Relatório Parcial - Projeto de Aprendizagem Computacional

Pedro Paulo Vezzá Campos 7538743
Camila Fernandez Achutti 6795610

8 de outubro de 2013

1 Proposta escolhida

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

2 Sobre o problema de aprendizagem

2.1 PERCEPTRON

Como explicado no enunciado o Perceptron é um algoritmo de classificação supervisionada. 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 esperamos implementar o perceptron para dimensão arbitrária $d \geq 2$.

3 Interface Gráfica

Foi e será 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 click,
- ler de um arquivo de entrada,
- gerar pontos de forma aleatória.

Com o dataset de entrada o algoritmo de aprendizagem entra em jogo e se implementado de forma adequada 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 csv's de 3 colunas.
  - Sorteio: dados aleátorios são gerados, somente com o controle de serem gerados dois conjuntos linearmente separáveis.
- Botão de 'TREINAR' para que o algoritmo comece a rodar, ao final do algorimo 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) 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.

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. click do usuário no botão 'TREINAR'
4. plotagem das retas que o algoritmo encontrou durante a execução
5. click no botão 'TESTAR' para que o conjunto de teste seja plotado e o algoritmo possa ser avaliado pelo usuário.

O botão de 'REINICIAR' pode ser usado para que o algoritmo refaça o treinamento com os mesmos dados, mas com a possibilidade de trocar os parâmetros.

4 Cronograma Proposto

Na tabela a seguir estão listadas as tarefas a serem 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.

5 O que já foi feito

- **Estudo e implementação do algoritmo perceptron em Python**

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

```
1  def train(self,data):
2    """
3     trains all the vector in data.
4     Every vector in data must have three elements,
5     the third element (x[2]) must be the label (desired output)
6    """
7    learned = False
8    iteration = 0
9    while not learned:
10    globalError = 0.0
11    for x in data: # for each sample
12     r = self.response(x)
13     if x[2] != r: # if we have a wrong response
14      iterError = x[2] - r # desired response - actual response
15      self.updateWeights(x,iterError)
16      globalError += abs(iterError)
17    iteration += 1
18    if globalError == 0.0 or iteration >= 100: # stop criteria
19     print 'iterations',iteration
20     learned = True # stop learning
```

  Nesse algoritmo limitamos o número máximo de iterações em 100 e η para efeito de teste.

- **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 exportacão 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

- Implementar 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 e a interface com o usuário já está implementada para suportar sequências de cliques errados.

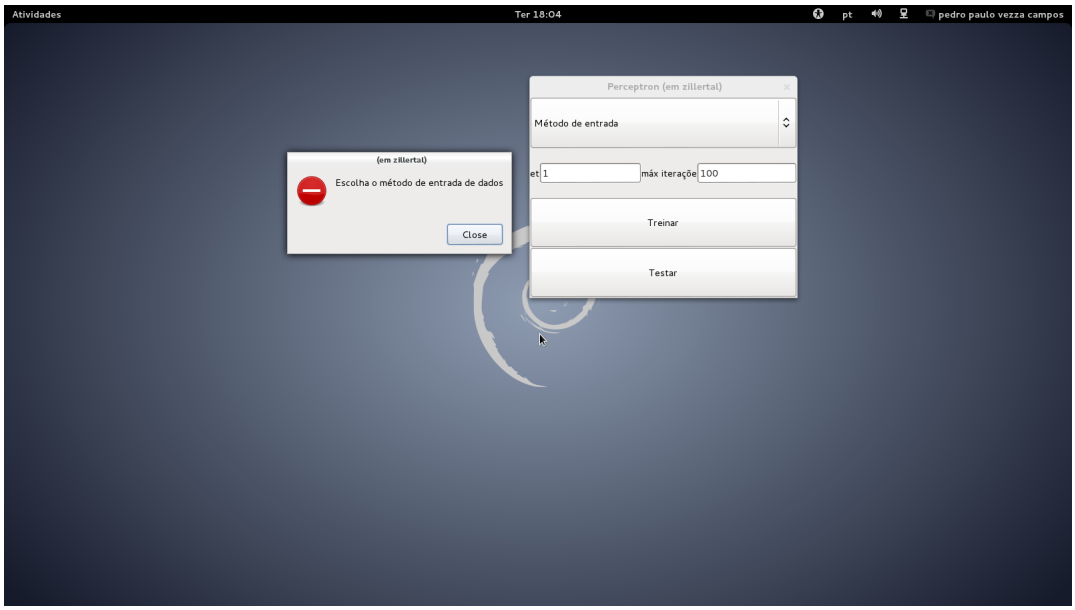

Figura 1: Interface com o usuário.

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

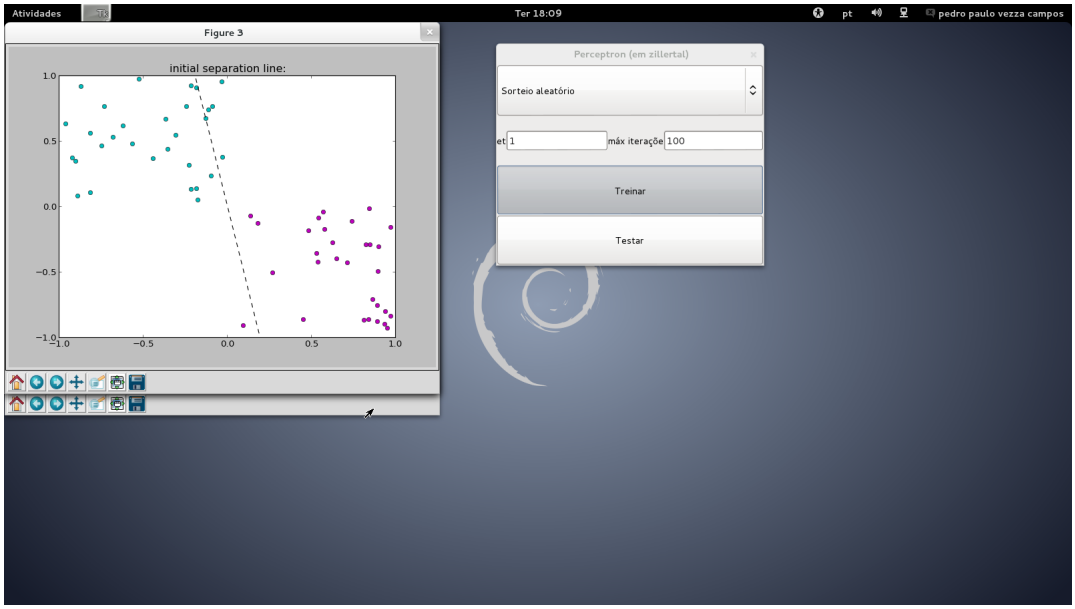

Figura 2: Situação inicial do exemplo 1, onde uma reta arbitrária foi sorteada.

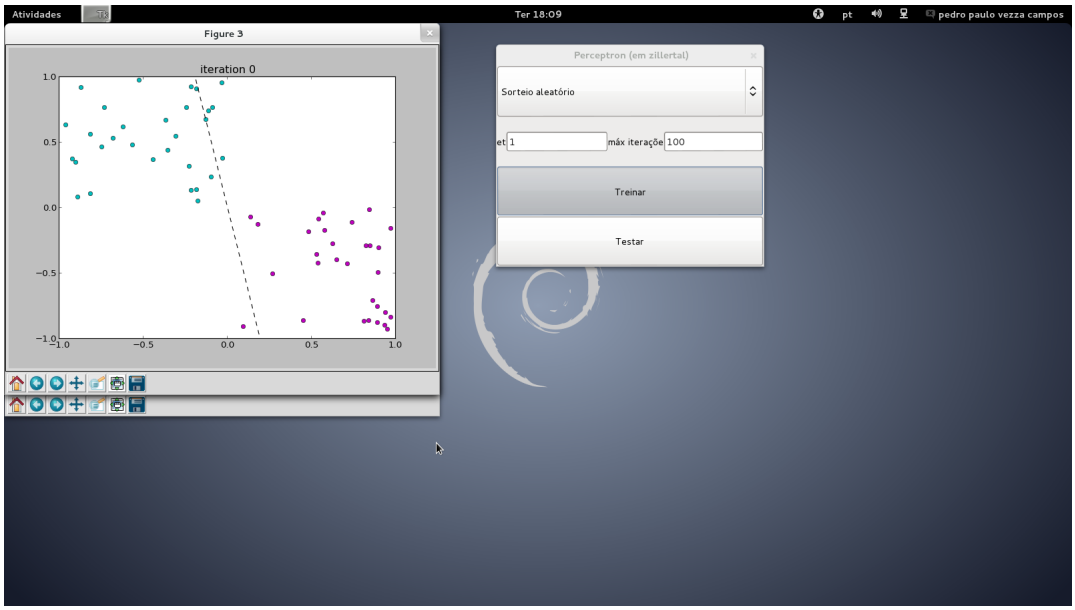

Figura 3: Começa a execução do algoritmo para o exemplo 1.

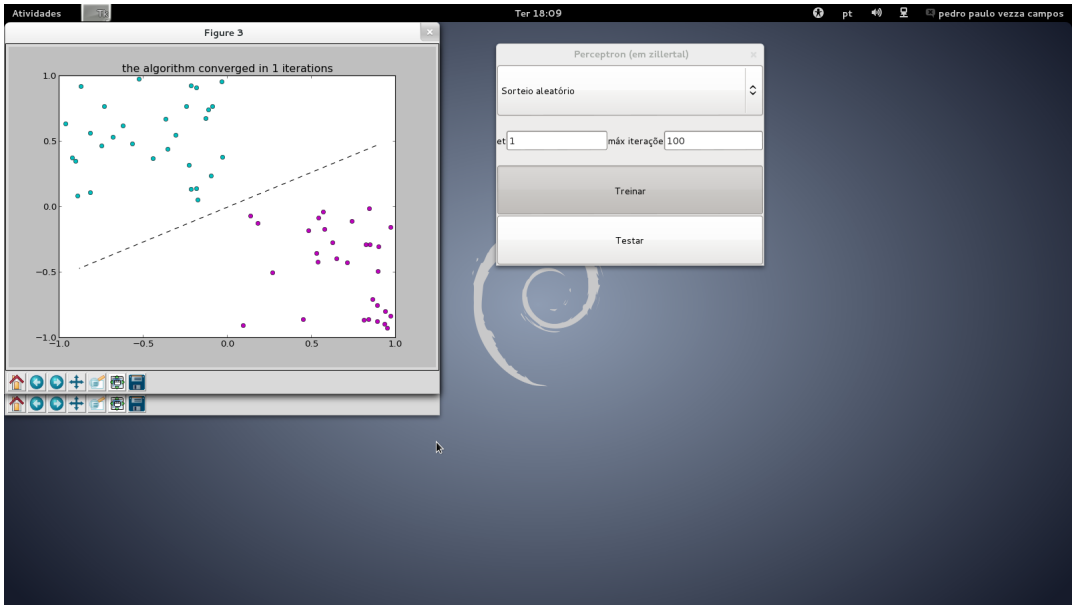

Figura 4: Fim do algoritmo, reta classificadora foi encontrada no exemplo 1.

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.

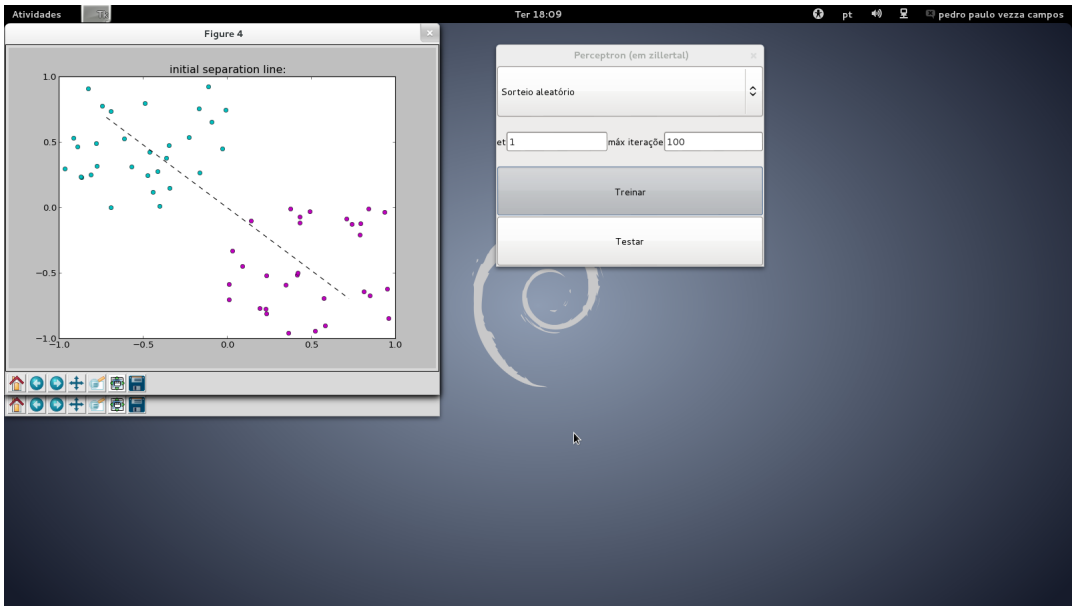

Figura 5: Situação inicial do exemplo 2, onde uma reta arbitrária foi sorteada.

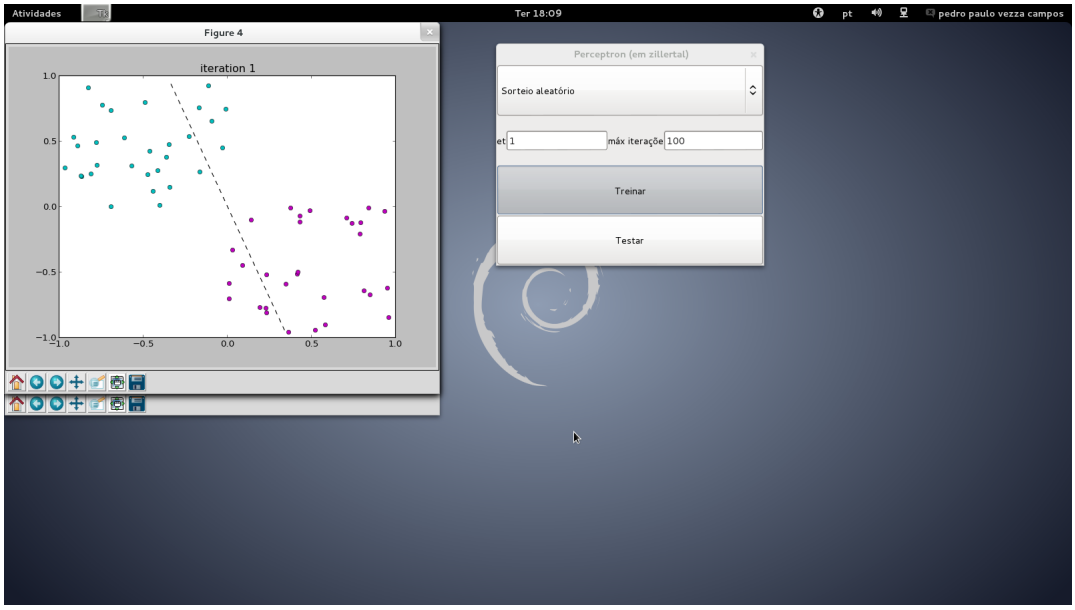

Figura 6: Começa a execução do algoritmo para o exemplo 2.

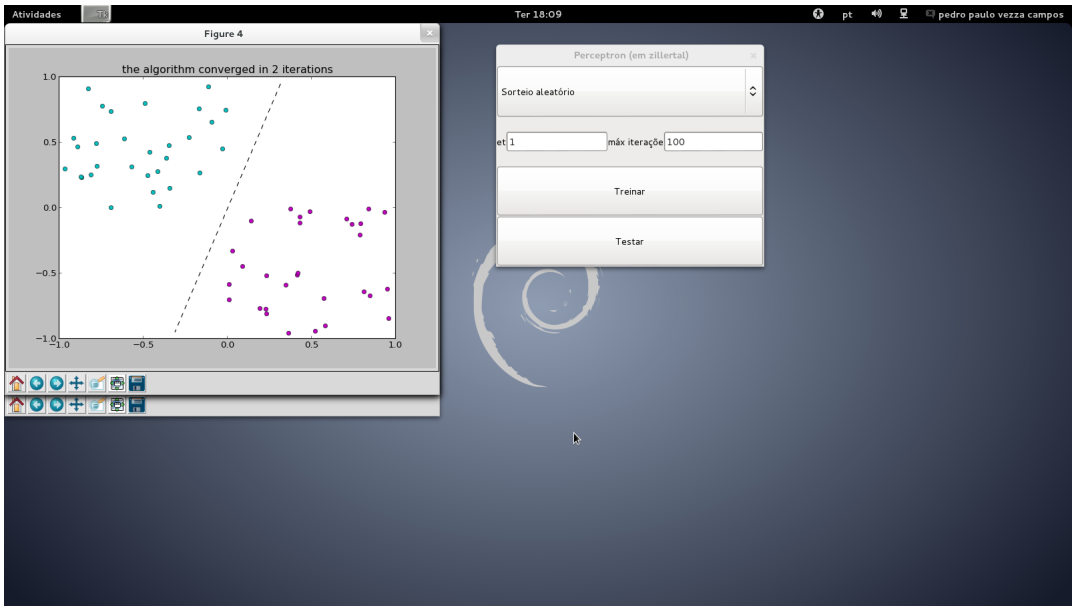

Figura 7: Fim do algoritmo, reta classificadora foi encontrada para o exemplo 2.

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:

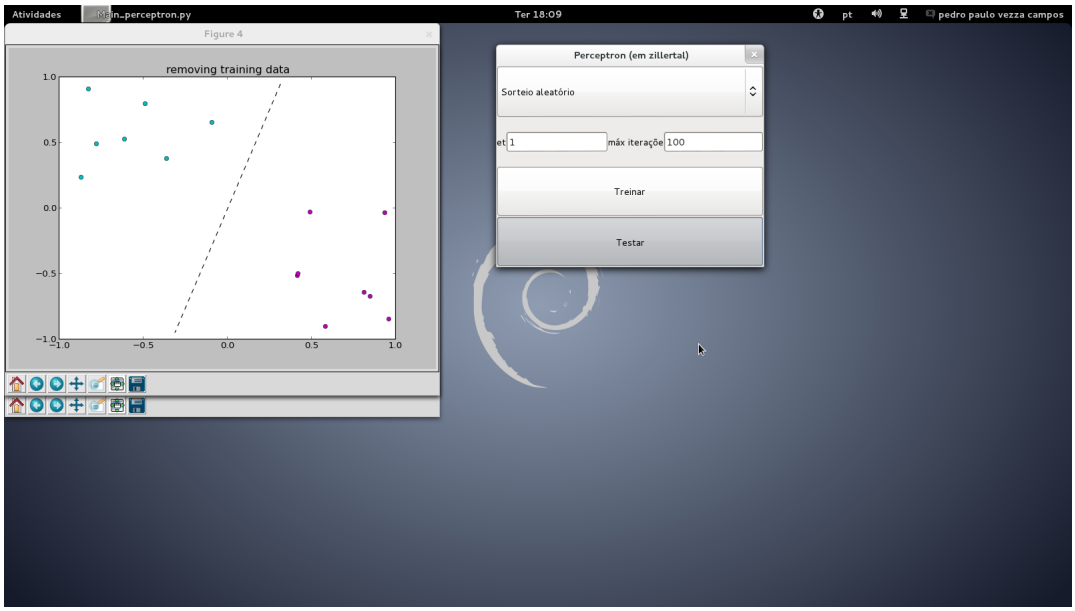

Figura 8: Remoção dos dados de treino.

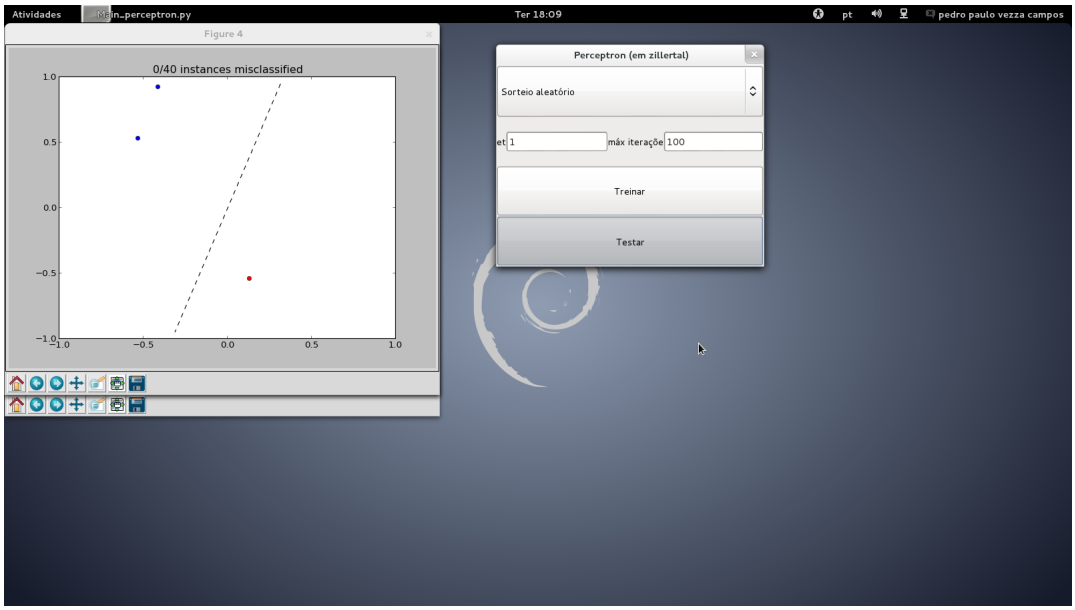

Figura 9: Exibição dos dados de teste.

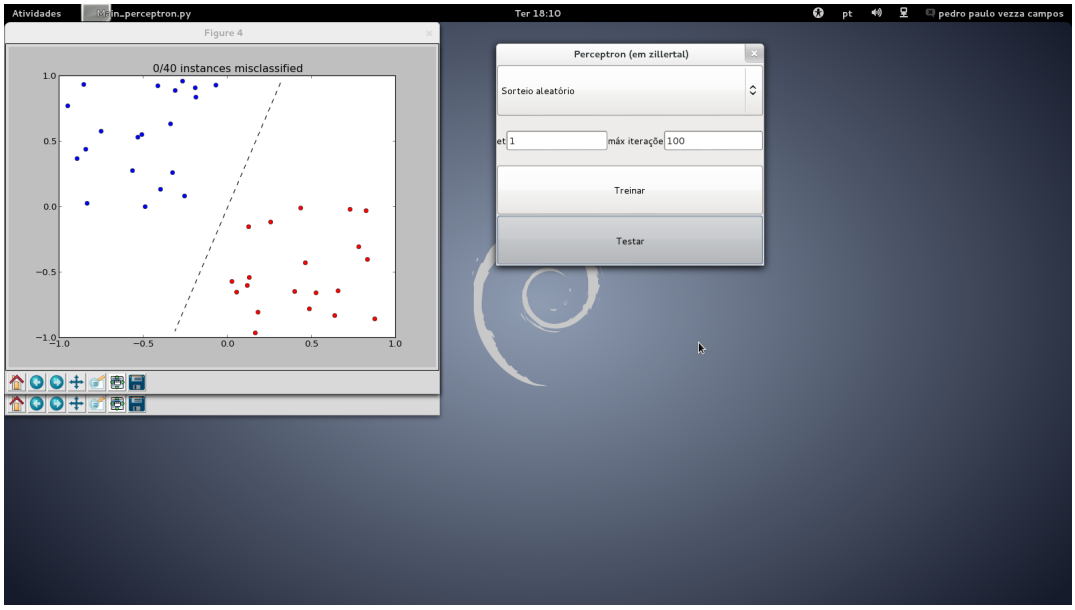

Figura 10: Fim do teste para o exemplo 2.

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

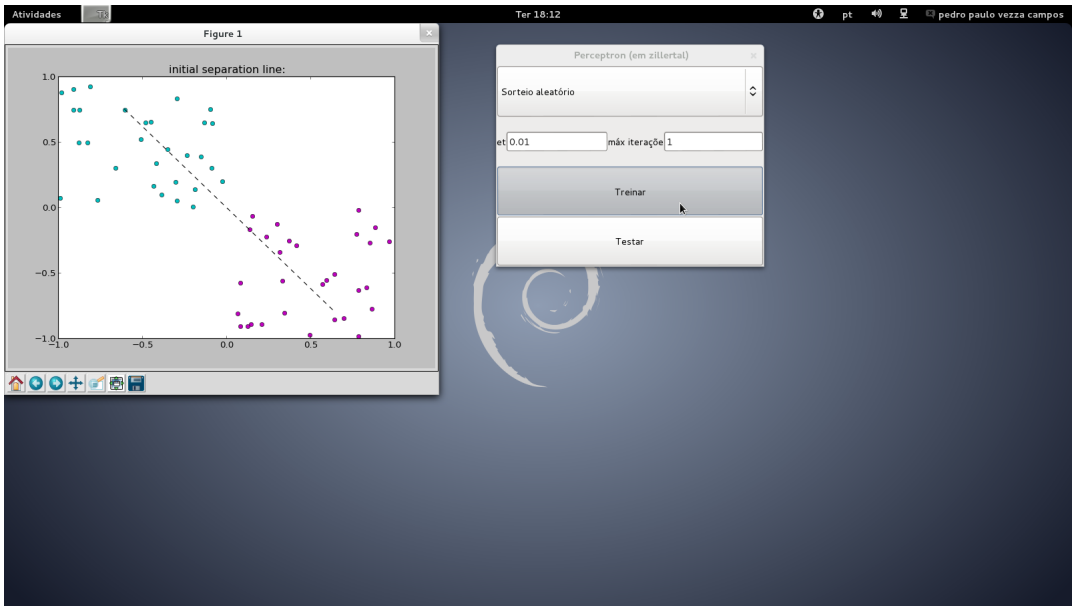

Figura 11: Exibição dos dados de teste.

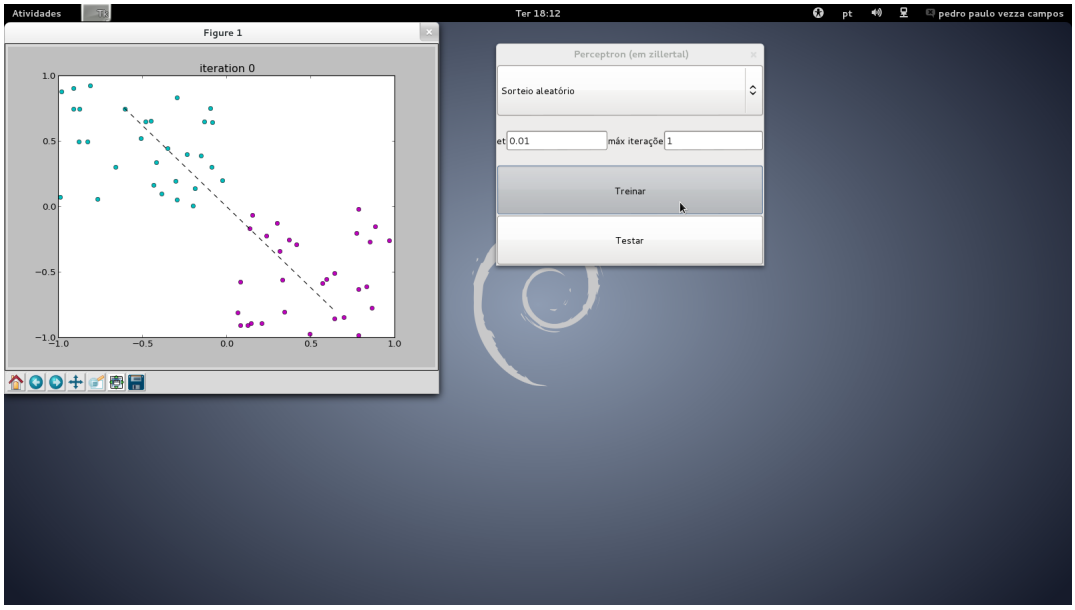

Figura 12: Fim do teste para o exemplo 2.

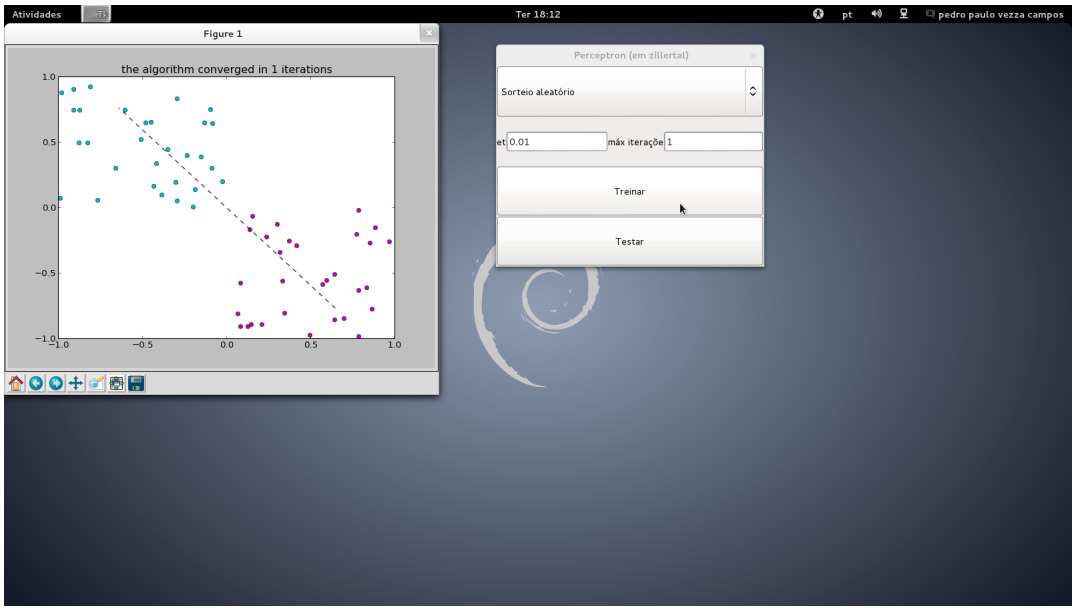

Figura 13: Exibição dos dados de teste.

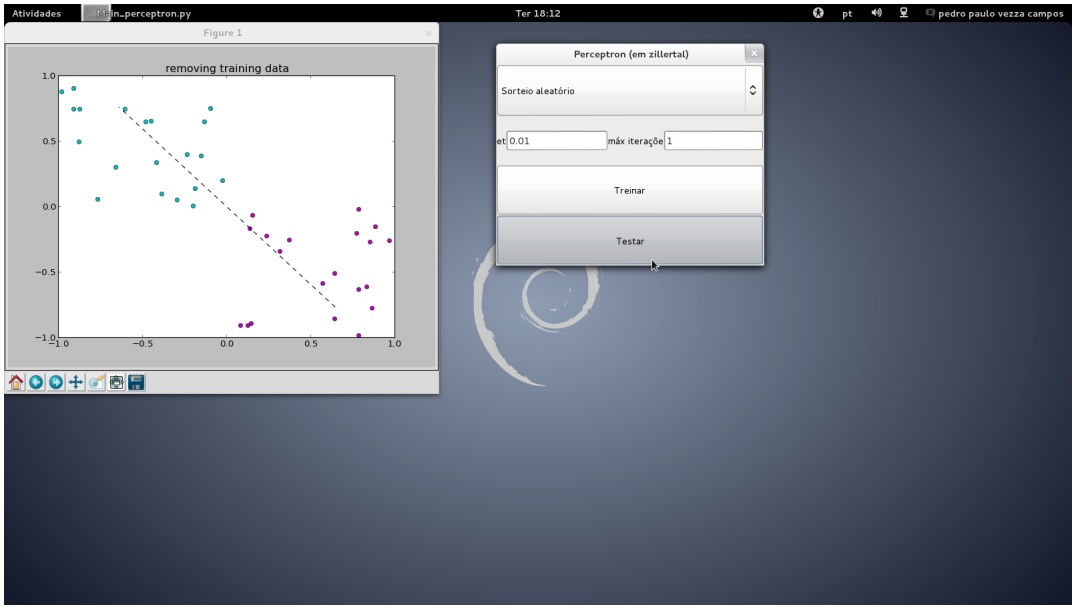

Figura 14: Fim do teste para o exemplo 2.

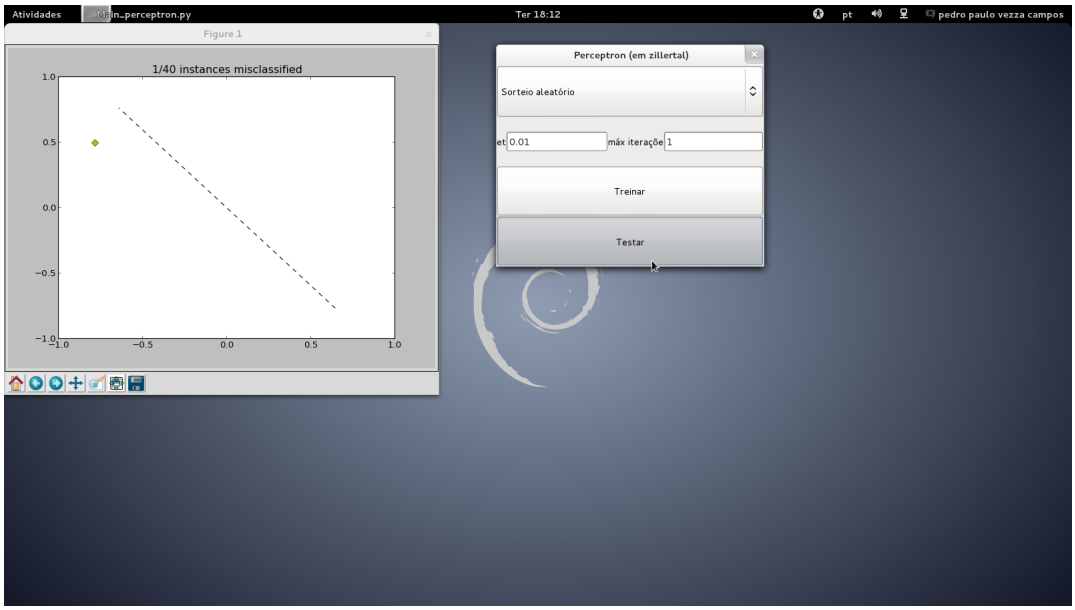

Figura 15: Exibição dos dados de teste.

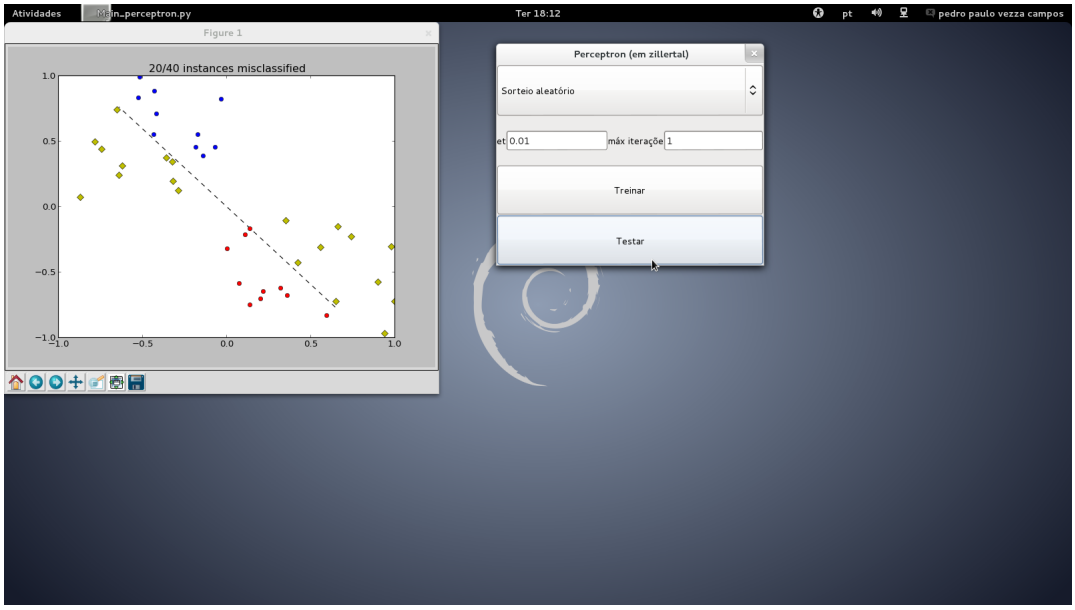

Figura 16: Fim do teste para o exemplo 3.

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

É possível também realizar a entrada de dados por arquivos externos. Devem ser produzidos e carregados dois arquivos separados um de treinamento e outro de teste, ambos no formato CSV com o seguinte padrão:

`x1, x2, classe`, onde a classe é 1 ou -1.

6 O que falta ser feito

- Estudo e implementação do algoritmo perceptron em Python

  Falta a implementação do algoritmo para mais de duas dimensões.

- Implementar interface gráfica

  Falta a implementação da interface com o usuário que permitirá a seleção do método de entrada por clique.

- Escrever Relatório Final

  A redação do relatório final será iniciada em novembro conforme consta no cronograma.

7 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/