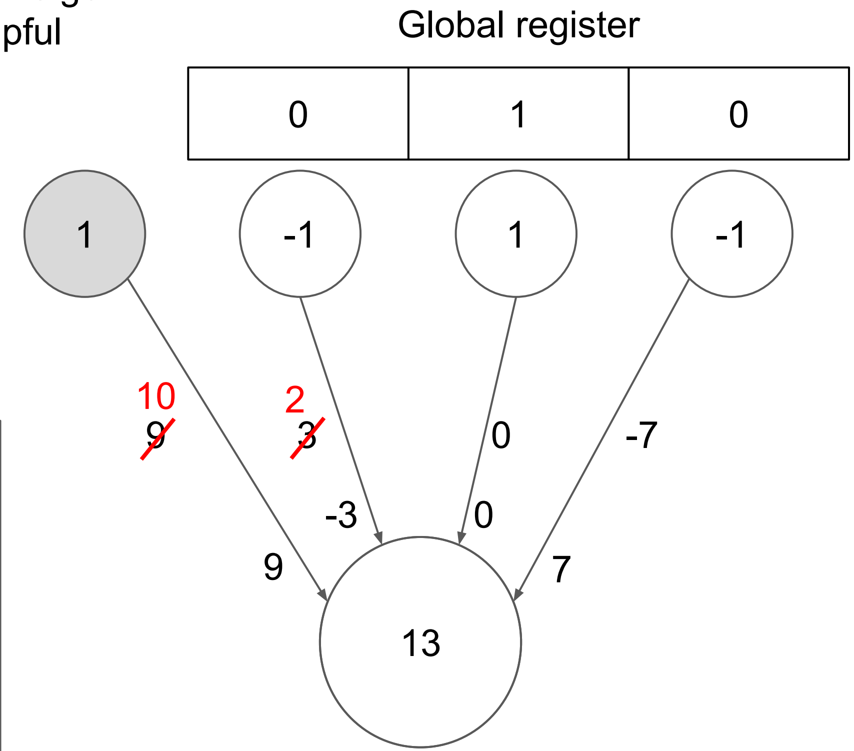
Example

When training, helpful weights get further from zero and unhelpful weights get closer to zero.

t is 1 if the branch was taken and -1 otherwise

Θ is the threshold used to decide when enough training has been done

$$
\begin{array}{l}
\textbf{if } \mathrm{sign}(y) \neq t \text{ or } \underline{|y| \leq \theta} \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



Let *Θ* = 15 and *t* = 1 (the branch was taken)

Jiménez 2001

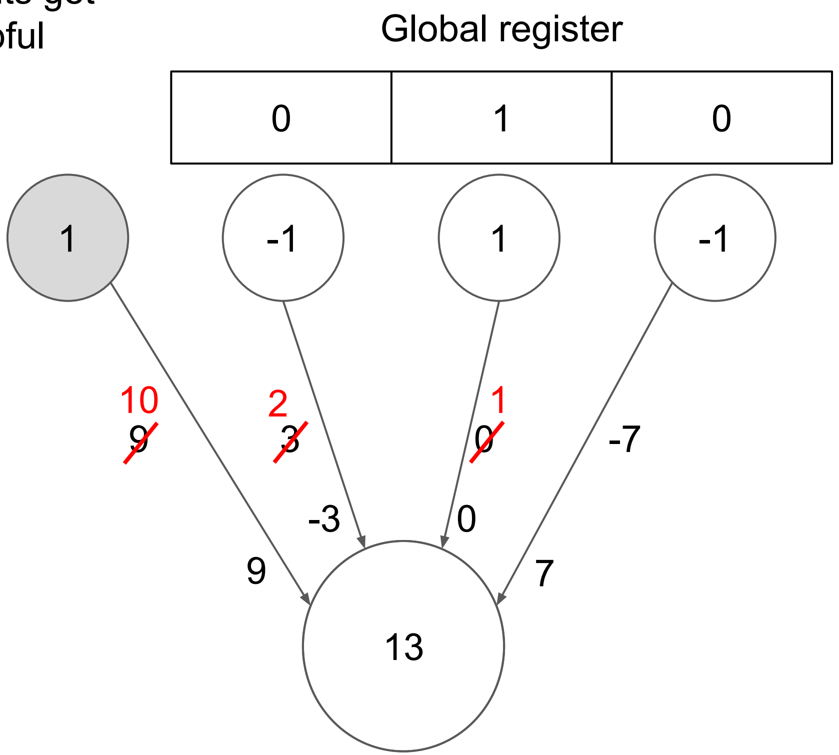
Example

When training, helpful weights get further from zero and unhelpful weights get closer to zero.

t is 1 if the branch was taken and -1 otherwise

Ɵ is the threshold used to decide when enough training has been done

$$
\begin{array}{l}
\textbf{if } \operatorname{sign}(y) \neq t \text{ or } \underline{|y| \leq \theta} \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



Let *Ɵ* = 15 and *t* = 1 (the branch was taken)

Jiménez 2001

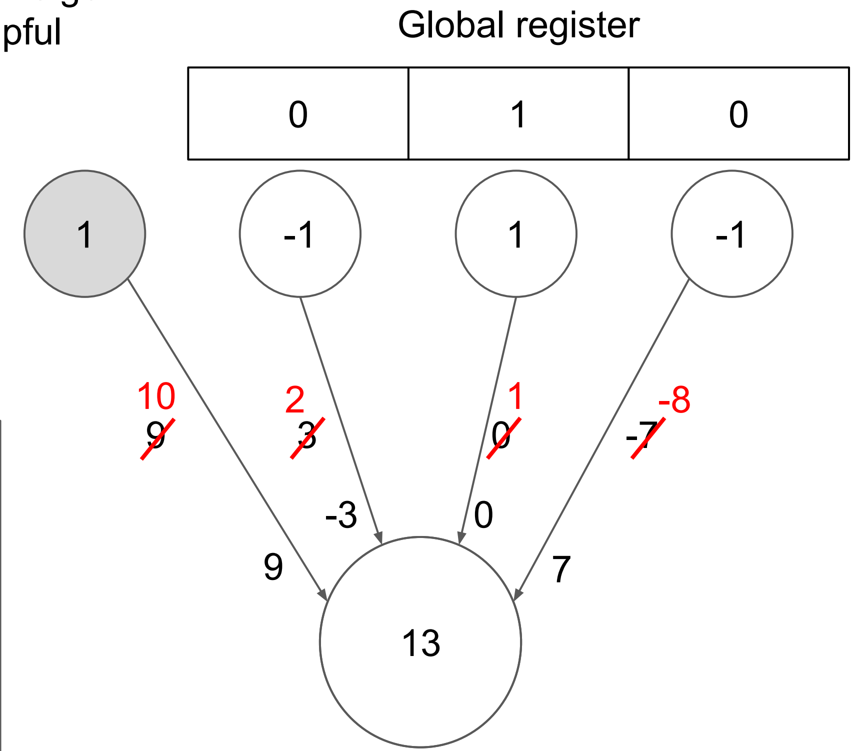
Example

When training, helpful weights get further from zero and unhelpful weights get closer to zero.

t is 1 if the branch was taken and -1 otherwise

ϴ is the threshold used to decide when enough training has been done

$$
\begin{array}{l}
\textbf{if } \operatorname{sign}(y) \neq t \text{ or } \underline{|y| \leq \theta} \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



Let ϴ = 15 and *t* = 1 (the branch was taken)

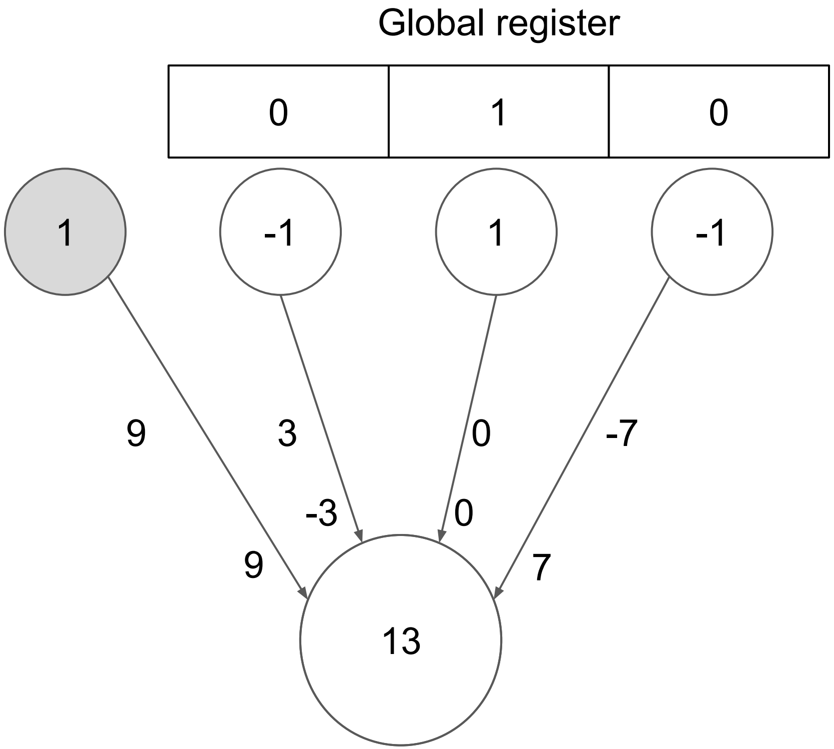
Jiménez 2001

Example

t is 1 if the branch was taken and
-1 otherwise

ϴ is the threshold used to decide when enough training has been done

$$
\begin{aligned}
&\textbf{if } \mathrm{sign}(y) \neq t \text{ or } |y| \leq \theta \textbf{ then}\\
&\quad \textbf{for } i := 0 \text{ to } n \textbf{ do}\\
&\qquad w_i := w_i + t x_i\\
&\quad \textbf{end for}\\
&\textbf{end if}
\end{aligned}
$$



Let *ϴ* = 10 and *t* = 1 (the branch was taken)

Jiménez 2001

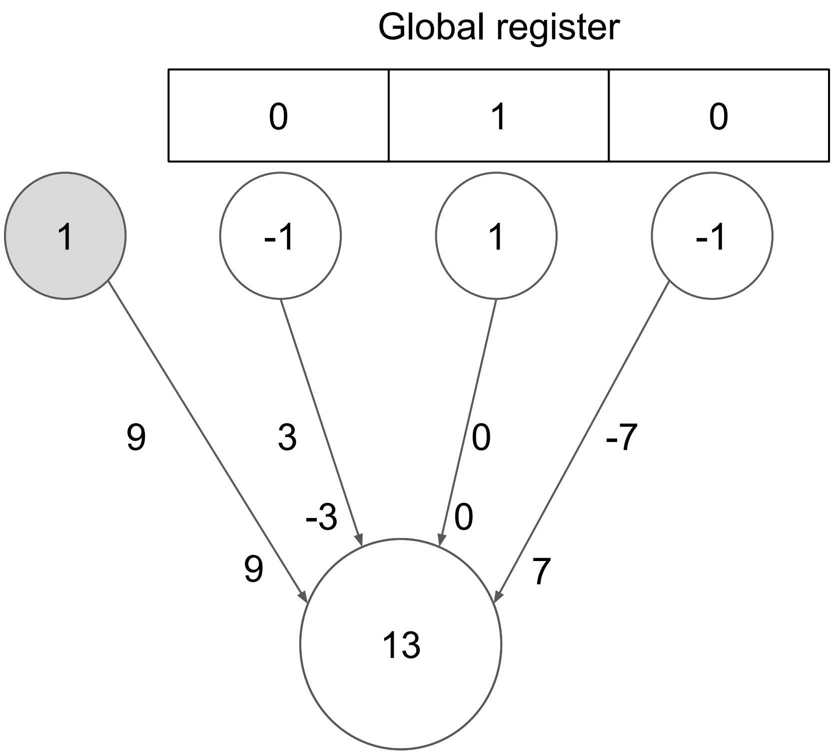
Example

When above the threshold, only train the weights if the prediction was incorrect

t is 1 if the branch was taken and -1 otherwise

Θ is the threshold used to decide when enough training has been done

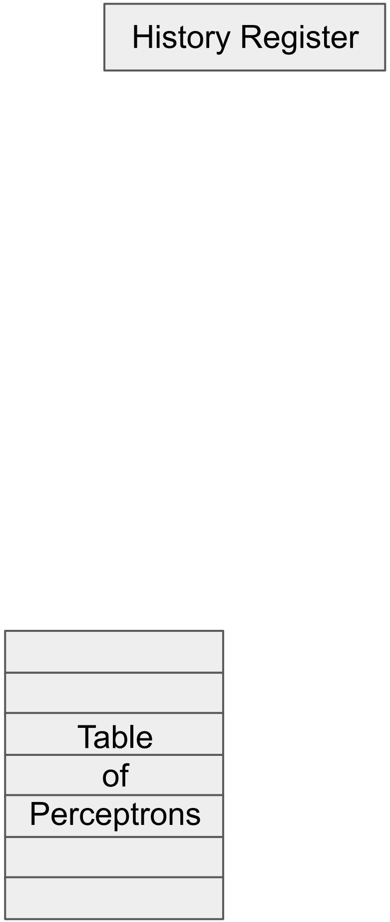
$$
\begin{array}{l}
\textbf{if } \operatorname{sign}(y) \neq t \text{ or } |y| \leq \theta \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$



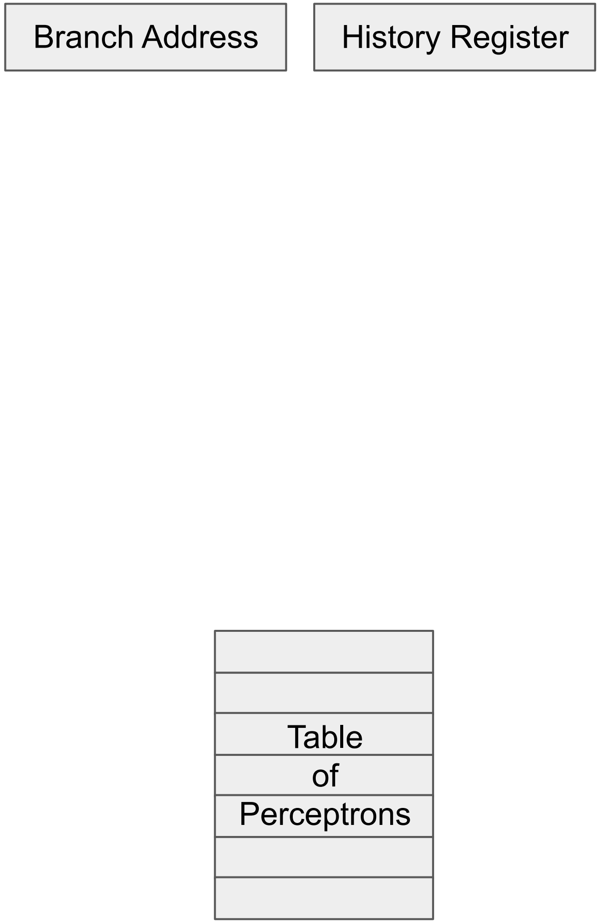
Let Θ = 10 and t = 1 (the branch was taken)

Jiménez 2001

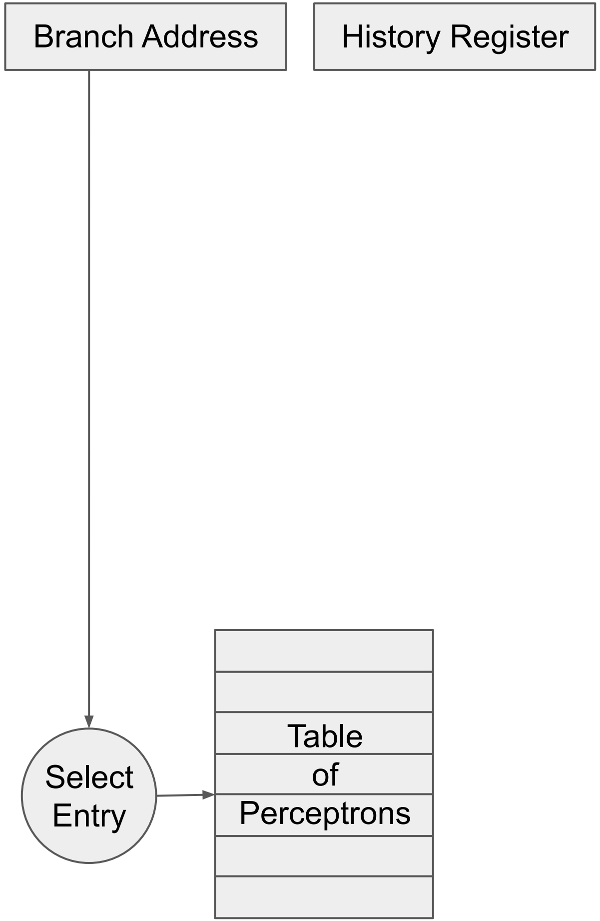
Jiménez 2001



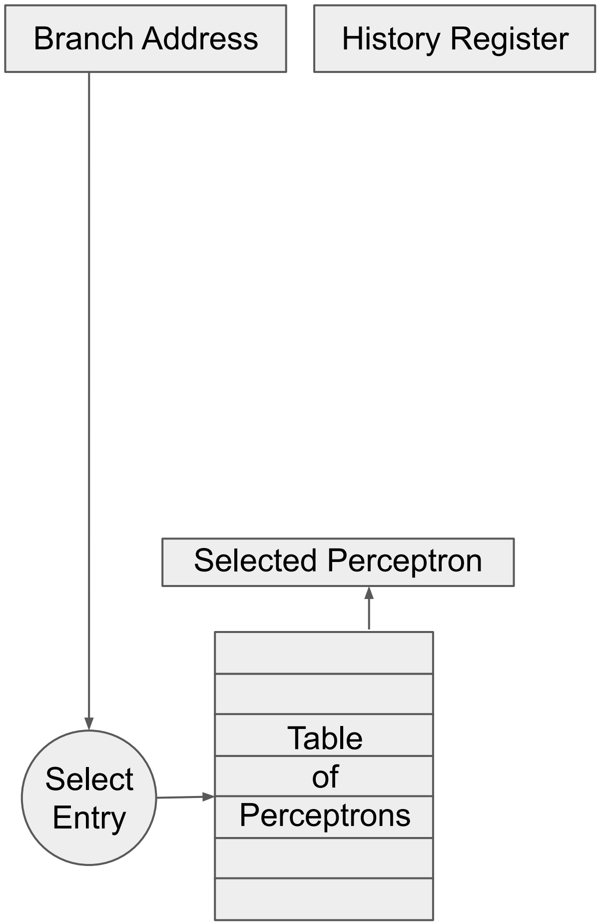
Jiménez 2001



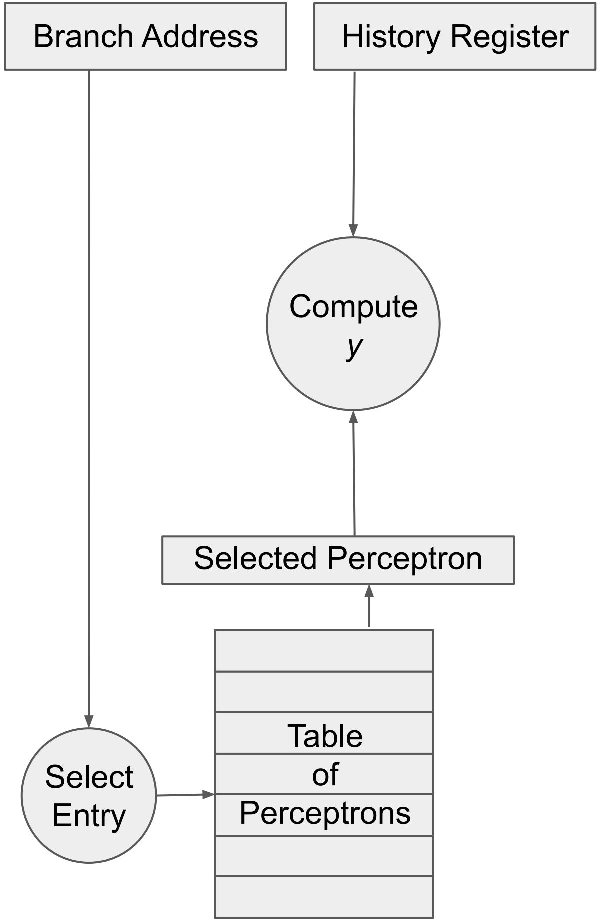
Jiménez 2001



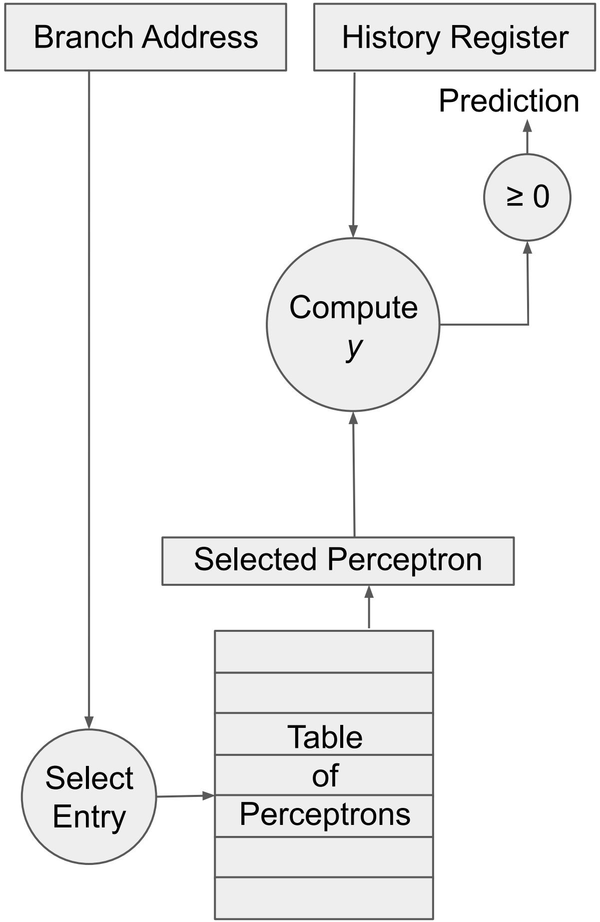
Jiménez 2001



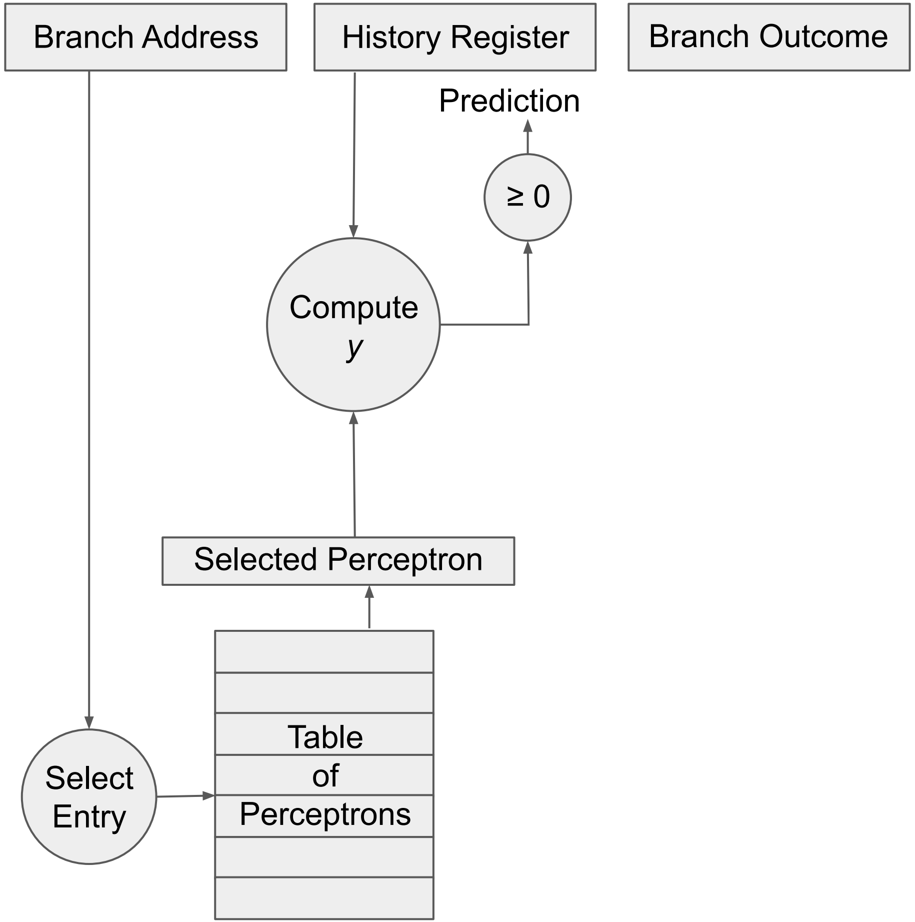
Jiménez 2001



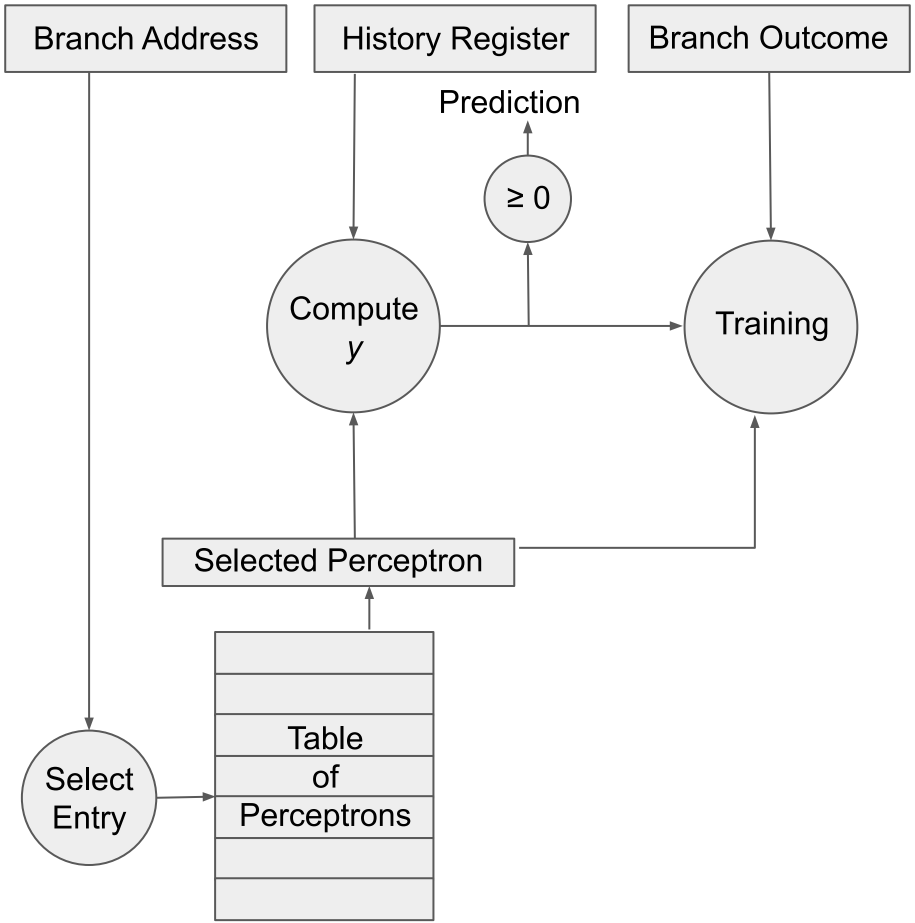
Jiménez 2001



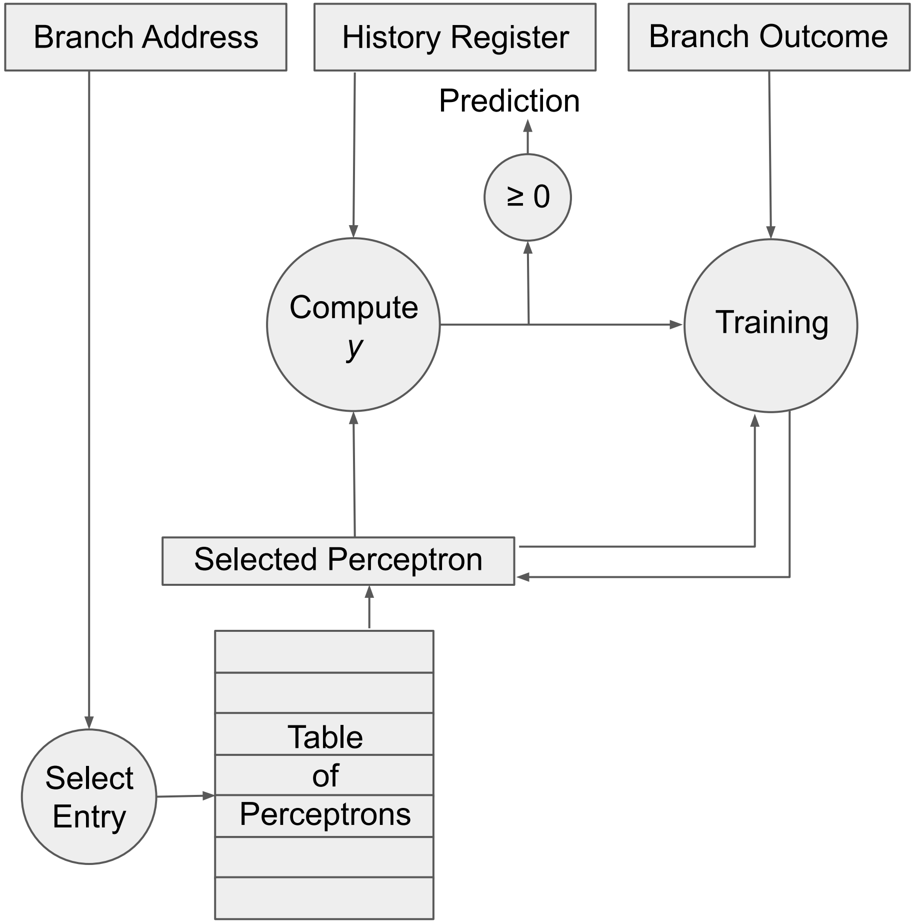
Jiménez 2001



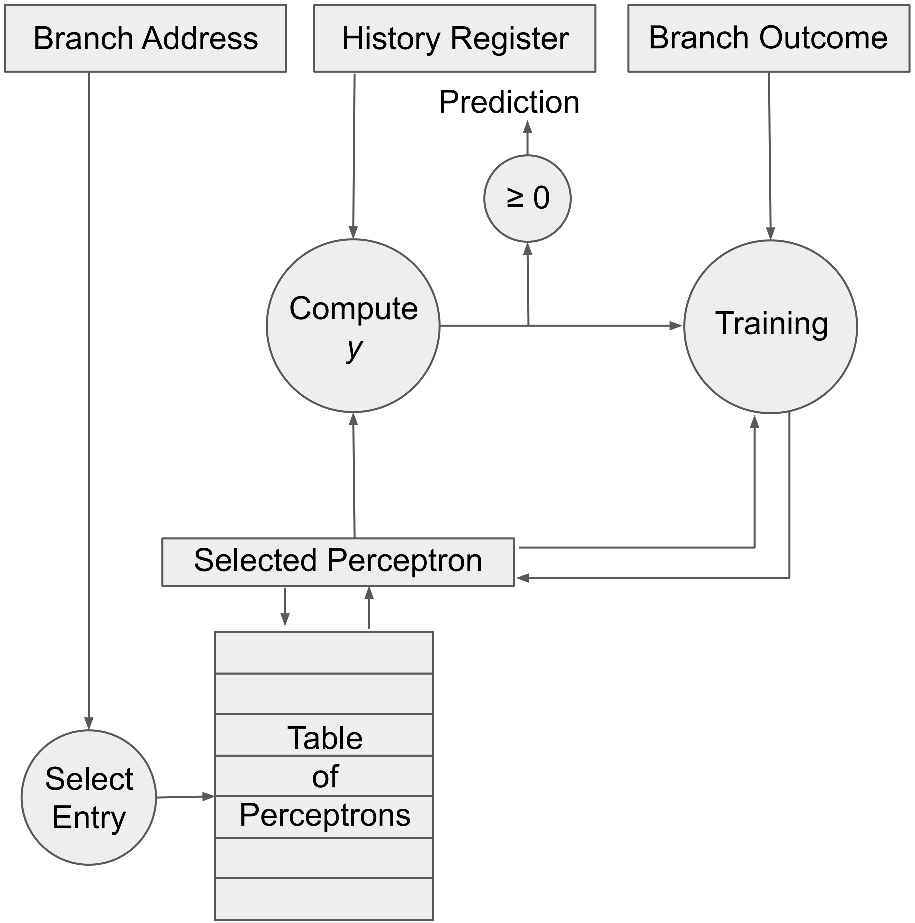
Jiménez 2001



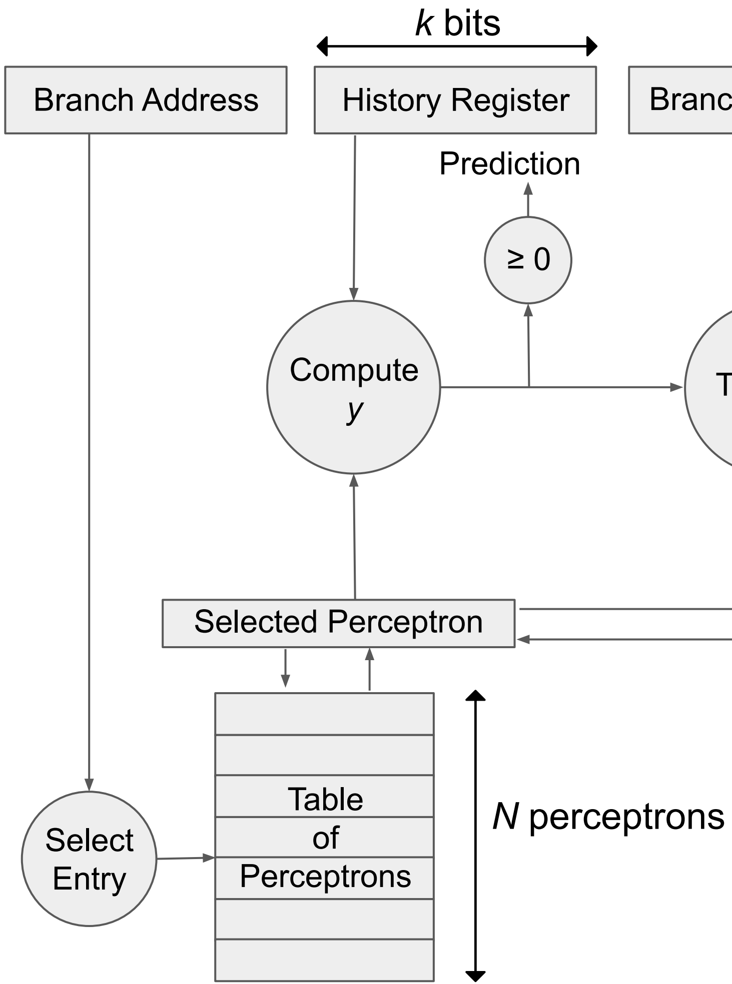
Jiménez 2001



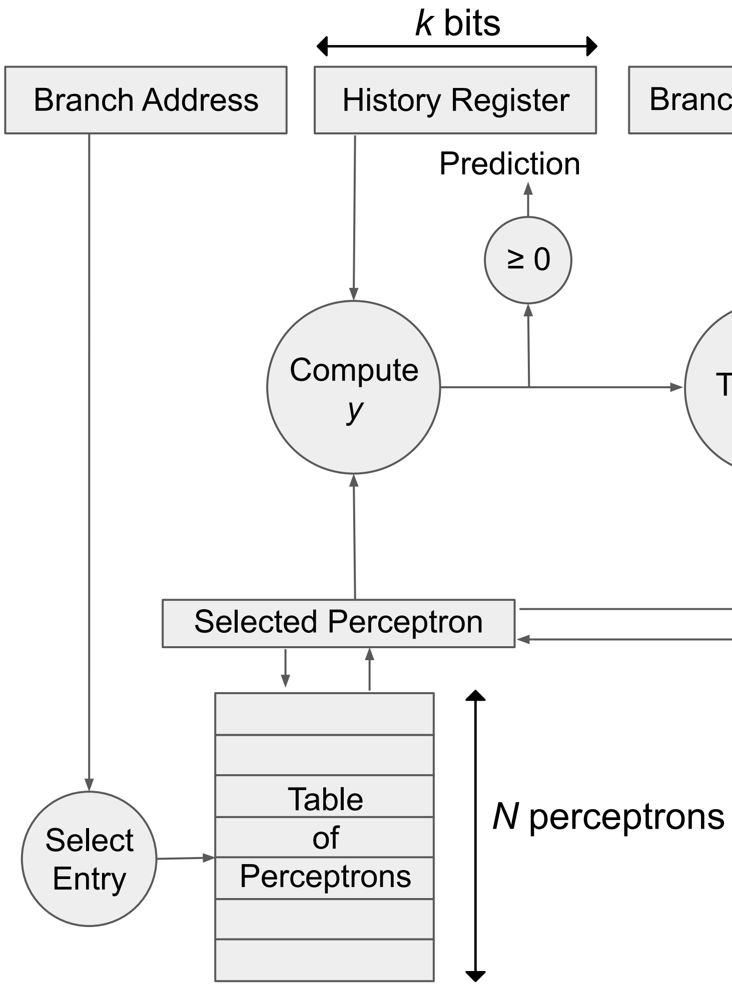
Jiménez 2001



Jiménez 2001

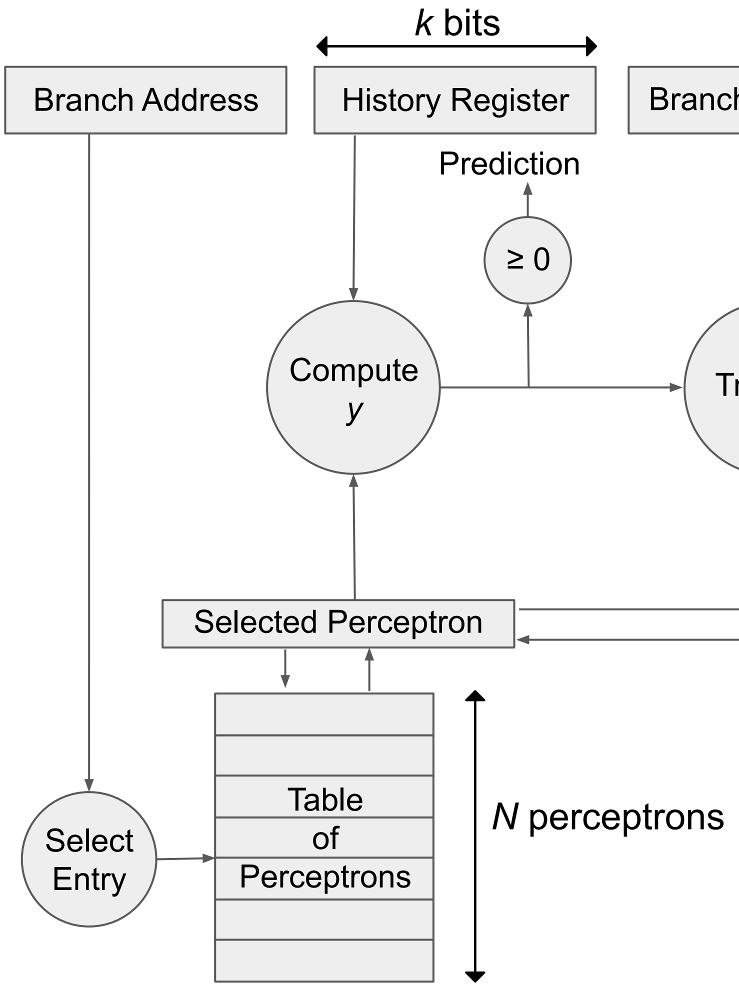


Jiménez 2001



Let m = # of bits for each weight in a perceptron

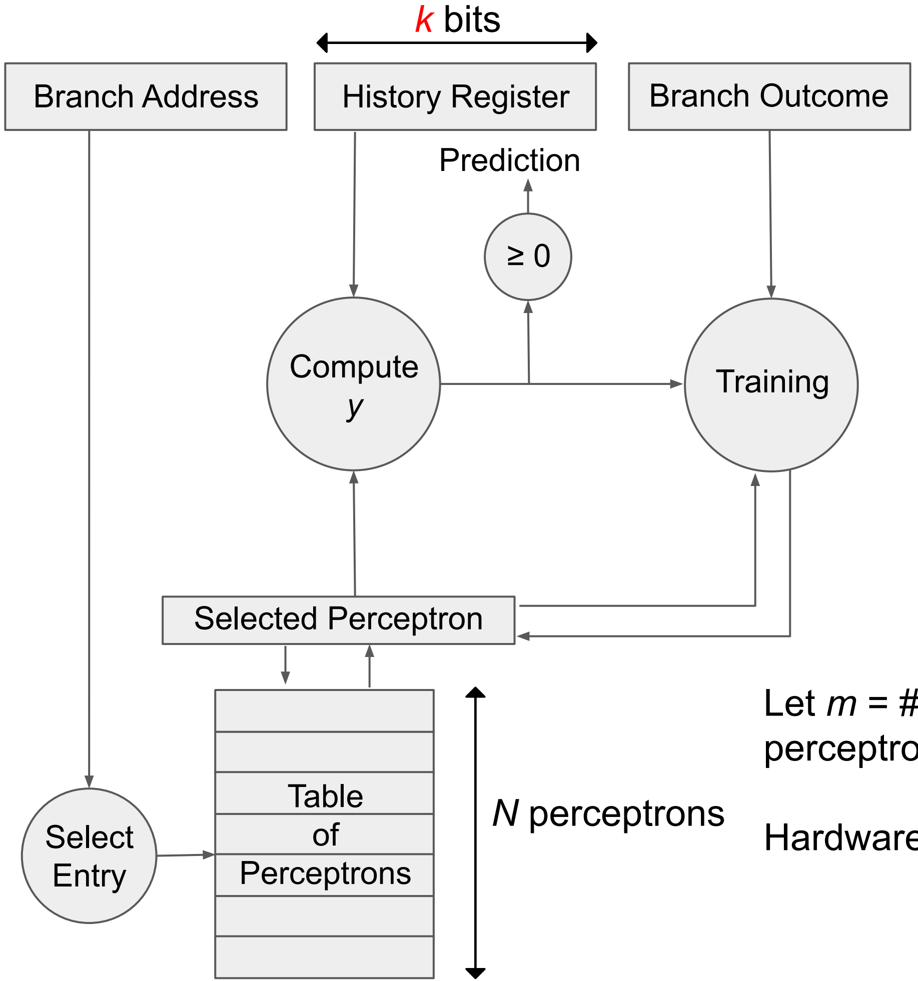
Jiménez 2001



Let m = # of bits for each weight in a perceptron

Hardware resources = $N * (k + 1) * m$

Jiménez 2001



Hardware resources scale linearly with the history length!

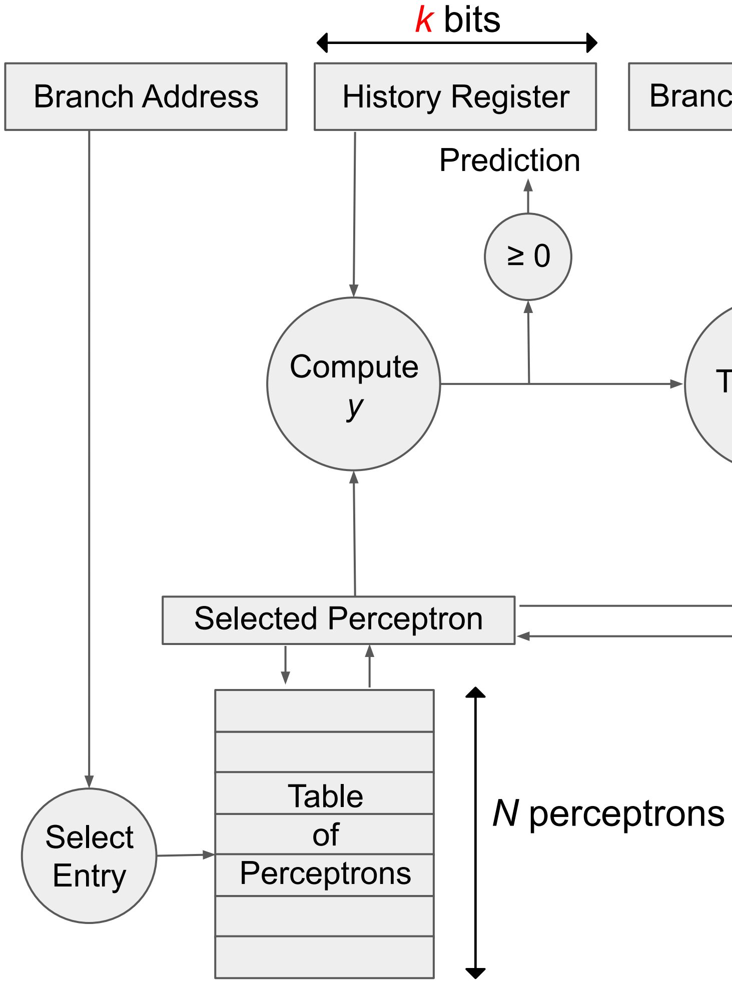
Let m = # of bits for each weight in a perceptron

Hardware resources = $N * (k + 1) * m$

Jiménez 2001

Hardware resources scale linearly with the history length!

Can consider much longer history lengths than a two-level predictor



Let m = # of bits for each weight in a perceptron

Hardware resources = $N * (k + 1) * m$

Jiménez 2001

By leveraging longer history lengths, neural branch predictors achieve higher prediction accuracy

Jiménez 2001

Problem

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

Computing a dot product each time you make a prediction is too slow!

Jiménez 2002

Solutions

Jiménez 2002

Solutions

- Avoid doing multiplication since each x_i is 1 or -1
  - Only need to add/subtract weights to compute y

Jiménez 2002

Solutions

- Avoid doing multiplication since each x_i is 1 or -1
  - Only need to add/subtract weights to compute y
- Only need the most significant bit of y to make a prediction
  - Make prediction before all of y is computed

Jiménez 2002

Solutions

- Avoid doing multiplication since each x_i is 1 or -1
  - Only need to add/subtract weights to compute y
- Only need the most significant bit of y to make a prediction
  - Make prediction before all of y is computed
- Override predictors
  - Have 2 predictors, a faster less accurate predictor and a slower more accurate predictor
    - If slower predictor disagrees with faster predictor, override prediction

Jiménez 2002

Solutions

Fast Path-Based Neural Branch Prediction

Daniel A. Jiménez
Department of Computer Science
Rutgers University, Piscataway, NJ 08854

Jiménez 2003

Solutions

Fast Path-Based Neural Branch Prediction

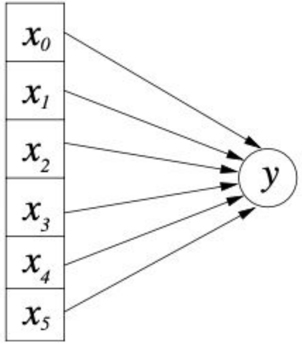
Daniel A. Jiménez
Department of Computer Science
Rutgers University, Piscataway, NJ 08854

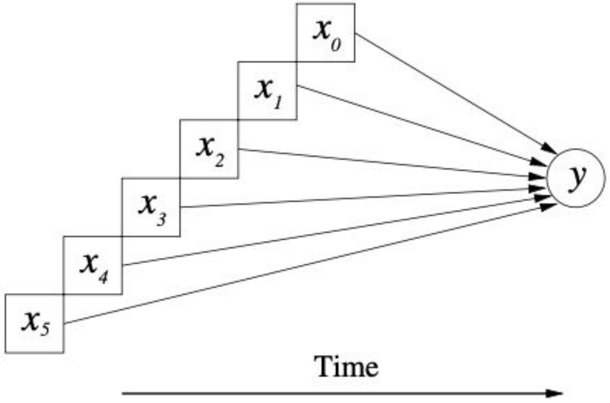
Staggers computation in time

Solutions

Fast Path-Based Neural Branch Prediction

Daniel A. Jiménez
Department of Computer Science
Rutgers University, Piscataway, NJ 08854





Staggers computation in time

Predicts a branch using neurons selected dynamically along the path to the branch, rather than selecting the neurons all at once based solely on the branch address

Jiménez 2003

Solutions

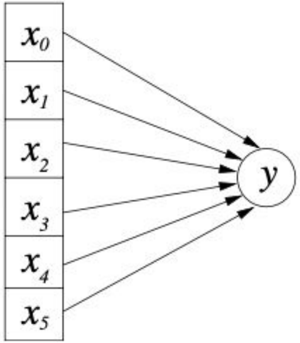
Fast Path-Based Neural Branch Prediction

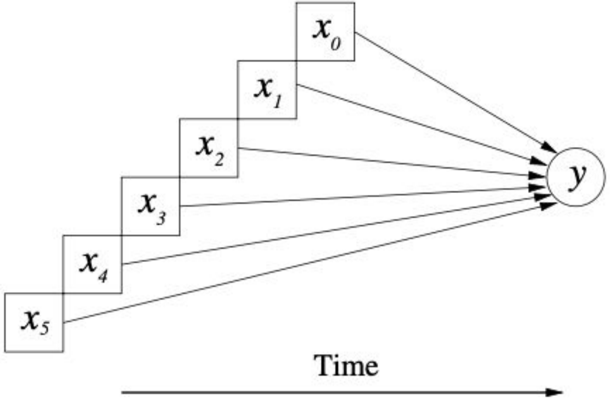
Daniel A. Jiménez
Department of Computer Science
Rutgers University, Piscataway, NJ 08854

Staggers computation in time

Predicts a branch using neurons selected dynamically along the path to the branch, rather than selecting the neurons all at once based solely on the branch address

Faster and more accurate





Jiménez 2003

Linear Separability

Jiménez 2002

Linear Separability

Perceptrons can only learn functions that are linearly separable

Jiménez 2002

Linear Separability

Perceptrons can only learn functions that are linearly separable

- A boolean function is linearly separable if and only if there exists a hyperplane that separates all of the true instances from all of the false instances

Jiménez 2002

Linear Separability

Perceptrons can only learn functions that are linearly separable

- A boolean function is linearly separable if and only if there exists a hyperplane that separates all of the true instances from all of the false instances

AND	0	1
0	0	0
1	0	1

Jiménez 2002

Linear Separability

Perceptrons can only learn functions that are linearly separable

- A boolean function is linearly separable if and only if there exists a hyperplane that separates all of the true instances from all of the false instances

AND	0	1
0	0	0
1	0	1

Jiménez 2002

Linear Separability

Perceptrons can only learn functions that are linearly separable

- A boolean function is linearly separable if and only if there exists a hyperplane that separates all of the true instances from all of the false instances

AND	0	1
0	0	0
1	0	1

XOR	0	1
0	0	1
1	1	0

Jiménez 2002

Linear Separability

Perceptrons can only learn functions that are linearly separable

- A boolean function is linearly separable if and only if there exists a hyperplane that separates all of the true instances from all of the false instances
- Most branch correlations are linearly separable

AND	0	1
0	0	0
1	0	1

XOR	0	1
0	0	1
1	1	0

Jiménez 2002

Linear Separability

Perceptrons can only learn functions that are linearly separable

- A boolean function is linearly separable if and only if there exists a hyperplane that separates all of the true instances from all of the false instances
- Most branch correlations are linearly separable
- Neural predictors work well in hybrid predictors

AND	0	1
0	0	0
1	0	1

XOR	0	1
0	0	1
1	1	0

Jiménez 2002

Have architectures implemented neural predictors?

Yes

https://www.computer.org/profiles/daniel-jimenez

Dynamic Branch Prediction with Perceptrons

Daniel A. Jiménez   Calvin Lin
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
{djimenez,lin}@cs.utexas.edu

Jiménez 2001

Dynamic Branch Prediction with Perceptrons

Daniel A. Jiménez   Calvin Lin
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
{djimenez,lin}@cs.utexas.edu

2019 HPCA Test of Time Award

The HPCA Test of Time (ToT) award recognizes the most influential papers published in prior sessions of the International Symposium of High Performance Computer Architecture (HPCA) each of whom have had significant impact in the field.

Jiménez 2001

Dynamic Branch Prediction with Perceptrons

Daniel A. Jiménez Calvin Lin
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
{djimenez,lin}@cs.utexas.edu

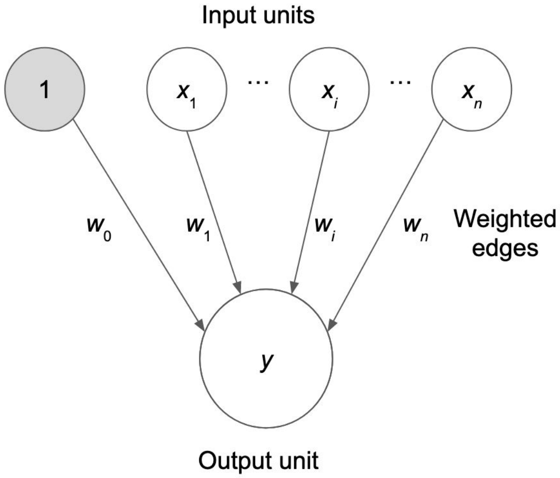
2019 HPCA Test of Time Award

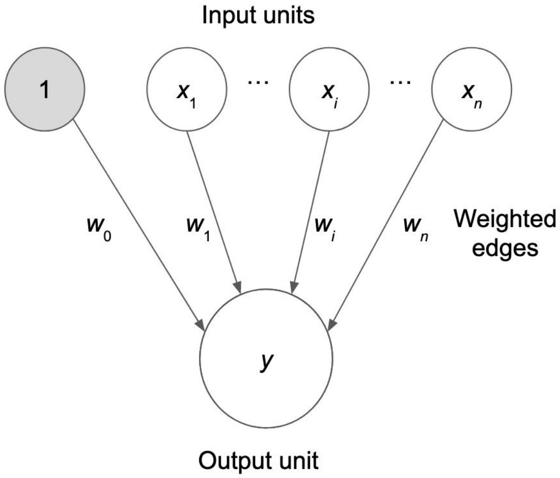
The HPCA Test of Time (ToT) award recognizes the most influential papers published in prior sessions of the International Symposium of High Performance Computer Architecture (HPCA) each of whom have had significant impact in the field.

2021 B. Ramakrishna Rau Award

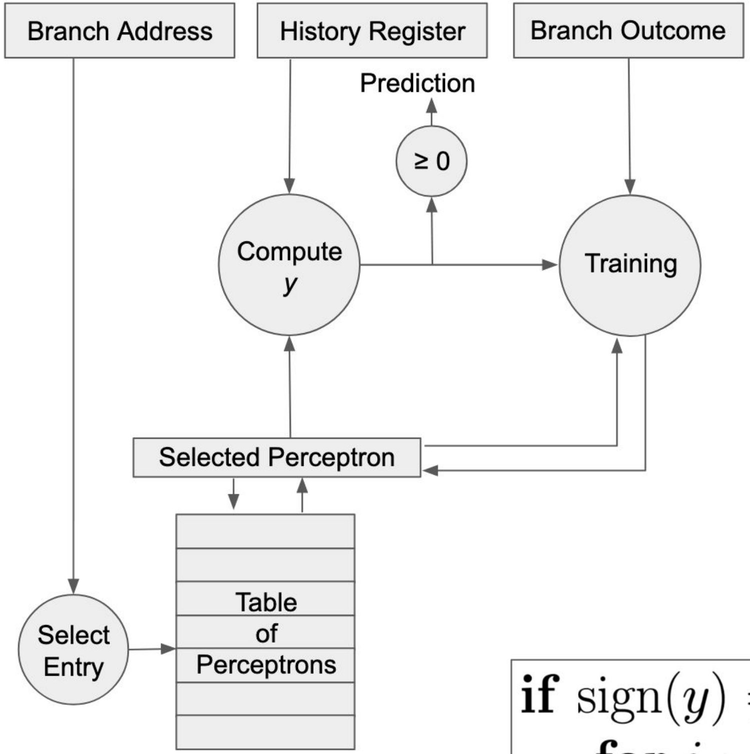
The B. Ramakrishna Rau award will be presented ***"in recognition of substantial contributions in the field of computer microarchitecture and compiler code generation."***

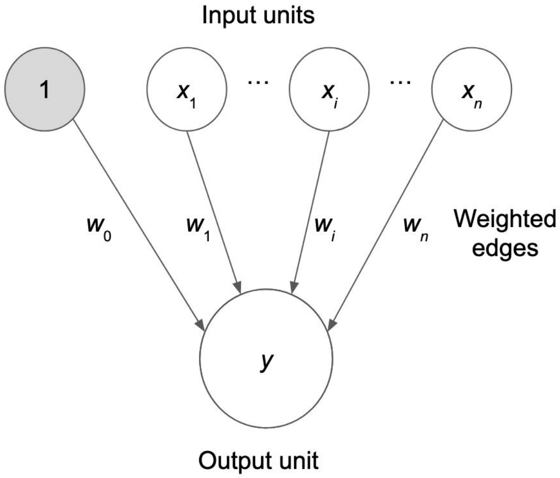
Jiménez 2001


Input units
1
x_1
…
x_i
…
x_n
w_0
w_1
w_i
w_n
Weighted edges
y
Output unit

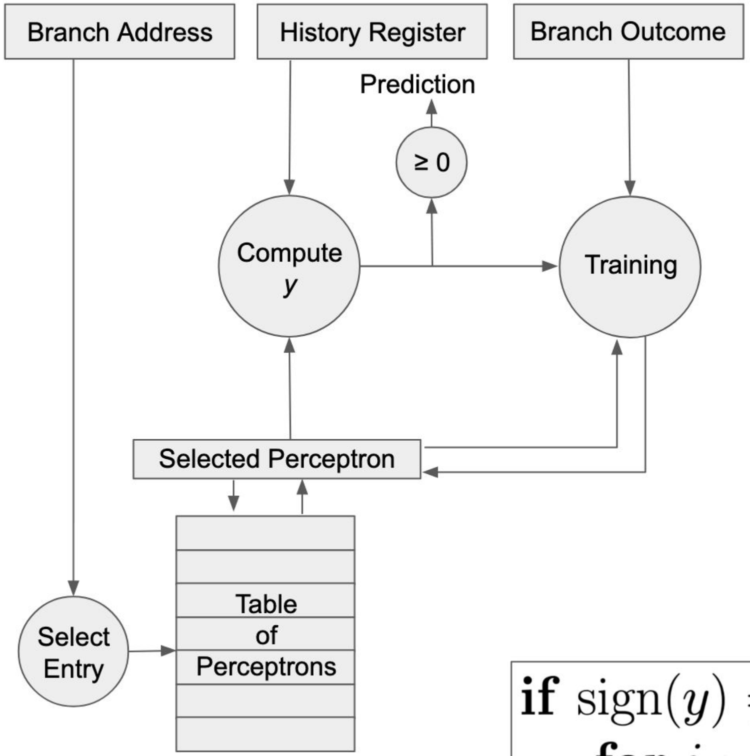


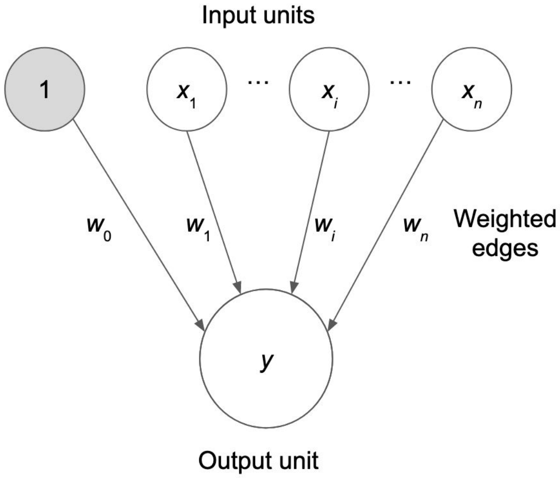
if $\text{sign}(y) \neq t$ or $|y| \leq \theta$ **then**
  for $i := 0$ to n **do**
    $w_i := w_i + t x_i$
  end for
end if





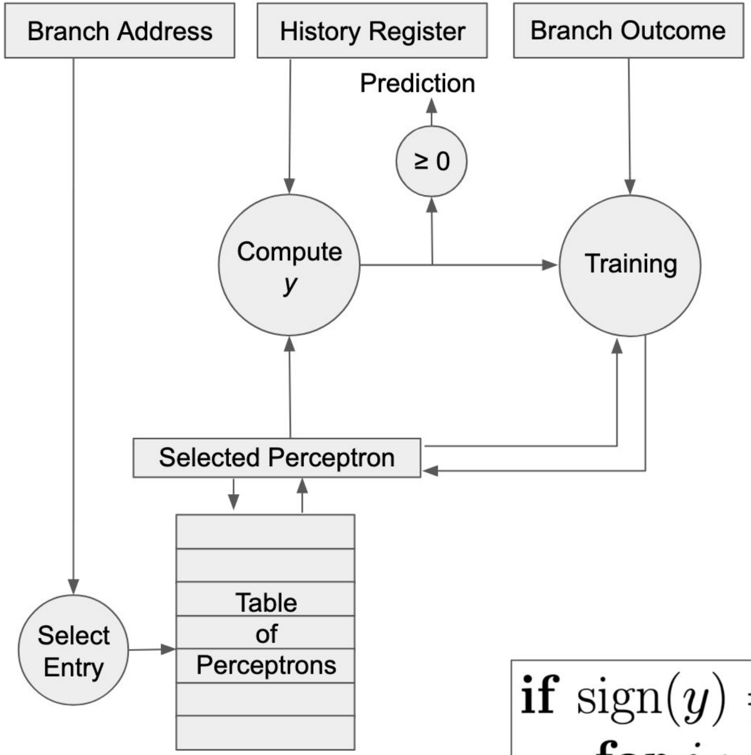
$$
\begin{array}{l}
\textbf{if } \operatorname{sign}(y) \neq t \text{ or } |y| \leq \theta \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\quad\quad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$

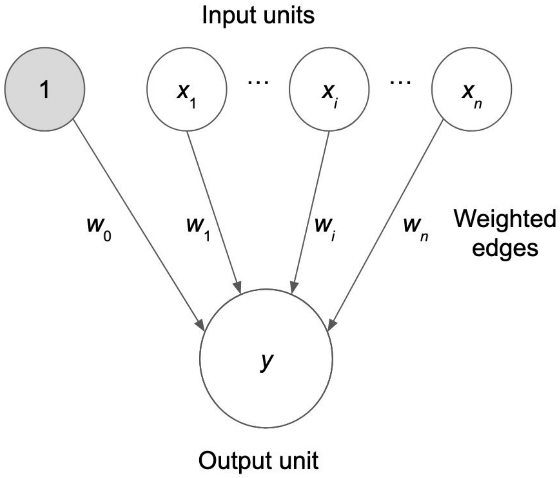




exponential vs. linear scaling of hardware resources

$$
\begin{array}{l}
\textbf{if } \operatorname{sign}(y) \neq t \text{ or } |y| \leq \theta \textbf{ then} \\
\quad \textbf{for } i := 0 \text{ to } n \textbf{ do} \\
\qquad w_i := w_i + t x_i \\
\quad \textbf{end for} \\
\textbf{end if}
\end{array}
$$

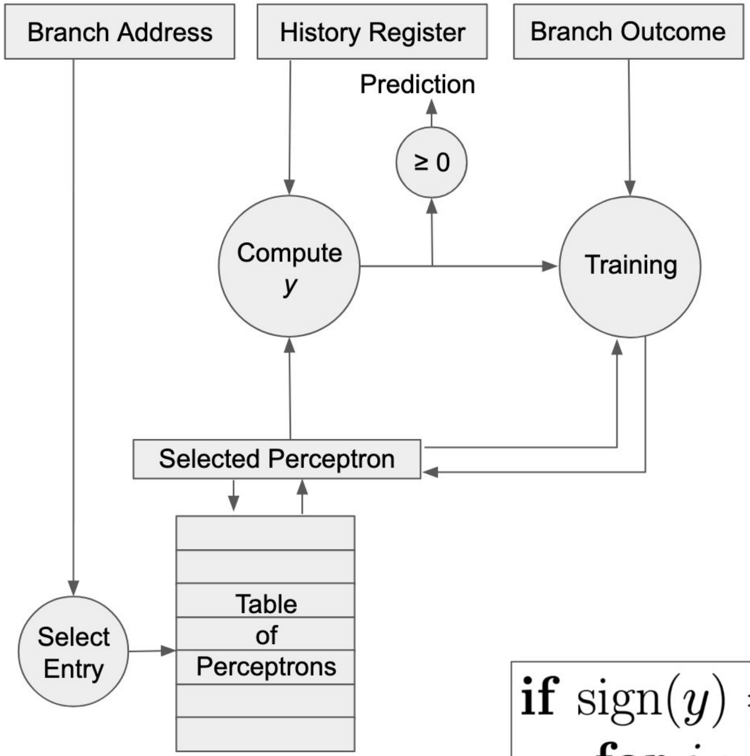


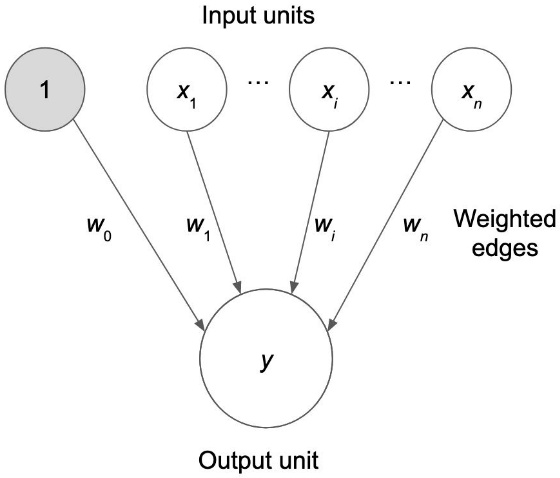


Neural branch predictors
leverage longer branch histories

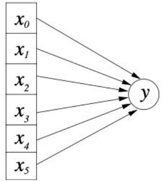
exponential vs. linear
scaling of hardware
resources

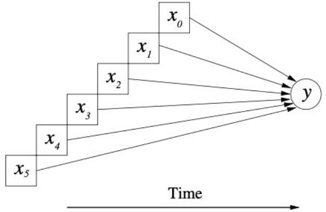
if $\mathrm{sign}(y) \neq t$ or $|y| \leq \theta$ **then**
  for $i := 0$ to n **do**
    $w_i := w_i + t x_i$
  end for
end if





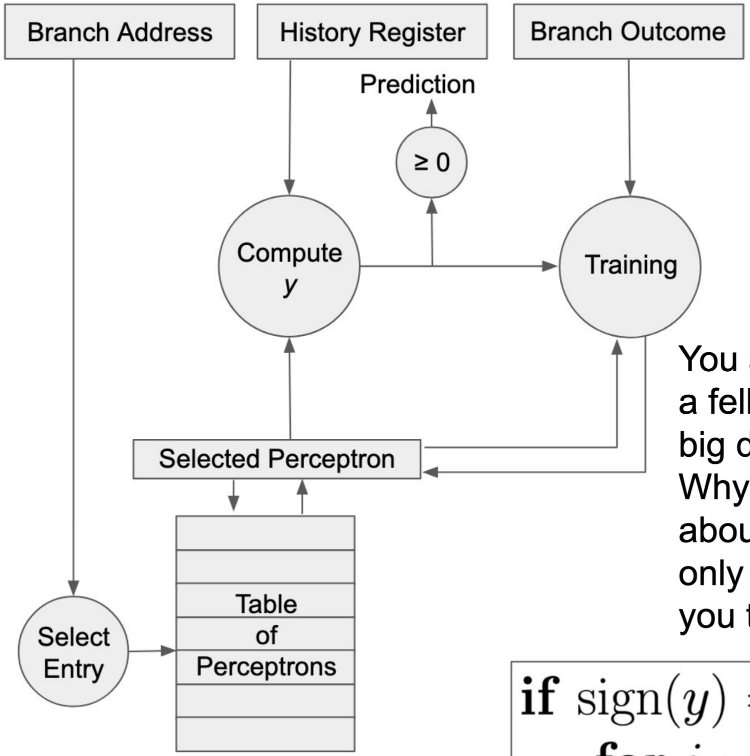
Neural branch predictors leverage longer branch histories



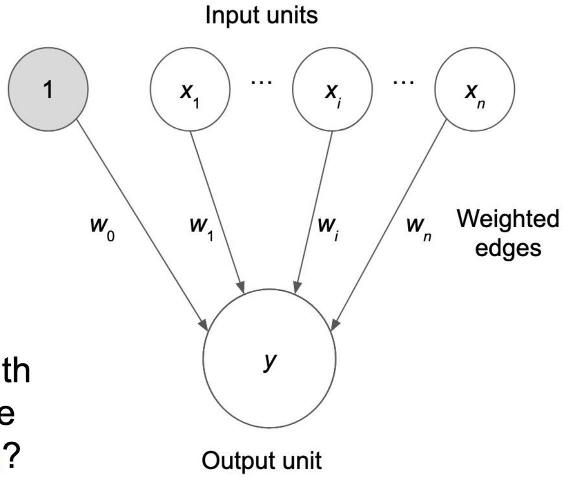


exponential vs. linear scaling of hardware resources

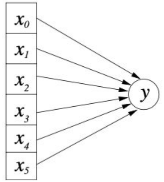
if $\mathrm{sign}(y) \neq t$ or $|y| \leq \theta$ **then**
  for $i := 0$ to n **do**
    $w_i := w_i + t x_i$
  end for
end if

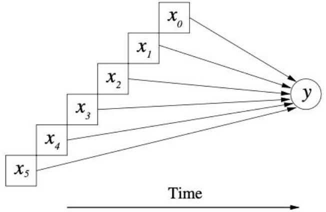


You are at the LRT station chatting with a fellow student. She asks, what is the big deal with neural branch prediction? Why did Quinn spend a lecture talking about it? Your train is coming, you can only say a few sentences. What would you tell her?



Neural branch predictors leverage longer branch histories





exponential vs. linear scaling of hardware resources

if $\text{sign}(y) \neq t$ or $|y| \leq \theta$ **then**
  for $i := 0$ to n **do**
    $w_i := w_i + t x_i$
  end for
end if

Thank you :)

Have a great reading week!

References

D. A. Jimenez and C. Lin, "Dynamic branch prediction with perceptrons," *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, Monterrey, Mexico, 2001, pp. 197-206, doi: 10.1109/HPCA.2001.903263.

Daniel A. Jiménez and Calvin Lin. 2002. Neural methods for dynamic branch prediction. ACM Trans. Comput. Syst. 20, 4 (November 2002), 369–397. https://doi.org/10.1145/571637.571639

D. A. Jimenez, "Fast path-based neural branch prediction," *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, San Diego, CA, USA, 2003, pp. 243-252, doi: 10.1109/MICRO.2003.1253199.