Relatório Final - Simulação Visual do Algoritmo Perceptron

Pedro Paulo Vezzá Campos 7538743
Camila Fernandez Achutti 6795610

4 de dezembro de 2013

1 Proposta escolhida

O grupo optou pela proposta 2, animação de algoritmos de aprendizagem computacional. O algoritmo implementado e simulado visualmente foi o Perceptron.

2 Sobre o problema de aprendizagem

2.1 PERCEPTRON

O algoritmo Perceptron foi inventado em 1957 no Cornell Aeronautical Laboratory por Frank Rosenblatt. Ele é um classificador supervisionado linear. Ou seja, a partir de um conjunto de dados de treinamento, o algoritmo ajusta uma função linear que será utilizada para a classificação de novas instâncias de teste.

Na sua versão de camada única, é capaz de classificar instâncias de duas classes distintas, mapeando entradas x para valores de saída $f(x)$ (valores binários simples) através da função.

$$f(x) = \begin{cases} 1 & \text{se } w \cdot x + b > 0 \\ 0 & \text{senão} \end{cases}$$

Onde x e w são vetores e $w \cdot x$ é o produto escalar. b é a 'inclinação', um termo constante que não depende de qualquer valor de entrada. Sua principal característica e limitação é que apenas possui resultados satisfatórios para a classificação de conjuntos linearmente separáveis.

Assim, instâncias interessantes para o classificador são conjuntos de dados que podem ou não ser classificados utilizando uma reta para dividir as categorias de elementos. O caso clássico de conjunto não linearmente separável é a função ou-exclusivo (XOR). Dentro dos problemas linearmente separáveis, é interessante considerar casos em que os pontos de dados estão muito ou pouco separados para verificar a eficiência do classificador.

Neste trabalho implementamos o perceptron para dimensão arbitrária d $\geq$ 2.

3 Interface Gráfica

Foi desenvolvida em 2D, ainda que a dimensão d seja maior que 2. Vamos sempre fazer a representação somente de 2 delas.

A implementação será feita em python e a plotagem será feita usando PyLab e a interface do usuário usando PyGTK.

A entrada pode ser efetuada de 3 maneiras diferentes:

- marcar os pontos no canvas através de clique (opção disponível somente para dimensão d igual a 2),
- ler de um arquivo de entrada,
- gerar pontos de forma aleatória.

Com o dataset de entrada o algoritmo de aprendizagem entra em jogo e usa os pontos de amostragem como um conjunto de treinamento para tentar descobrir qual é a linha mais adequada para realizar a classificação.

A animação mostra como o algoritmo evolui e quais são os ajustes feitos para alcançar o resultado final.

A ferramenta de animação terá as seguintes funcionalidades básicas:

- Seleção do modo de entrada: Arquivo, Sorteio, Clique;
  - Arquivo: serão necessários a submissão de dois arquivos diferentes, um de dados de treino e outro de testes. Ambos são arquivos CSV onde o número de colunas deve ser a dimensão escolhida mais uma unidade para designação da classe de cada amostra (-1 ou 1).
  - Sorteio: dados aleatórios são gerados, somente com o controle de serem gerados dois conjuntos linearmente separáveis. Esse conjuntos são gerados por uma distribuição uniforme.
  - Clique: na primeira janela que for aberta serão marcados os pontos de treinamento. Para diferenciar uma classe da outra utilizamos os botões direito e esquerdo do mouse. Assim que todos os dados de treinamento tiverem sido desenhados, o usuário fecha a janela de marcação e outra janela se abrirá para marcação dos dados de teste. ATENÇÃO: essa opção de entrada somente está disponível para dimensão $d = 2$.
- Botão de 'TREINAR' para que o algoritmo comece a rodar, ao final do algoritmo ele pode ser novamente apertado para que o treinamento seja refeito;
- Botão 'TESTAR' para que o conjunto de teste seja plotado e o resultado possa ser validado pelo usuário;
- Possibilidade de ajustar o valor de η (taxa de aprendizagem, `eta` no programa) para controlar a rapidez com que o perceptron aprende;
- Definição do número máximo de iterações, onde uma iteração representa a atualização do vetor de pesos a cada ponto que é analisado, sendo assim, se a reta inicial arbitrária for correta, encontramos a solução em 0 iterações do algoritmo.

- Definição do número de dimensões dos dados de entrada e consequentemente do algoritmo perceptron.

A taxa de aprendizagem é a que dimensiona o vetor de treinamento antes de ser adicionado ao vetor de peso durante as atualizações. Experimente valores diferentes, se quiser, valores maiores afetará a taxa de convergência, mas não a própria convergência.

O número de iterações é colocado de modo que o algoritmo não será executado para sempre se os vetores de entrada não são separáveis.

Normalmente, um valor maior de η fará com que ele encontre uma solução mais rapidamente, mas pode causar a perda de soluções de casos difíceis. E a definição do número máximo de iteração garante que o algoritmo vai ter fim, ainda que não encontre um solução de erro global nulo.

A sequência esperada do usuário e da animação é a seguinte:

1. seleção do modo de entrada
2. plotagem do conjunto de treinamento
3. clique do usuário no botão 'TREINAR'
4. plotagem das retas que o algoritmo encontrou durante a execução
5. clique no botão 'TESTAR' para que o conjunto de teste seja plotado e o algoritmo possa ser avaliado pelo usuário.

O botão de 'TREINAR' pode ser usado para que o algoritmo refaça o treinamento com os mesmos dados, mas com a possibilidade de trocar os parâmetros `eta` e `máximo de iterações`.

4 Cronograma

Na tabela a seguir estão listadas as tarefas cumpridas para o projeto de MAC0460 de 2013:

Atividade	setembro	outubro	novembro	dezembro
Implementar o Perceptron	X			
Buscar na Internet e testar entradas interessantes como exemplo	X	X		
Implementar a interface gráfica	X	X	X	
Escrever o relatório parcial		X		
Escrever o relatório final			X	X

Tabela 1: Cronograma das tarefas a serem realizadas no semestre

O cronograma proposto foi cumprido e não sofreu qualquer alteração durante o decorrer do trabalho.

5 O que foi feito

5.1 Estudo e implementação do algoritmo perceptron em Python

Realizamos um estudo detalhado do funcionamento do algoritmo e realizamos a seguinte implementação em Python:

```
1  class Perceptron:
2    """
3    Classe que implementa o algoritmo Perceptron propriamente
4    dito. Adaptado de:
5
6    http://glowingpython.blogspot.com.br/2011/10/perceptron.html
7    """
8    def __init__(self, eta, max_iterations, dimension):
9      """ Construtor do objeto Perceptron, recebe a taxa de
10     aprendizado (eta) e o número máximo de iterações
11     que o algoritmo pode executar durante um treinamento. """
12     self.w = rand(dimension)*2-1 # weights
13     self.learningRate = eta
14     self.max_iterations = max_iterations
15     self.history = [list(self.w)]
16     self.dimension = dimension;
17
18   def response(self,x):
19     """ Para um ponto x = [a, b] informa a classificação
20     calculada pelo algoritmo. """
21     y = 0
22     for i in xrange(0,self.dimension):
23       y += x[i]*self.w[i]
24
25     if y >= 0:
26       return 1
27     else:
28       return -1
29
30   def updateWeights(self,x,iterError):
31     """
32     Atualiza o vetor de pesos, w, seguindo a fórmula
33     para um tempo t + 1:
34       w(t + 1) = w(t) + eta * (d - r) * x
35     com d sendo o resultado desejado e r a resposta do
36     algoritmo perceptron
37     """
38     for i in xrange(0, self.dimension):
39       self.w[i] += self.learningRate*iterError*x[i]
40     self.history.append(list(self.w));
41
42   def train(self,data):
43     """
44     Treina o algoritmo com base nos dados passados via
45     parâmetro. O parâmetro é da seguinte forma:
46
47     data = [
48       [x1, y1, ..., r1],
49       [x2, y2, ..., r2],
50       ...
51     ]
52     """
53     learned = False
54     iteration = 0
```

```
55      while not learned:
56        globalError = 0.0
57        for x in data: # for each sample
58          if iteration >= self.max_iterations:
59            break
60          r = self.response(x)
61          if x[self.dimension] != r:
62            iterError = x[self.dimension] - r
63            self.updateWeights(x,iterError)
64            globalError += abs(iterError)
65            iteration += 1
66        #critério de parada
67        if globalError == 0.0 or iteration >= self.max_iterations:
68          print 'iterations',iteration
69          learned = True # stop learning
70
71    def getHistory(self):
72      """
73      Retorna todos os vetores de pesos produzidos durante o
74      treinamento do algoritmo
75      """
76      return list(self.history);
77
78  def generateData(n, dimension):
79    """
80    Gera dois conjuntos de dados linearmente separáveis
81    com n amostras. Para cada elemento, a terceira coordenada
82    indica a classe do elemento.
83    """
84    x = []
85    y = []
86    inputs = []
87
88    for _ in range(n):
89      x = []
90      for i in range(dimension):
91        x.append(random.uniform(0,1))
92      x.append(1)
93      inputs.append(x)
94
95    for _ in range(n):
96      y = []
97      for i in xrange(dimension):
98        y.append(random.uniform(-1, 0))
99      y.append(-1)
100     inputs.append(y)
101
102   return inputs
```

5.2 Busca de exemplos interessantes para teste da animação

XOR O exemplo aqui é a função XOR. Nele não é possível traçar uma única reta (função linear) tal que divida o plano de maneira que as saídas com valor 0 ficam situadas de um lado da reta e as com valor 1 do outro. Entretanto, este problema pode ser solucionado com a criação de uma camada intermediária na rede e graficamente com uma estrutura em três (ou mais) dimensões.

x1	x2	u
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 2: Conjunto de treinamento do exemplo XOR

Violeta para exportação Neste exemplo temos que classificar violetas para serem importadas e as que vão ficar no mercado.

As violetas são classificadas de acordo com uma escala de intensidade de cor que vai de 1 a 8 e outra escala de textura que vai de 1 a 5.

Definimos que a classe 1 é a de exportação e a classe 2 é a para mercado interno e temos o seguinte conjunto de teste:

cor	textura	classe
4	5	1
5	4	1
6	3	1
7	1	1
8	2	1
1	3	2
1	5	2
2	2	2
3	4	2
4	2	2

Tabela 3: Conjunto de treinamento do exemplo das violetas

cor	textura	classe
5	1	1
5	3	2
6	2	1

Tabela 4: Conjunto de teste do exemplo das violetas

5.3 Implementação da interface gráfica

A interface gráfica já desenvolvida se restringe a exibição dos pontos de treinamento e teste e das linhas obtidas durante o treinamento.

Assim que o programa rodar ele abre uma janela como ilustrada abaixo:

A interface com o usuário já está implementada para suportar sequências de cliques errados, como visto abaixo:

Se escolhermos sortear os dados o nosso aplicativo se comporta da seguinte maneira representada pela sequência de fotos abaixo:

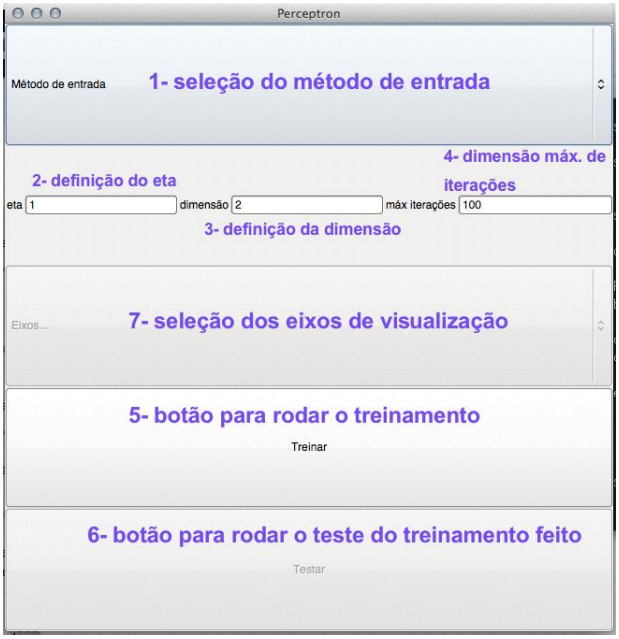

Figura 1: Interface com o usuário.

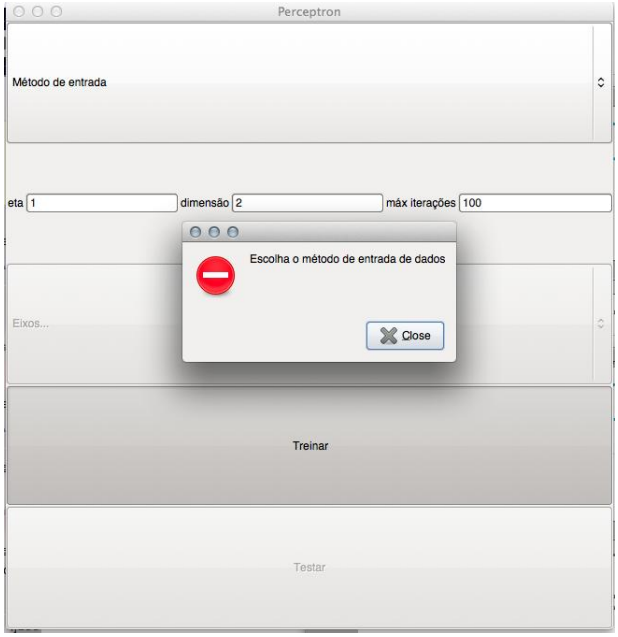

Figura 2: Interface com o usuário.

Agora queremos mostrar como é feito o teste, que é quando o usuário clica no botão TESTAR. Lembrando que só será valido se ele já tiver realizado o treino.

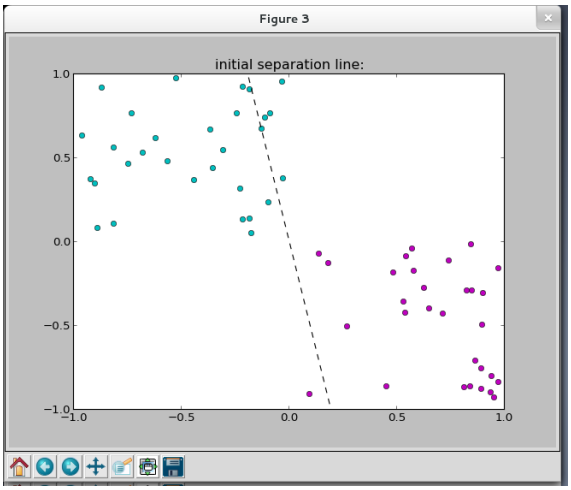

Figura 3: Situação inicial do exemplo 1, onde uma reta arbitrária foi sorteada.

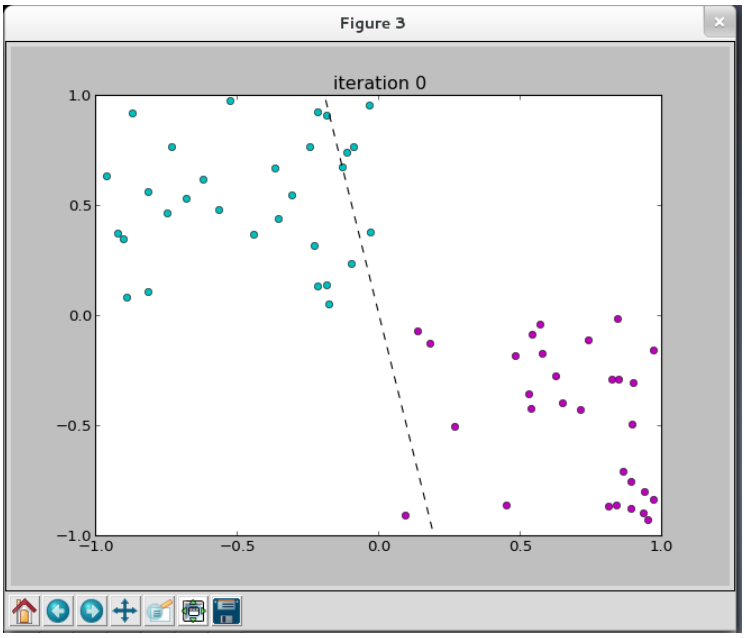

Figura 4: Começa a execução do algoritmo para o exemplo 1.

Agora se o usuário apertar o botão TESTAR os dados de treino serão retirados do gráfico e os dados de teste serão colocados conforme ilustrado nas imagens abaixo:

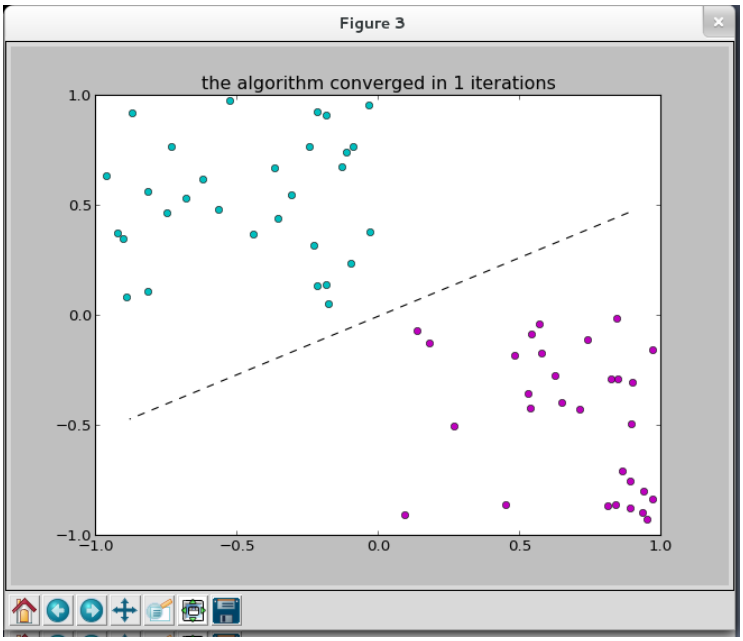

Figura 5: Fim do algoritmo, reta classificadora foi encontrada no exemplo 1.

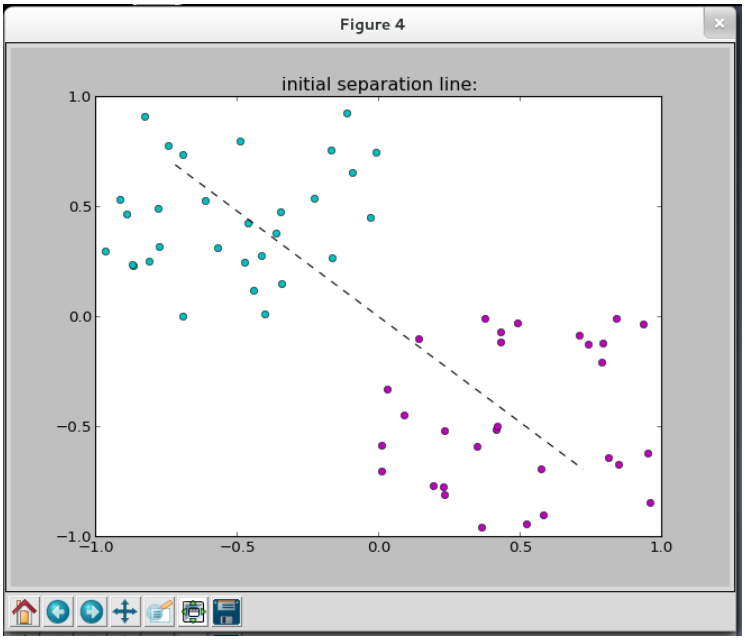

Figura 6: Situação inicial do exemplo 2, onde uma reta arbitrária foi sorteada.

Neste caso apresentado acima temos um treinamento bem sucedido. Agora vamos ilustrar uma caso de treinamento mal-sucedido. Para isso alteramos o eta para um valor baixo para que a taxa de aprendizado fosse bem baixa e

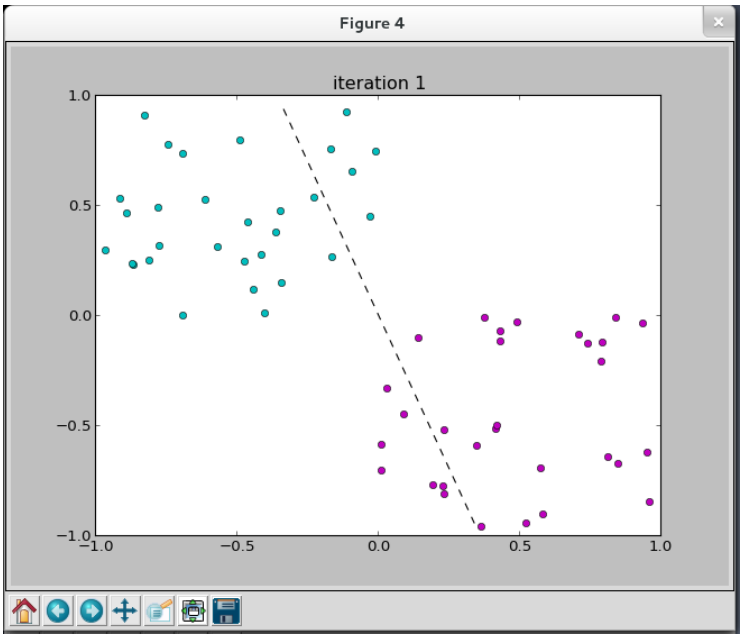

Figura 7: Começa a execução do algoritmo para o exemplo 2.

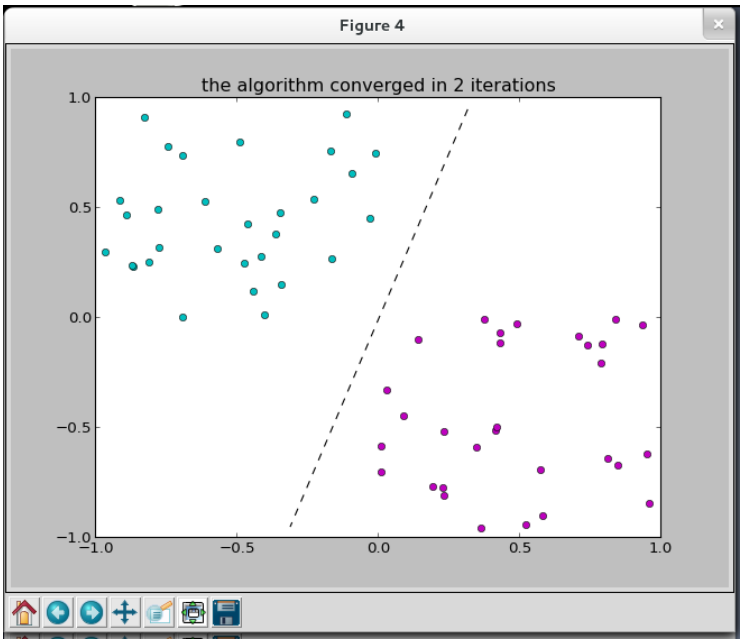

Figura 8: Fim do algoritmo, reta classificadora foi encontrada para o exemplo 2.

limitamos o número de iterações para 1. O resultado obtido está ilustrado abaixo:

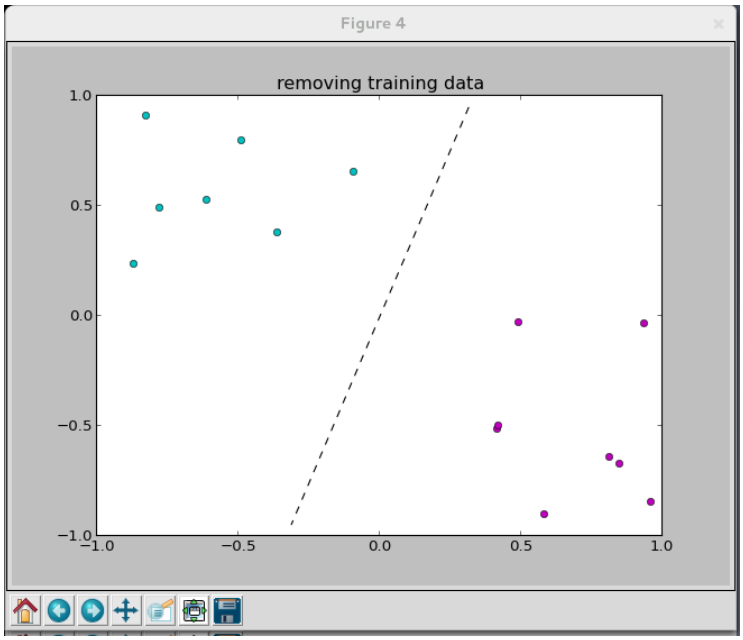

Figura 9: Remoção dos dados de treino.

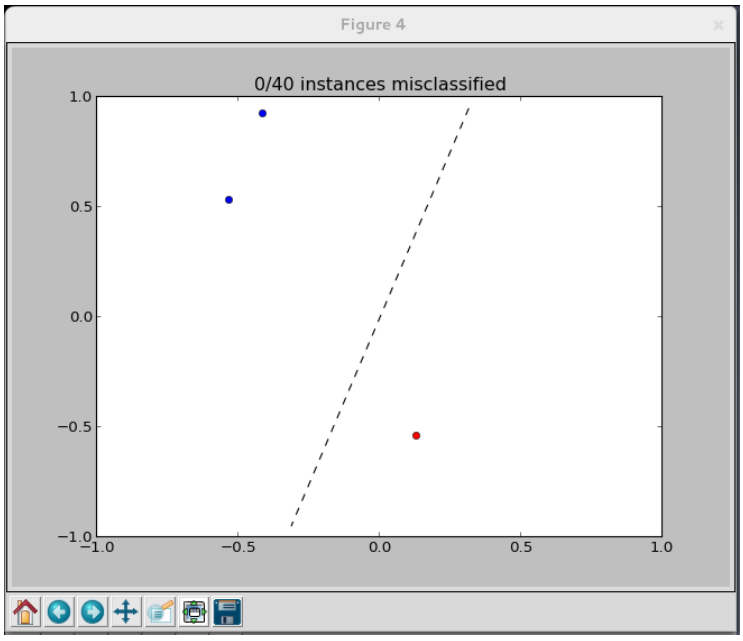

Figura 10: Exibição dos dados de teste.

Aqui esclarecemos que o diamante amarelo representa os dados de teste mal classificados de ambas as amostras.

Depois que o algoritmo terminou de processar o treinamento ou o teste do

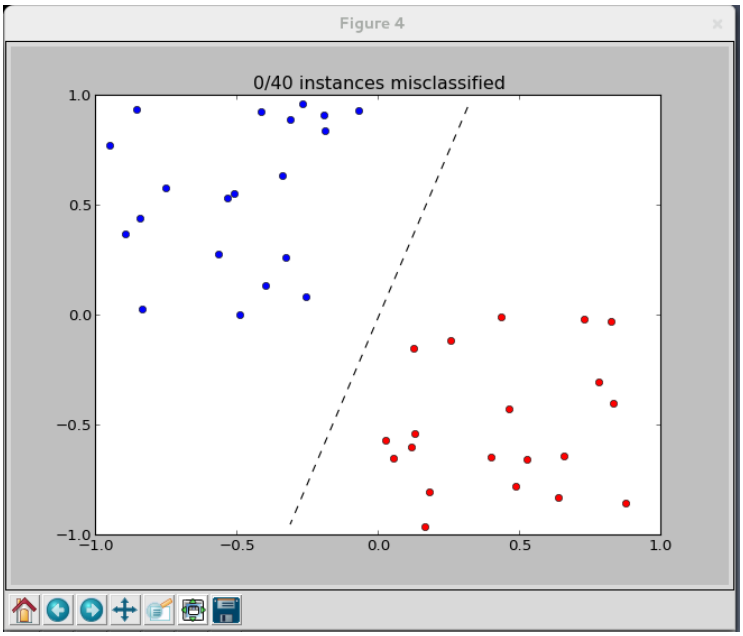

Figura 11: Fim do teste para o exemplo 2.

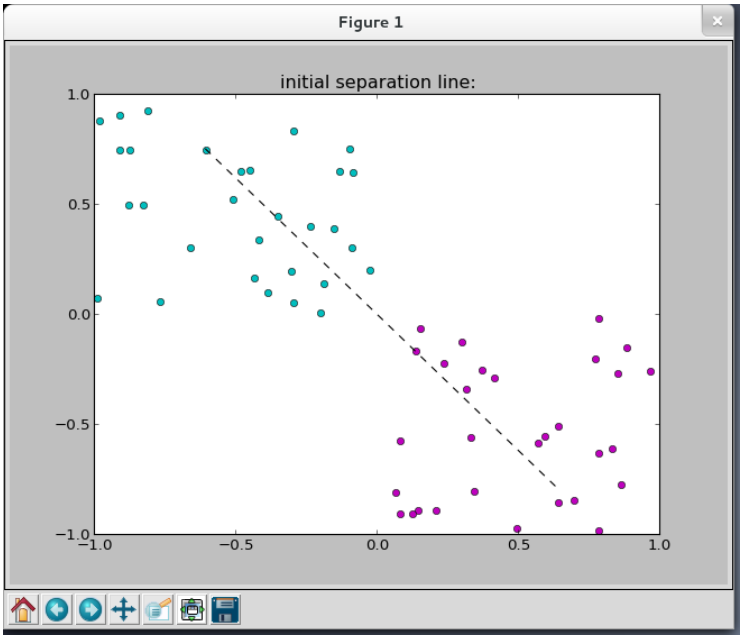

Figura 12: Exibição dos dados de teste.

conjunto de dados, a combobox de seção dos eixos que devem ser exibidos fica habilitada e podemos observar o resultado por todas as perpectivas possíveis. Por exemplo como visto abaixo na sequência de imagens para o

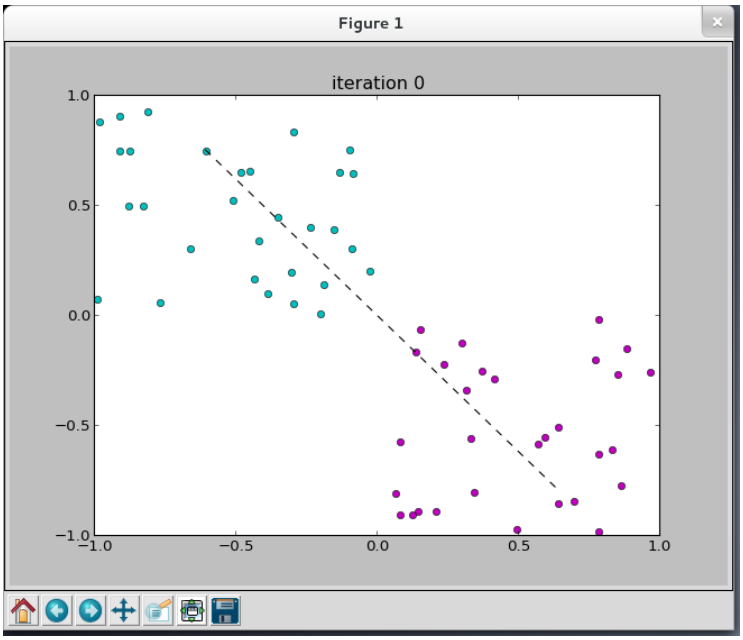

Figura 13: Fim do teste para o exemplo 2.

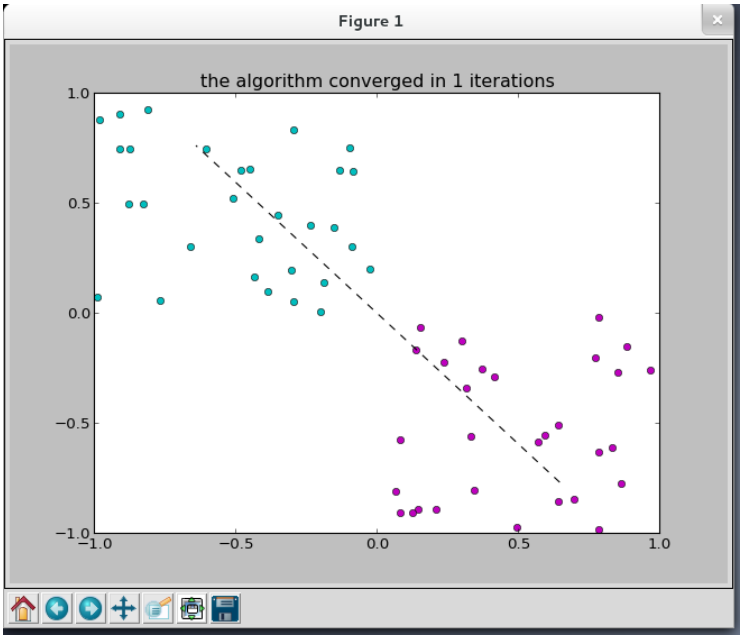

Figura 14: Exibição dos dados de teste.

mesmo resultado de um treinamento em dimensão 3:

- Eixos 1 e 2:
- Eixos 1 e 3:

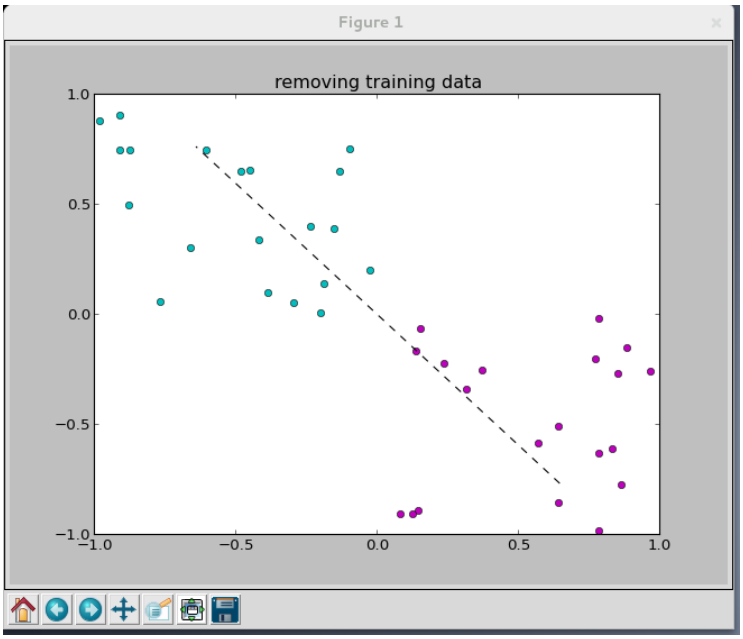

Figura 15: Fim do teste para o exemplo 2.

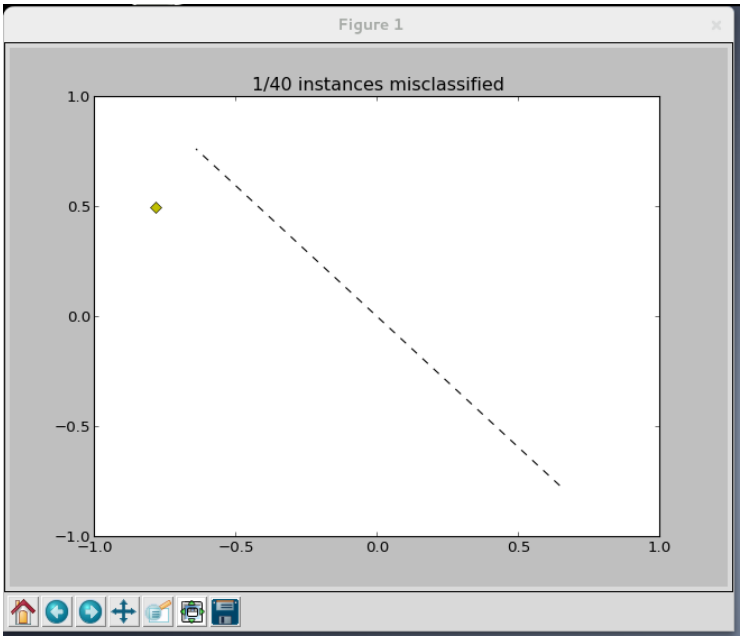

Figura 16: Exibição dos dados de teste.

- Eixos 2 e 3:

É possível também realizar a entrada de dados por arquivos externos. Devem ser produzidos e carregados dois arquivos separados um de treina-

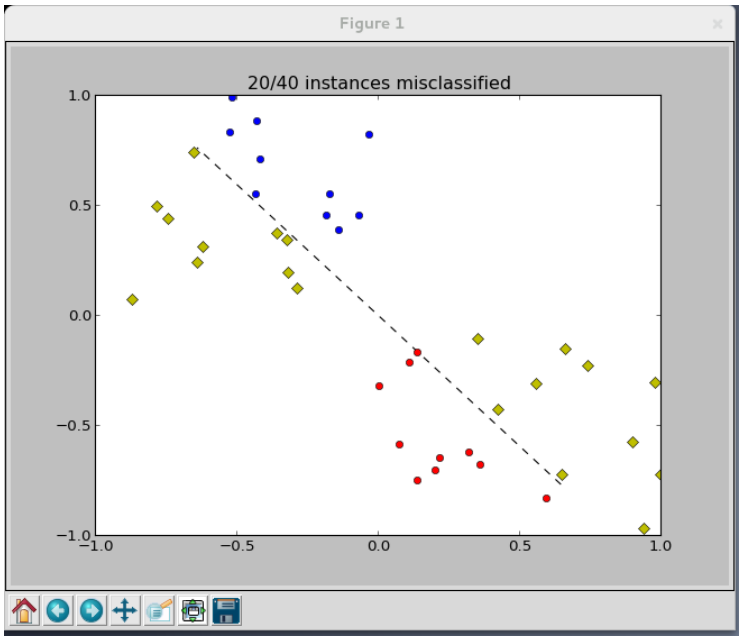

Figura 17: Fim do teste para o exemplo 3.

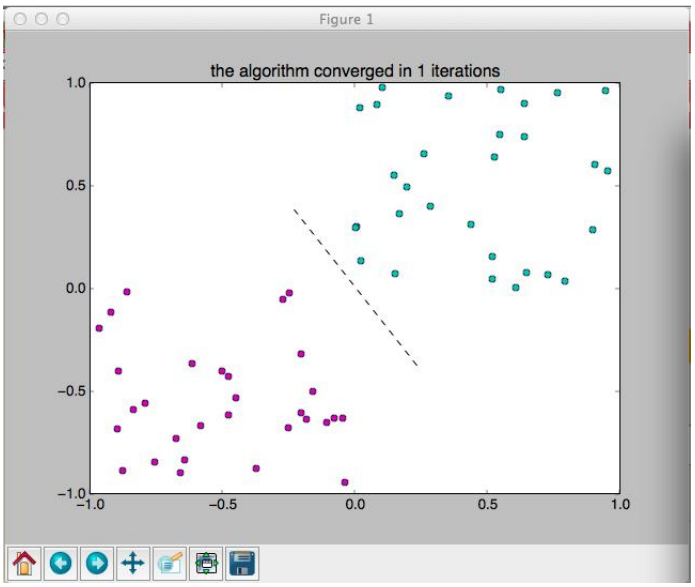

Figura 18: Fim do teste para o exemplo 3.

mento e outro de teste, ambos no formato CSV com o seguinte padrão: `x1, x2, ..., xn, classe`, onde a classe é 1 ou -1 e n é a dimensão escolhida. A entrada pode ainda ser gerada por clique de mouse, onde cliques com o botão esquerdo representam uma classe e cliques com botão direito representam outra classe.

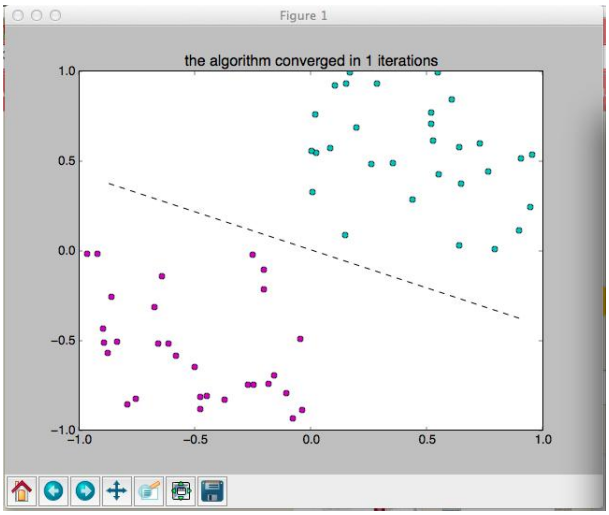

Figura 19: Fim do teste para o exemplo 3.

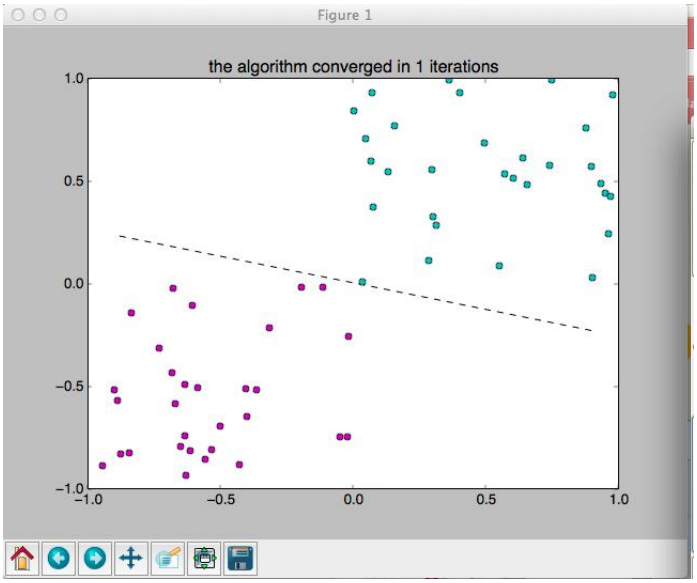

Figura 20: Fim do teste para o exemplo 3.

6 Possíveis trabalhos futuros

- Representação 3D do resultado para dimensões maiores que 2
- Aumentar o número de classes possíveis.

7 Parte Subjetiva

Nesta seção os alunos fazem um balanço da disciplina MAC0460, êxitos e frustrações enfrentadas durante o semestre:

7.1 O que deu certo

Exercícios Práticos Aplicar os algoritmos em programas reais e ver o resultado, muitas vezes visual, incentiva bastante a experimentação e motiva o aluno a buscar a interpretação dos resultados que está recebendo.

Projeto Final O projeto final da disciplina aliou a aplicação de conceitos vistos em aula com a geração de um produto de software potencialmente útil para outros alunos de Aprendizagem Computacional. O contato com alunos da pós-graduação é bastante enriquecedor durante as apresentações finais. Foi possível ver a grande gama de aplicações que os tópicos estudados em sala de aula permitem desenvolver.

7.2 O que não deu certo

Comprometimento dos alunos de graduação A maioria dos alunos de graduação que cursa MAC0460 são alunos de quarto ano ou posterior. No caso do nosso grupo, e de ouro ambos os membros da equipe estavam cumprindo iniciações científicas juntamente com seus respectivos TCCs. Conciliar todas as atividades acadêmicas foi algo extremamente difícil aos alunos e infelizmente nossa participação em sala de aula foi menor que o desejável.

7.3 O que pode melhorar

Descrição da disciplina MAC0460 exigiu uma maturidade matemática maior que o esperado pelo grupo. No início do semestre, já esperávamos um curso instrumental, porém com menor profundidade do que o que foi mostrado. Isto não é um problema por si só. Apenas seria interessante que os alunos fossem melhor avisados do que devem esperar no curso. Uma possibilidade seria disponibilizar as notas de aula para que quem estivesse interessado em MAC0460 soubesse como os tópicos são abordados. Outra ideia é a disponibilização de um vídeo curto no qual a professora descreve a disciplina em mais detalhes. O Projeto Apoio BCC, orientado pelo Prof. Coelho está disponível para operacionalizar a produção do vídeo.

Exercícios programa EPs, em substituição a algumas listas de exercício, poderiam ser aplicados aos alunos para fixar os conceitos vistos em aula. Programar para resolver algum problema proposto tende a ser mais divertido a um aluno de Computação que resolver uma lista de exercícios em papel.

8 Bibliografia

- `http://computing.dcu.ie/~humphrys/Notes/Neural/single.neural.html`

- http://eecs.wsu.edu/~cook/ai/lectures/applets/perceptron/
- http://matplotlib.org/
- http://www.pygtk.org/