Nome: Dennis José da Silva, NUSP: 9176517, dennis@ime.usp.br
Universidade de São Paulo
Instituto de Matemática e Estatística
MAC5832 - Aprendizagem Computacional: Modelos, Algoritmos e Aplicações

Problem 1:

Resposta.:

Exemplos positivos: 101, 1100, 1010101, 1111, 0011
Exemplos Negativos: 010, 1000, 101010, 111, 1

Problem 2:

Resposta.:

Exemplos positivos: 101, 1001, 1111, 0000, 110011, 1, 0
Exemplos negativos: 10, 1100, 1010, 1110, 011101, 01

Problem 3:

Resposta:

Exemplos positivos: (0.1, 0.1, 0.1), (0.1, 0.5), (0.5, 0.3, 0.6, 0.1)
Exemplos negativos: (1, 1, 1), (2, 1, 2), (3.5, 4.5, 1.1)

Problem 4:

Resposta:

Função alvo = $função\ alvo = < u_1 \bar{u}_2 u_4 u_7 >$

(1001111, 1) = $U = \{\cancel{\bar{u}_1}, u_1, \bar{u}_2, \cancel{u_2}, \bar{u}_3, \cancel{u_3}, \cancel{\bar{u}_4}, u_4, \cancel{\bar{u}_5}, u_5, \cancel{\bar{u}_6}, u_6, \cancel{\bar{u}_7}, u_7\}$
(0110110, 0) = Conjunto U não modificado por que $b_2 = 0$
(1011101, 1) = $U = \{u_1, \bar{u}_2, \cancel{\bar{u}_3}, u_4, u_5, \cancel{u_6}, u_7\}$
(1011001, 1) = $U = \{u_1, \bar{u}_2, u_4, \cancel{u_5}, u_7\}$

$$U = \{u_1, \bar{u}_2, u_4, u_7\} = < u_1 \bar{u}_2 u_4 u_7 >$$

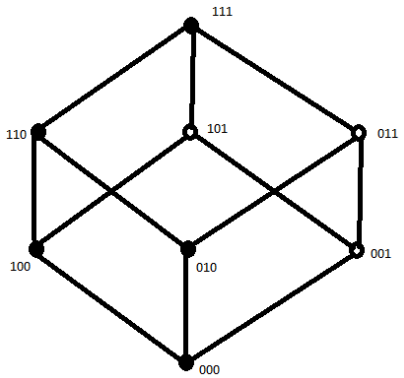
Problem 5:

Resposta:

Três possíveis formas são por reticulado booleano, função booleana e portas lógicas, por exemplo, dada a seguinte tabela para a função $f: \{0,1\}^3 \rightarrow \{0, 1\}$:

x_1	x_2	x_3	$f(x)$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Pode ser representado como:
Por reticulado:

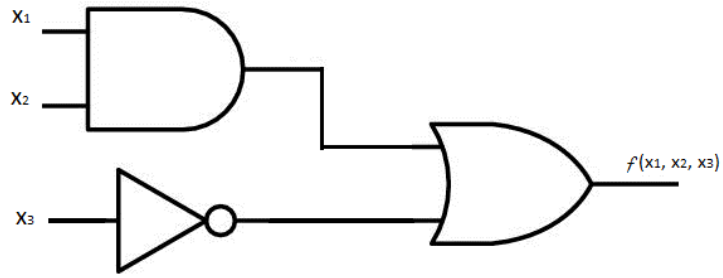


Repare que os pontos pintados em preto representam os pontos onde a função vale 1.

Por Função Booleana:

$$f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee \neg x_3$$

Por portas lógicas:



Problem 6:

Resposta:

ATENÇÃO: Os programas foram desenvolvidos em python versão = 3.4.1 no S.O Windows 7

Perceptron desenvolvido em linguagem Python. O arquivo fonte está na pasta problem 6 juntamente com as instruções de uso, resumo abaixo:

Executar programa de geração de amostras: “samples_generator.py” que recebe os parâmetros:
-s = sample-number, quantidade de amostras a serem geradas. Valor padrão = 100
-t = threshold, valor de θ; valor padrão = 5.0
-i = ini, valor inicial do domínio da função. Valor padrão = 0.0
-e = end, valor final do domínio da função. Valor padrão = 10.0
-o = output file, arquivo de amostras no formato json, Valor padrão ./samples.json

A executar o gerador de amostras serão gerados amostras em json com o seguinte formato: [[valor gerado, {1 ou 0}], [valor gerado, {1 ou 0}], [valor gerado, {1 ou 0}]]

Então executar o programa “main .py” que gera o classificador apartir do algoritmo de perceptron, esse programa recebe o seguinte parâmetro:
-t = training_data_file, arquivo json geradpo pelo programa samples_generator. Valor padrão = “./samples.json”

Então entre com os valores de x, e o programa retorna o y de acordo com o conceito aprendido.

O algoritmo do perceptron está no arquivo “Perceptron.py”

Problem 7:

Resposta:

ATENÇÃO: Os programas foram desenvolvidos em python versão = 3.4.1 no S.O Windows 7

Os perceptrons foram desenvolvido em linguagem Python. O arquivo fonte está na pasta problem 7 juntamente com as instruções de uso, resumo abaixo:

Executar programa de geração de amostras: “samples_generator.py” que recebe os parâmetros:
-b = bound, limitante superior ou inferior, pode receber os valores “upper” ou “lower”, sem valor padrão;
-s = sample-number, quantidade de amostras a serem geradas. Valor padrão = 100
-t = threshold, valor de θ; valor padrão = 5.0
-i = ini, valor inicial do domínio da função. Valor padrão = 0.0
-e = end, valor final do domínio da função. Valor padrão = 10.0
-o = output file, arquivo de amostras no formato json, Valor padrão ./samples.json

A executar o gerador de amostras serão gerados amostras em json com o seguinte formato: [[valor gerado, {1 ou 0}], [valor gerado, {1 ou 0}], [valor gerado, {1 ou 0}]]

Para a solução do exercício, será necessário executar duas vezes o “sample_generator.py”, um para o limitante inferior “lower” e outro para o superior “upper”, grave os arquivos json gerados com nomes diferentes e execute o “main.py”

Então executar o programa “main .py” que gera o classificador apartir do algoritmo de perceptron, esse programa recebe os seguinte parâmetro:
-l = lower_tranning_data_file, arquivo json gerado pelo programa samples_generator para limitante inferior. Valor padrão = “./lower_samples.json”
-u = upper_bound_tranning_data_file, arquivo json gerado pelo programa samples_generator para limitante superior. Valor padrão = “./upper_samples.json”

O algoritmo do perceptron está no arquivo “Perceptron.py”, nesse o algoritmo gerara dois perceptron e juntara os dois para classificar a entrada pelo console.

O algoritmo do perceptron está no arquivo “Perceptron.py”

Então entre com os valores de x, e o programa retorna o y de acordo com o conceito aprendido.

Problem 8:

Resposta:

Sabendo que um monômio booleano representa um número qualquer com no máximo k literais podemos dizer que que $M_{k,n} = 2^n$, porque teremos uma função que mapeia um monômio qualquer em 1 e outra em 0, como temos "n" monômios podemos ter 2^n combinações de 1 e 0 como resultado de cada monômio, logo podemos ter 2^n funções. Para D_{mk} cada função de M_{mk} se torna uma variável das disjunções de D_{mk} como sabemos que para qualquer disjunção representa a operação OU da álgebra booleana essas funções de D_{mk} só podem ser 0 quando todas as suas variáveis são 0 (quando todas as funções de M_{mk} assumem valor 0 ao mesmo tempo) e 1 quando possui alguma variável 1 (quando pelo menos uma função de M_{mk} assume valor 1), sabemos que podemos ter D_{mk} é igual a números de combinações dos resultados das funções de M_{mk} e como sabemos que M_{mk} pode representar 2^n funções, logo $D_{mk} = 2^n$**. Logo** Podemos calcular |H| como:

$$\mathbf{|H| = 2^n \cdot 2^n = 2^{2n}}$$

Problem 9:

Resposta:

Rodar programa main.py na pasta problem9, ele gera duas imagens a problem9-1.png para o exercício 9 (1) e problem9-2.png para o exercício 9 (**2)**