COMP3411 Week 05 Tutorial

Yifan He

z5173587@unsw.edu.au
`https://github.com/hharryyf/COMP3411-24T1-tutoring`

Q1 Decision Tree

- Branch on the feature that has the highest information gain
  - Entropy difference between the parent and the children
  - $H(<p_1,\ldots,p_n>) = \sum_{i=1}^{n} -p_i \cdot \log_2 p_i$

Q1 Decision Tree

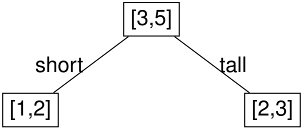
height	hair	eyes	hired
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

- $E_{parent} = -\frac{3}{8} \cdot \log_2 \frac{3}{8} - \frac{5}{8} \cdot \log_2 \frac{5}{8} = 0.954$

Q1 Decision Tree

height	hair	eyes	hired
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

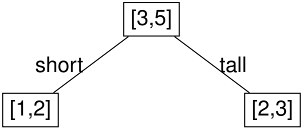
- $E_{parent} = -\frac{3}{8} \cdot \log_2 \frac{3}{8} - \frac{5}{8} \cdot \log_2 \frac{5}{8} = 0.954$
- Branching on **height**



Q1 Decision Tree

height	hair	eyes	hired
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

- $E_{parent} = -\frac{3}{8} \cdot \log_2 \frac{3}{8} - \frac{5}{8} \cdot \log_2 \frac{5}{8} = 0.954$
- Branching on **height**

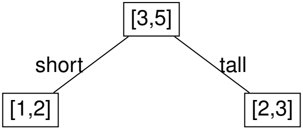


  - $E_{short} = -\frac{1}{3} \cdot \log_2 \frac{1}{3} - \frac{2}{3} \cdot \log_2 \frac{2}{3} = 0.918$
  - $E_{long} = -\frac{2}{5} \cdot \log_2 \frac{2}{5} - \frac{3}{5} \cdot \log_2 \frac{3}{5} = 0.971$

Q1 Decision Tree

height	**hair**	**eyes**	**hired**
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

- $E_{parent} = -\frac{3}{8} \cdot \log_2 \frac{3}{8} - \frac{5}{8} \cdot \log_2 \frac{5}{8} = 0.954$
- Branching on **height**



  - $E_{short} = -\frac{1}{3} \cdot \log_2 \frac{1}{3} - \frac{2}{3} \cdot \log_2 \frac{2}{3} = 0.918$
  - $E_{long} = -\frac{2}{5} \cdot \log_2 \frac{2}{5} - \frac{3}{5} \cdot \log_2 \frac{3}{5} = 0.971$
- $\triangle_{height} = E_{parent} - (\frac{3}{8} \cdot E_{short} + \frac{5}{8} \cdot E_{long}) = 0.003$

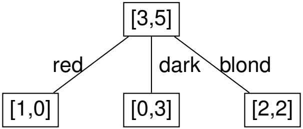
Q1 Decision Tree

height	hair	eyes	hired
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

Q1 Decision Tree

height	hair	eyes	hired
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

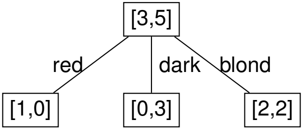
- Branching on **hair**



Q1 Decision Tree

height	**hair**	**eyes**	**hired**
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

- Branching on **hair**

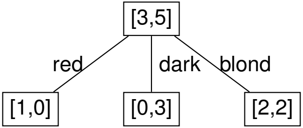


  - $E_{red} = E_{dark} = 0$
  - $E_{blond} = -\frac{2}{4} \cdot \log_2 \frac{2}{4} - \frac{2}{4} \cdot \log_2 \frac{2}{4} = 1$

Q1 Decision Tree

height	hair	eyes	hired
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

- Branching on **hair**

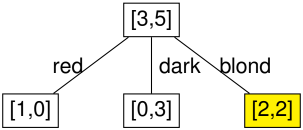


  - $E_{red} = E_{dark} = 0$
  - $E_{blond} = -\frac{2}{4} \cdot \log_2 \frac{2}{4} - \frac{2}{4} \cdot \log_2 \frac{2}{4} = 1$
- $\triangle_{hair} = E_{parent} - (\frac{1}{8} \cdot E_{red} + \frac{3}{8} \cdot E_{dark} + \frac{4}{8} \cdot E_{blond}) = 0.454$

Q1 Decision Tree

height	hair	eyes	hired
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

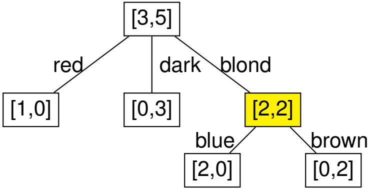
- You can calculate the information gain for **eyes**
- Branching on **hair** is the best
- Branching on eyes in the yellow node is the best



Q1 Decision Tree

height	hair	eyes	hired
short	blond	blue	+
tall	red	blue	+
tall	blond	blue	+
tall	blond	brown	–
short	dark	blue	–
tall	dark	blue	–
tall	dark	brown	–
short	blond	brown	–

- Branching on **hair** is the best
- Branching on eyes in the yellow node is the best



Q2 Laplace Pruning

- Construct the DT to the leaf might overfit

Q2 Laplace Pruning

- Construct the DT to the leaf might overfit
- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes

Q2 Laplace Pruning

- Construct the DT to the leaf might overfit
- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes
- $E = 0$ when the class is pure

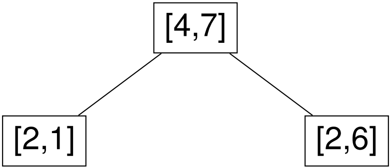
Q2 Laplace Pruning

- Construct the DT to the leaf might overfit
- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes
- $E = 0$ when the class is pure
- We stop growing the tree when the Laplace Error of the parent is smaller than the Backed Up Error

Q2 Laplace Pruning

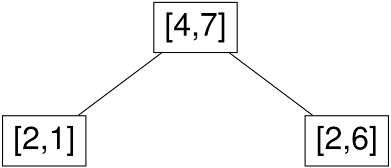
- Construct the DT to the leaf might overfit
- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes
- $E = 0$ when the class is pure
- We stop growing the tree when the Laplace Error of the parent is smaller than the Backed Up Error
- Backed up error is the weighted Laplace Error

Q2 Laplace Pruning



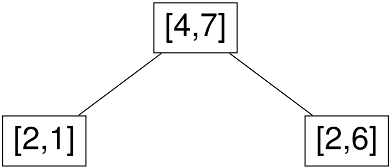
- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes

Q2 Laplace Pruning



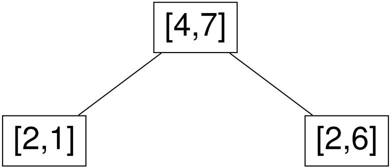
- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes
- $E_{parent} = 1 - \frac{7+1}{11+2} = 0.385$

Q2 Laplace Pruning



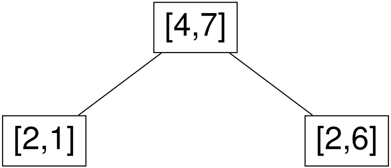
- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes
- $E_{parent} = 1 - \frac{7+1}{11+2} = 0.385$
- $E_{left} = 1 - \frac{2+1}{3+2} = 0.4$

Q2 Laplace Pruning



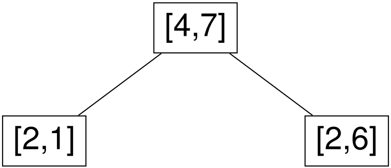
- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes
- $E_{parent} = 1 - \frac{7+1}{11+2} = 0.385$
- $E_{left} = 1 - \frac{2+1}{3+2} = 0.4$
- $E_{right} = 1 - \frac{6+1}{8+2} = 0.3$

Q2 Laplace Pruning



- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes
- $E_{parent} = 1 - \frac{7+1}{11+2} = 0.385$
- $E_{left} = 1 - \frac{2+1}{3+2} = 0.4$
- $E_{right} = 1 - \frac{6+1}{8+2} = 0.3$
- $E_{back} = \frac{3}{11} \cdot E_{left} + \frac{8}{11} \cdot E_{right} = 0.327$

Q2 Laplace Pruning



- Laplace Error $E = 1 - \frac{n+1}{N+k}$
  - n is the items in the majority class
  - N is the total number of items
  - k is the number of classes
- $E_{parent} = 1 - \frac{7+1}{11+2} = 0.385$
- $E_{left} = 1 - \frac{2+1}{3+2} = 0.4$
- $E_{right} = 1 - \frac{6+1}{8+2} = 0.3$
- $E_{back} = \frac{3}{11} \cdot E_{left} + \frac{8}{11} \cdot E_{right} = 0.327$
- $E_{parent} > E_{back}$, don't prune

Q3 Perceptron Learning

Requirement

- Aim: train a linear separable function, that separates positive and negative instances.
- Each instance is a n-dim point $(x_1, x_2, \ldots, x_n)$
- Target function: weights $w_1, \ldots, w_n$ and a bias w_0
- Transfer: $g(x) = w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n + w_0$
  - Positive: $g(x) \geq 0$
  - Negative: $g(x) < 0$

Q3 Perceptron Learning

Perceptron learning algorithm

1. Initialize $w_0, \ \ldots \ , w_n$ ($\vec{\mathbf{w}} =< w_1, w_2, \ldots, w_n >$)
2. Initialize $converge = False$
3. Repeat the following step until $converge = True$
4. $converge = True$
5. For each training instance $< \vec{\mathbf{x}}_{\mathbf{i}}, y_i >$
    1. Let $y' = \vec{\mathbf{x}}_{\mathbf{i}} \cdot \vec{\mathbf{w}} + w_0$
    2. If ($y_i = 1$ and $y' < 0$) or ($y_i = -1$ and $y' \geq 0$)
        - $\vec{\mathbf{w}} = \vec{\mathbf{w}} + \eta \cdot y_i \cdot \vec{\mathbf{x}}_{\mathbf{i}}$
        - $w_0 = w_0 + \eta \cdot y_i$
        - $converge = False$

Q3 Perceptron Learning ($\eta = 1.0$)

Instance	x_1	x_2	y
a	0	1	-1
b	2	0	-1
c	1	1	+1

#	w_0	w_1	w_2	ID	x_1	x_2	y	$y' = \vec{\mathbf{x}}_{\mathbf{i}} \cdot \vec{\mathbf{w}} + w_0$
1	-1.5	0	2	a	0	1	-1	0.5

Q3 Perceptron Learning ($\eta = 1.0$)

Instance	x_1	x_2	y
a	0	1	-1
b	2	0	-1
c	1	1	+1

#	w_0	w_1	w_2	ID	x_1	x_2	y	$y' = \vec{\mathbf{x}}_{\mathbf{i}} \cdot \vec{\mathbf{w}} + w_0$
1	-1.5	0	2	a	0	1	-1	0.5
2	-2.5	0	1	b	2	0	-1	-2.5

Q3 Perceptron Learning ($\eta = 1.0$)

Instance	x_1	x_2	y
a	0	1	-1
b	2	0	-1
c	1	1	+1

#	w_0	w_1	w_2	ID	x_1	x_2	y	$y' = \vec{\mathbf{x}}_\mathbf{i} \cdot \vec{\mathbf{w}} + w_0$
1	-1.5	0	2	a	0	1	-1	0.5
2	-2.5	0	1	b	2	0	-1	-2.5
3	-2.5	0	1	c	1	1	+1	-1.5

Q3 Perceptron Learning ($\eta = 1.0$)

Instance	x_1	x_2	y
a	0	1	-1
b	2	0	-1
c	1	1	+1

#	w_0	w_1	w_2	ID	x_1	x_2	y	$y' = \vec{\mathbf{x}}_\mathbf{i} \cdot \vec{\mathbf{w}} + w_0$
1	-1.5	0	2	a	0	1	-1	0.5
2	-2.5	0	1	b	2	0	-1	-2.5
3	-2.5	0	1	c	1	1	+1	-1.5
4	-1.5	1	2	a	0	1	-1	0.5

Q3 Perceptron Learning ($\eta = 1.0$)

Instance	x_1	x_2	y
a	0	1	-1
b	2	0	-1
c	1	1	+1

#	w_0	w_1	w_2	ID	x_1	x_2	y	$y' = \vec{\mathbf{x}}_{\mathbf{i}} \cdot \vec{\mathbf{w}} + w_0$
1	-1.5	0	2	a	0	1	-1	0.5
2	-2.5	0	1	b	2	0	-1	-2.5
3	-2.5	0	1	c	1	1	+1	-1.5
4	-1.5	1	2	a	0	1	-1	0.5
5	-2.5	1	1	b	2	0	-1	-0.5

Q3 Perceptron Learning ($\eta = 1.0$)

Instance	x_1	x_2	y
a	0	1	-1
b	2	0	-1
c	1	1	+1

#	w_0	w_1	w_2	ID	x_1	x_2	y	$y' = \vec{\mathbf{x}}_{\mathbf{i}} \cdot \vec{\mathbf{w}} + w_0$
1	-1.5	0	2	a	0	1	-1	0.5
2	-2.5	0	1	b	2	0	-1	-2.5
3	-2.5	0	1	c	1	1	+1	-1.5
4	-1.5	1	2	a	0	1	-1	0.5
5	-2.5	1	1	b	2	0	-1	-0.5
6	-2.5	1	1	c	1	1	+1	-0.5

Q3 Perceptron Learning ($\eta = 1.0$)

Instance	x_1	x_2	y
a	0	1	-1
b	2	0	-1
c	1	1	+1

#	w_0	w_1	w_2	ID	x_1	x_2	y	$y' = \vec{\mathbf{x}}_{\mathbf{i}} \cdot \vec{\mathbf{w}} + w_0$
1	-1.5	0	2	a	0	1	-1	0.5
2	-2.5	0	1	b	2	0	-1	-2.5
3	-2.5	0	1	c	1	1	+1	-1.5
4	-1.5	1	2	a	0	1	-1	0.5
5	-2.5	1	1	b	2	0	-1	-0.5
6	-2.5	1	1	c	1	1	+1	-0.5
7	-1.5	2	2	a	0	1	-1	0.5

Q3 Perceptron Learning ($\eta = 1.0$)

Instance	x_1	x_2	y
a	0	1	-1
b	2	0	-1
c	1	1	+1

#	w_0	w_1	w_2	ID	x_1	x_2	y	$y' = \vec{\mathbf{x}}_\mathbf{i} \cdot \vec{\mathbf{w}} + w_0$
1	-1.5	0	2	a	0	1	-1	0.5
2	-2.5	0	1	b	2	0	-1	-2.5
3	-2.5	0	1	c	1	1	+1	-1.5
4	-1.5	1	2	a	0	1	-1	0.5
5	-2.5	1	1	b	2	0	-1	-0.5
6	-2.5	1	1	c	1	1	+1	-0.5
7	-1.5	2	2	a	0	1	-1	0.5
8	…	…	…	…	…	…	…	…

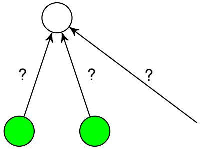
Q4 Neural network for logical formula

Requirement

- Input for each variable is 0 (False) or 1 (True)
- The expression is true iff *output* ≥ 0

Perceptron for logical OR

- 2 inputs



- The output is greater than 0 if either green node is 1
- The output is less than 0 if both green nodes are 0

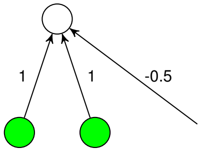
Q4 Neural network for logical formula

Requirement

- Input for each variable is 0 (False) or 1 (True)
- The expression is true iff *output* $\geq$ 0

Perceptron for logical OR

- 2 inputs



- The output is greater than 0 if either green node is 1
- The output is less than 0 if both green nodes are 0

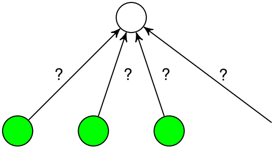
Q4 Neural network for logical formula

Requirement

- Input for each variable is 0 (False) or 1 (True)
- The expression is true iff *output* $\geq$ 0

Perceptron for logical OR

- m inputs



- The output is greater than 0 if either green node is 1
- The output is less than 0 if all green nodes are 0

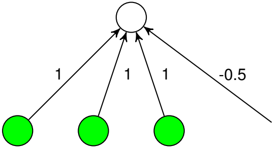
Q4 Neural network for logical formula

Requirement

- Input for each variable is 0 (False) or 1 (True)
- The expression is true iff *output* $\geq$ 0

Perceptron for logical OR

- m inputs



- The output is greater than 0 if either green node is 1
- The output is less than 0 if all green nodes are 0
- The weights are 1, the bias is -0.5

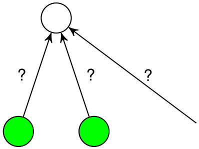
Q4 Neural network for logical formula

Requirement

- Input for each variable is 0 (False) or 1 (True)
- The expression is true iff *output* $\geq$ 0

Perceptron for logical AND

- 2 inputs



- Output is greater than 0 if both green nodes are 1
- The output is less than 0 if either green node is 0

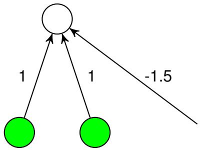
Q4 Neural network for logical formula

Requirement

- Input for each variable is 0 (False) or 1 (True)
- The expression is true iff *output* $\geq$ 0

Perceptron for logical AND

- 2 inputs



- Output is greater than 0 if both green nodes are 1
- The output is less than 0 if either green node is 0

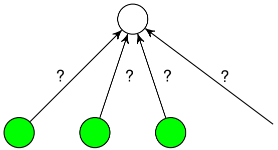
Q4 Neural network for logical formula

Requirement

- Input for each variable is 0 (False) or 1 (True)
- The expression is true iff *output* $\geq$ 0

Perceptron for logical AND

- m inputs



- Output is greater than 0 if all green nodes are 1
- The output is less than 0, otherwise

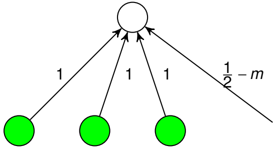
Q4 Neural network for logical formula

Requirement

- Input for each variable is 0 (False) or 1 (True)
- The expression is true iff *output* $\geq$ 0

Perceptron for logical AND

- m inputs

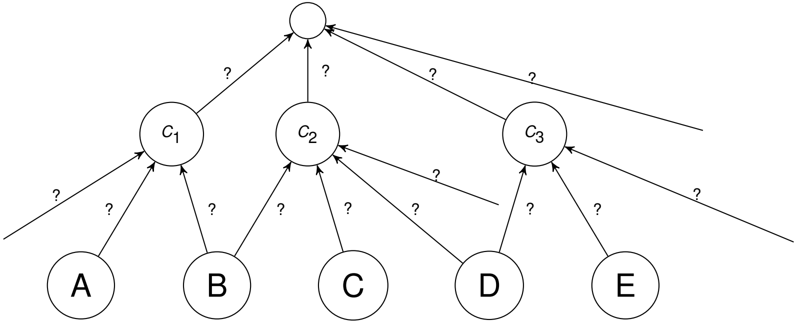


- Output is greater than 0 if all green nodes are 1
- The output is less than 0, otherwise
- The weights are all 1, the bias is $\frac{1}{2} - m$

Q4 Neural network for logical formula

2-layer Neural network for CNF

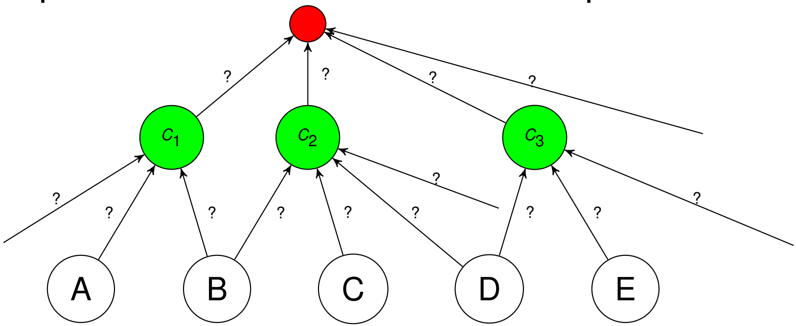
- $(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$
- 1 input node for each variable, 1 output node



Q4 Neural network for logical formula

2-layer Neural network for CNF

- $(A \lor B) \land (\neg B \lor C \lor \neg D) \land (D \lor \neg E)$
- 1 input node for each variable, 1 output node

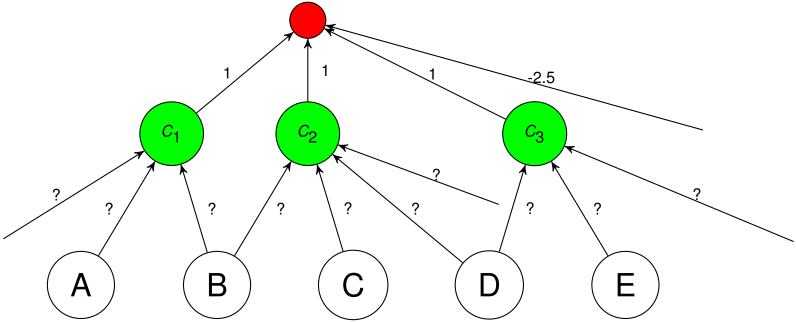


- Solve components by components
- Red node is true iff all green nodes are true

Q4 Neural network for logical formula

2-layer Neural network for CNF

- $(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$
- 1 input node for each variable, 1 output node

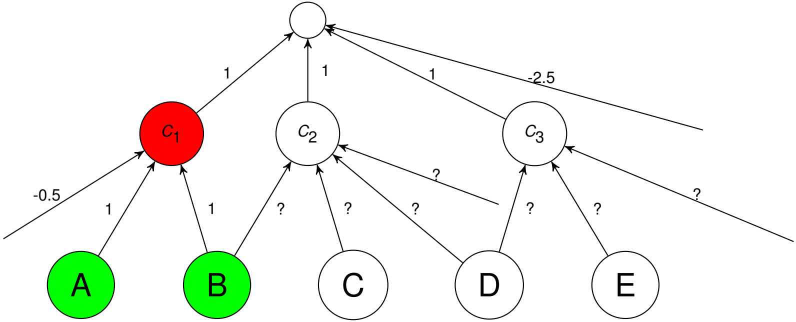


- Red node is true iff all green nodes are true
- Use the ”AND modeling“ method

Q4 Neural network for logical formula

2-layer Neural network for CNF

- $(A \lor B) \land (\neg B \lor C \lor \neg D) \land (D \lor \neg E)$
- 1 input node for each variable, 1 output node

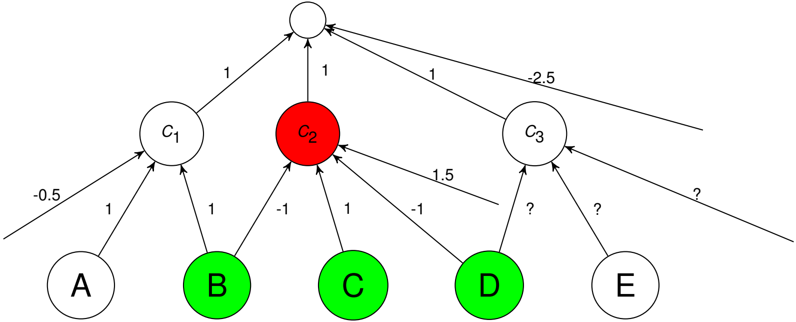


- C_1 node is true iff either A or B is true
- Use the "OR modeling" method

Q4 Neural network for logical formula

2-layer Neural network for CNF

- $(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$
- 1 input node for each variable, 1 output node

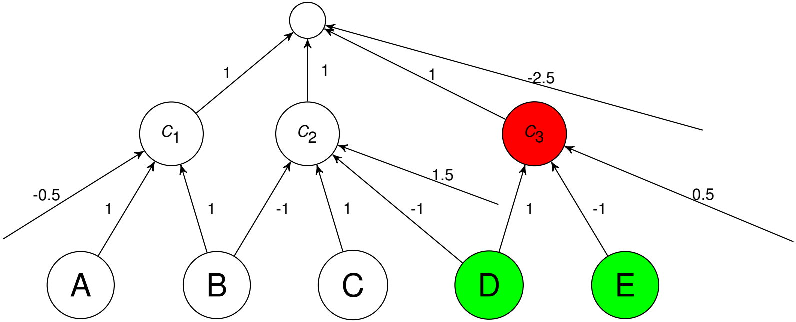


- C_2 node is false iff B and D are true, C is false

Q4 Neural network for logical formula

2-layer Neural network for CNF

- $(A \lor B) \land (\neg B \lor C \lor \neg D) \land (D \lor \neg E)$
- 1 input node for each variable, 1 output node



- C_2 node is false iff D is false and E is true