Aluno: Cleiton Moya de Almeida

Relatório do Trabalho Prático

CPS844 - Inteligência Computacional I

Professores:
Carlos Eduardo Pedreira
Carolina Marcelino

Rio de Janeiro, 15 de maio de 2020

Sumário

1 O Algoritmo de Aprendizagem Perceptron

1.1 Introdução

O *Perceptron Learning Algorithm* (PLA) é um algoritmo de classificação linear. A ideia básica do algoritmo é estabelecer pesos diferentes (w_i) para cada uma das d coordenadas de $\mathbf{x} \in \mathbb{R}^d$ (x_i), e então combiná-los linearmente, gerando uma "pontuação". Esta pontuação é comparada com um *bias* (b) por uma função de avaliação, resultando na hipótese $h(\mathbf{x})$ [1]:

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + b\right) \tag{1}$$

Na equação acima, estamos assumindo um espaço de saída $\mathcal{Y} = \{+1, -1\}$ binário e utilizando sign(x) como função de avaliação.

Para facilitar a manipulação algébrica e implementação do algoritmo, podemos re-escrever a equação 1 na forma matricial

$$h(\mathbf{x}) = \text{sign}\left(\mathbf{w}^\intercal \mathbf{x}\right) \tag{2}$$

fixando uma nova coordenada $x_0 = 1$ em $\mathbf{x}$ e fazendo $w_0 = b$.

O algoritmo pode ser descrito e implementado com os seguintes passos:

1. Inicialize o vetor de pesos $\mathbf{w}$ com valores arbitrários, gerando uma hipótese inicial $h_0(\mathbf{x})$;
2. Escolha um exemplar $(\mathbf{x}_n, y_n)$ mal-classificado pela hipótese;
3. Atualize o vetor de pesos pela regra:
$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + y(t)\mathbf{x}(t);$$
4. O exemplar $(\mathbf{x}_n, y_n)$ agora será classificado corretamente pela nova hipótese;
5. Repita os passos 2 a 5 até que todos os exemplares estejam corretamente classificados ou até que o erro da amostra E_{in} esteja dentro do limite tolerado.

Se os dados são linearmente separáveis, há garantia de convergência do algoritmo [1].

1.2 Versão "*Pocket* PLA"

No algoritmo PLA original, após uma nova iteração, pode ocorrer do erro da amostra ficar maior nesta nova iteração do que na anterior. Ou seja, pode ocorrer $E_{in}(t+1) > E_{in}(t)$. Assim, se não houver convergência do algoritmo, o mesmo pode terminar com uma hipótese final pior do que alguma outra intermediária.

No algoritmo "*Pocket* PLA", toda vez que (em alguma iteração t) se encontrar uma hipótese $\mathbf{w}$ que resulte em um menor E_{in}, esta hipótese é "colocada no bolso" , e a que anteriormente "estava no bolso" é descartada. Desta forma, ao final das iterações, o algoritmo seleciona como hipótese-final g aquela com o menor E_{in} (dentre todos $\mathbf{w}$ avaliados).

1.3 Experimentos e Resultados

Questão 1

Algoritmo PLA, $N = 10$, 1.000 experimentos.

O número médio de iterações requeridas para a convergência foi:

- $\bar{t} \approxeq 10$

Resposta: **b**

Questão 2

Algoritmo PLA, $N = 10$, 1.000 experimentos e 1.000 pontos de teste.

O erro fora da amostra foi:

- $\overline{E}_{out} \approxeq 0.1$;

Resposta: **c**

Questão 3

Algoritmo PLA, $N = 100$, 1.000 experimentos.

O número médio de iterações requeridas para a convergência foi:

- $\bar{t} \approxeq 100$;

Resposta: **b**

Questão 4

Algoritmo PLA, $N = 100$, 1.000 experimentos e 1.000 pontos de teste.

O erro fora da amostra foi:

- $\overline{E}_{out} \approxeq 0.01$;

Resposta: **b**

Questão 5

O gráfico da figura 1 mostra a reta da função f, a reta da hipótese final g, o conjunto de dados $\mathcal{D}$ e os 1.000 pontos fora da amostra para o cenário $N = 10$ (questões 1 e 2). Estes dados referem-se ao último experimento simulado.

A figura 2, por sua vez, mostra o gráfico com os dados do último experimento simulado para o cenário de $N = 100$ pontos (questões 3 e 4).

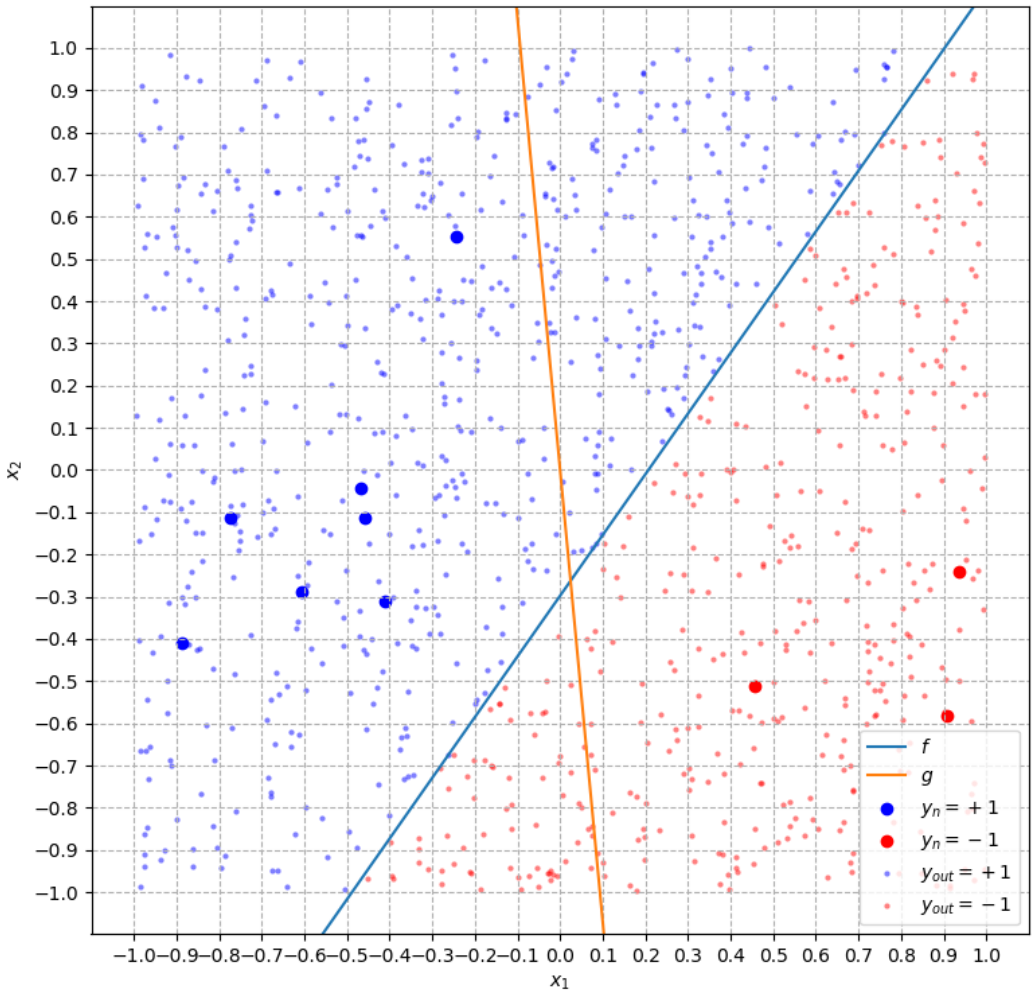

Figura 1: Gráfico das questões 1 e 2

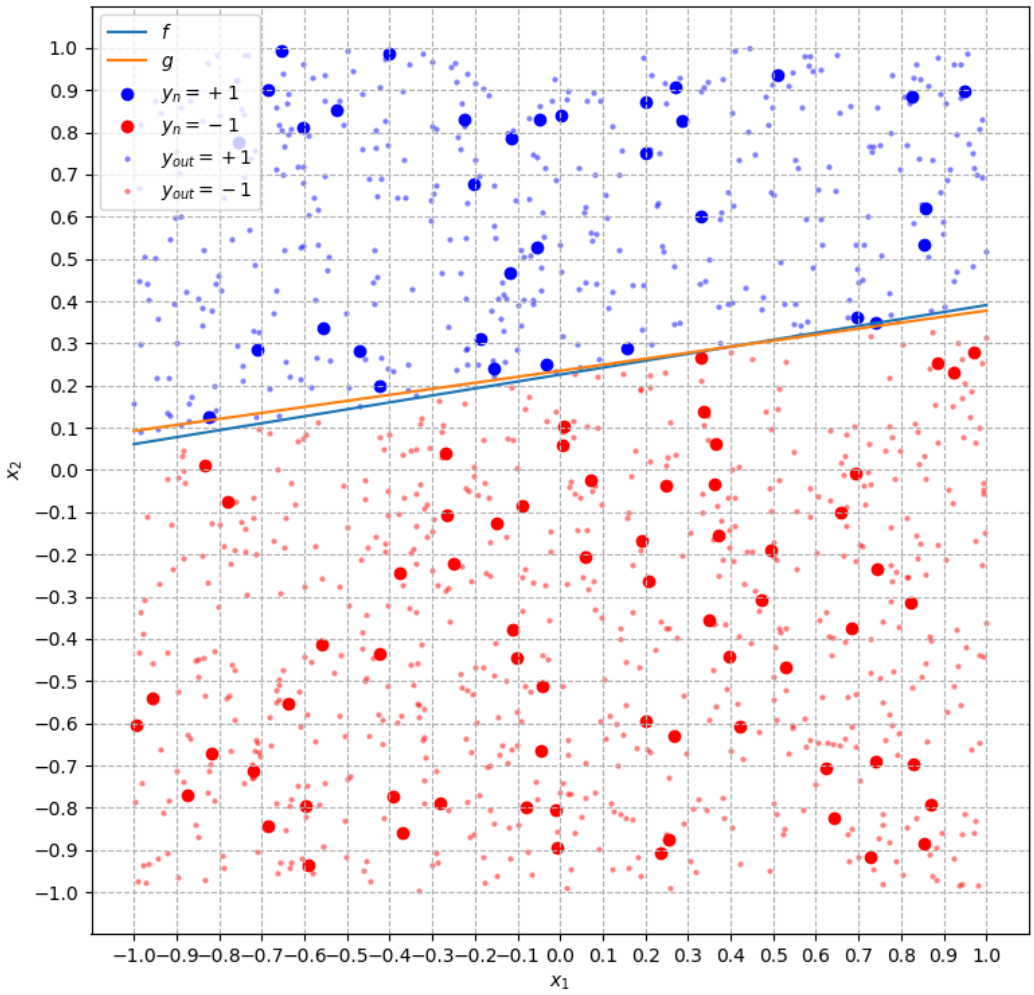

Figura 2: Gráfico das questões 3 e 4

1.4 Análise e Conclusões

Com relação à convergência, foi possível observar que, para um cenário de N dados, em média o número de iterações necessárias é da mesma ordem de N (considerando o caso em que inicialmente todos os dados estavam mal-classificados). Isso ocorre porque o algoritmo *Perceptron* ajusta o vetor $\mathbf{w}$ para cada ponto mal-classificado, até a eventual convergência.

Como todos os dados utilizados nos experimentos são linearmente separáveis, em todos os experimentos o algoritmo convergiu, ainda que ao custo de um elevado número de iterações para alguns cenários específicos.

Também observou-se que erro médio esperado fora da amostra ($\overline{E}_{out}$) é inversamente proporcional ao número de dados, ou seja, quanto maior o número de dados, menor tende ser E_{out}. Isto pode ser observado graficamente: quanto menor o número de dados, maior o "espaço" para a reta de g divergir de f e mesmo assim termos $E_{in} = 0$ (Figura 1). Por outro lado, quando o número de dados é grande, há pouco "espaço" para que esta divergência ocorra (Figura 1).

2 Regressão Linear

2.1 Introdução

No problema de classificação, a função-alvo $f(\mathbf{x}) = y$ tipicamente assume valores discretos (por exemplo, aprovar ou não o crédito para um cliente bancário). Já no problema de regressão, geralmente lidamos com funções-alvo para as quais $y \in \mathbb{R}$. Entretanto, dado que na regressão y pode assumir qualquer valor real, y também pode assumir valores discretos (ex.: $\mathcal{Y} = \{-1, +1\}$). Ou seja, podemos também utilizar o algoritmo de regressão em um problema de classificação, como explorado neste trabalho.

O algoritmo de regressão é baseado na minimização do erro quadrático entre $h(\mathbf{x})$ e y^2. Como não temos como computar E_{out}, definimos este erro em termos de E_{in} [1]:

$$E_{in}(h) = \frac{1}{N} \left(h(\mathbf{x}) - y_n\right)^2.$$

Na regressão linear, h assume a forma uma de combinação linear dos componentes de $\mathbf{x}$:

$$h(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i = \mathbf{w}^\intercal \mathbf{x}.$$

Com alguma manipulação algébrica, podemos escrever

$$E_{in}(h) = \frac{1}{N} ||X\mathbf{w} - \mathbf{y}||^2$$

e o problema de regressão se resume a encontrar um $\mathbf{w}_{lin}$ que minimize E_{in}.

A solução deste problema de otimização é dada por [1]:

$$\mathbf{w}_{lin} = X^\dagger \mathbf{y},$$

onde $X^\dagger$ é pseudo-inversa de X.

2.1.1 Uso de regressão no problema de classificação

O algoritmo de regressão linear pode ser usado no problema de classificação de duas formas distintas:

1. Utilizar como hipótese-final o vetor de pesos $\mathbf{w}_{lin}$ obtido pelo algoritmo de regressão (questões 6 e 7); ou

2. Utilizar $\mathbf{w}_{lin}$ como vetor de pesos inicial no algoritmo *Perceptron* (questões 8 e 9).

2.2 Resultados

Questão 6

Classificação por regressão linear.
Para $N = 100$, tivemos $\overline{E}_{in} \approxeq 0.04 \approxeq 0.01$.
Resposta: **c**

Questão 7

Classificação por regressão linear.
Para $N = 100$, tivemos $\overline{E}_{out} \approxeq 0.05 \approxeq 0.01$.
A figura 3 mostra o gráfico da simulação do primeiro experimento ($N = 1$).
Resposta: **c**

Questão 8

Algoritmo PLA com vetor $\mathbf{w}$ inicializado por regressão linear.
Para $N = 10$, a média de convergência em 1.000 experimentos fo de $\bar{t} \approxeq 5$;
A figura 4 mostra o gráfico da simulação do último experimento.
Resposta: **a**

Questão 9

Algoritmo *Pocket PLA* com ruído nos dados de treinamento (10%) e com o vetor $\mathbf{w} = 0$ (cenários a e b) *versus* $\mathbf{w}$ inicializado por regressão linear (c e d).

(a) $\mathbf{w}_0 = 0$, $i = 10$, $N1 = 100$, $N2 = 1.000$.
$\overline{E}_{out} \approxeq 0.1$.
Figura 5.

(b) $\mathbf{w}_0 = 0$, $i = 50$, $N1 = 100$, $N2 = 1.000$.
$\overline{E}_{out} \approxeq 0.06$.
Figura 6.

(c) $\mathbf{w}_0$ inicializado por regressão, $i = 10$, $N1 = 100$, $N2 = 1.000$.
$\overline{E}_{out} \approxeq 0.09$.
Figura 7.

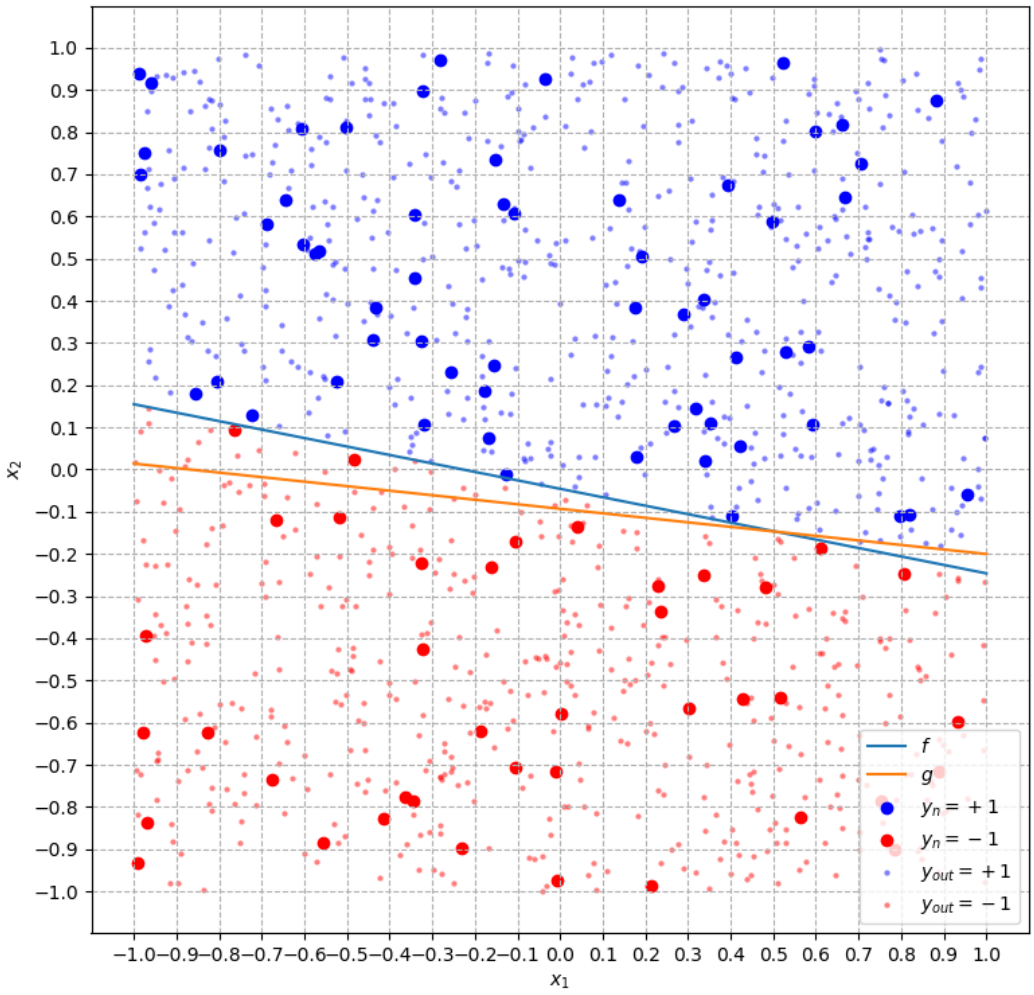

Figura 3: Gráfico das questões 6 e 7

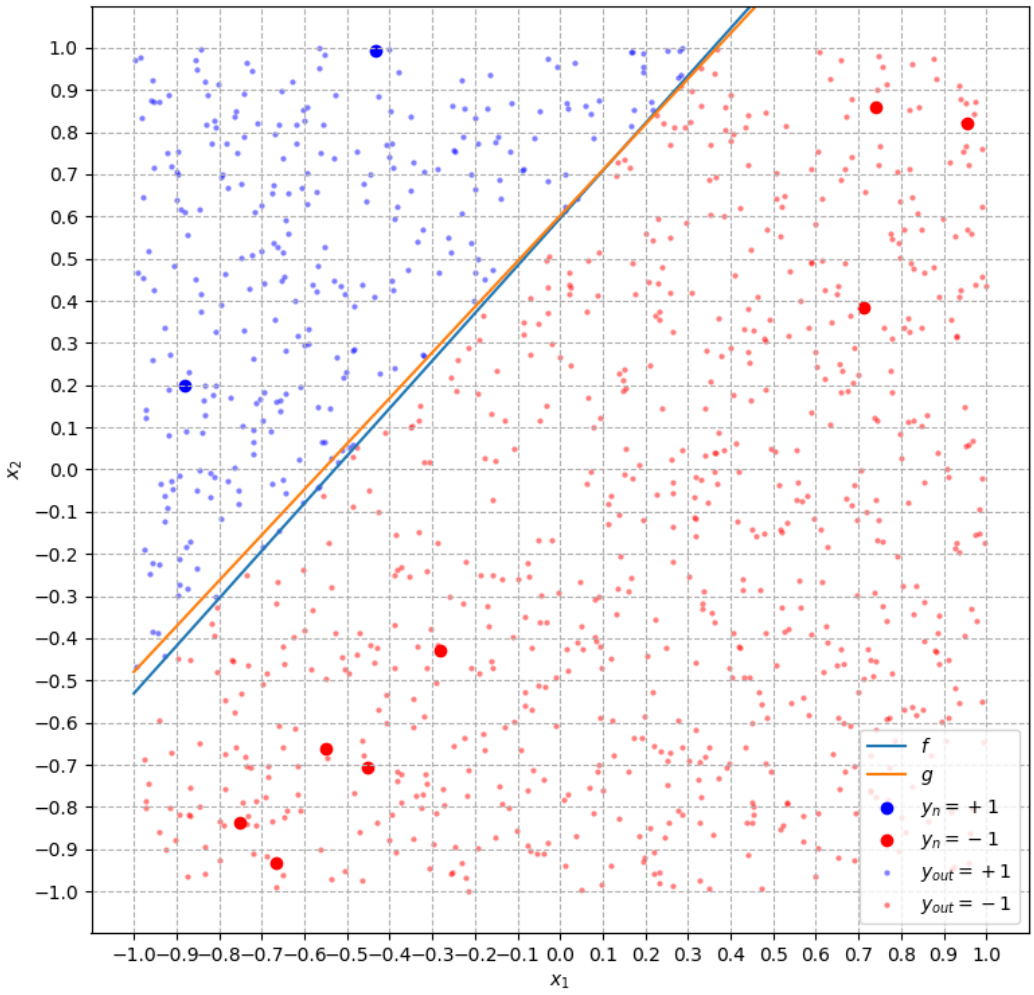

Figura 4: Gráfico da questão 8

(d) $\mathbf{w}_0$ inicializado por regressão, $i = 50$, $N1 = 100$, $N2 = 1.000$.

$\overline{E}_{out} \approxeq 0.05$.

Figura 8.

2.3 Análise e Conclusões

Comparando o desempenho em termos de E_{out}, observou-se que a classificação por regressão linear mostrou desempenho semelhante (ligeiramente inferior) ao algoritmo PLA para o caso de $N = 100$ (questões 2 e 7).

Em termos de número iterações necessárias para a convergência do PLA, observou-se na questão 8 que, ao inicializar o vetor $\mathbf{w}$ com a hipótese retornada pelo algoritmo o PLA convergiu de maneira mais rápida do que utilizar $\mathbf{w} = 0$.

Na questão 9, introduziu-se ruído nos dados de treinamento e estes deixaram de ser linearmente separáveis. Neste caso, o algoritmo PLA não converge, por isso é necessário utilizar o algoritmo *Pocket* PLA.

Comparou-se então, na questão 9, o desempenho do algoritmo *Pocket* PLA nos cenários de $\mathbf{w}_0 = 0$ e $\mathbf{w}_0$ inicializado com regressão linear. Nesta questão, também observou-se desempenho semelhante (em termos de E_{out}) em ambos cenários.

3 Transformação Não-Linear

3.1 Introdução

Quando os dados de treinamento $\mathbf{x}_n \in \mathcal{X}$ não são linearmente separáveis e é inviável utilizar o modelo linear, um possível artifício (quando viável) é realizar uma transformação não-linear a qual leva os dados a um espaço $\mathcal{Z}$, tal que $\mathbf{z}_n \in \mathcal{Z}$ passam a ser linearmente separáveis. Pode-se então realizar a separação dos dados em $\mathcal{Z}$ e depois transportar a hipótese $\tilde{g}(\mathbf{z})$ para o espaço $\mathcal{X}$.

Os passos podem-se ser resumidos em [1]:

1. Encontrar uma transformação não-linear Φ que torne os dados linearmente separáveis em $\mathcal{Z}$;
2. Aplicar a transformação: $\mathbf{z}_n = \Phi(\mathbf{x}_n) \in \mathcal{Z}$;
3. Separar os dados em $\mathcal{Z}$: $\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^\intercal \mathbf{z})$;
4. Classificar no espaço $\mathcal{X}$: $g(\mathbf{x}) = \tilde{g}(\Phi(\mathbf{x})) = \text{sign}(\tilde{\mathbf{w}}^\intercal \Phi(\mathbf{x}))$.

É interessante observar que cada vetor $\mathbf{x}$ é transformado em um novo vetor $\mathbf{z}$. Entretanto, os dados em $\mathcal{Y}$ não são transformados:

$$(\mathbf{x}_n, y_n) \overset{\Phi}{\mapsto} (\mathbf{z}_n, y_n).$$

Desta forma, a classificação dos dados em $\mathcal{Z}$, utilizando a hipótese $\tilde{g}(\mathbf{z})$, produz o mesmo y_n que a classificação no espaço $\mathcal{X}$, usando $g(\mathbf{x})$. Outro ponto interessante é que o vetor de pesos ($\tilde{\mathbf{w}}$) é calculado somente no espaço $\mathcal{Z}$.

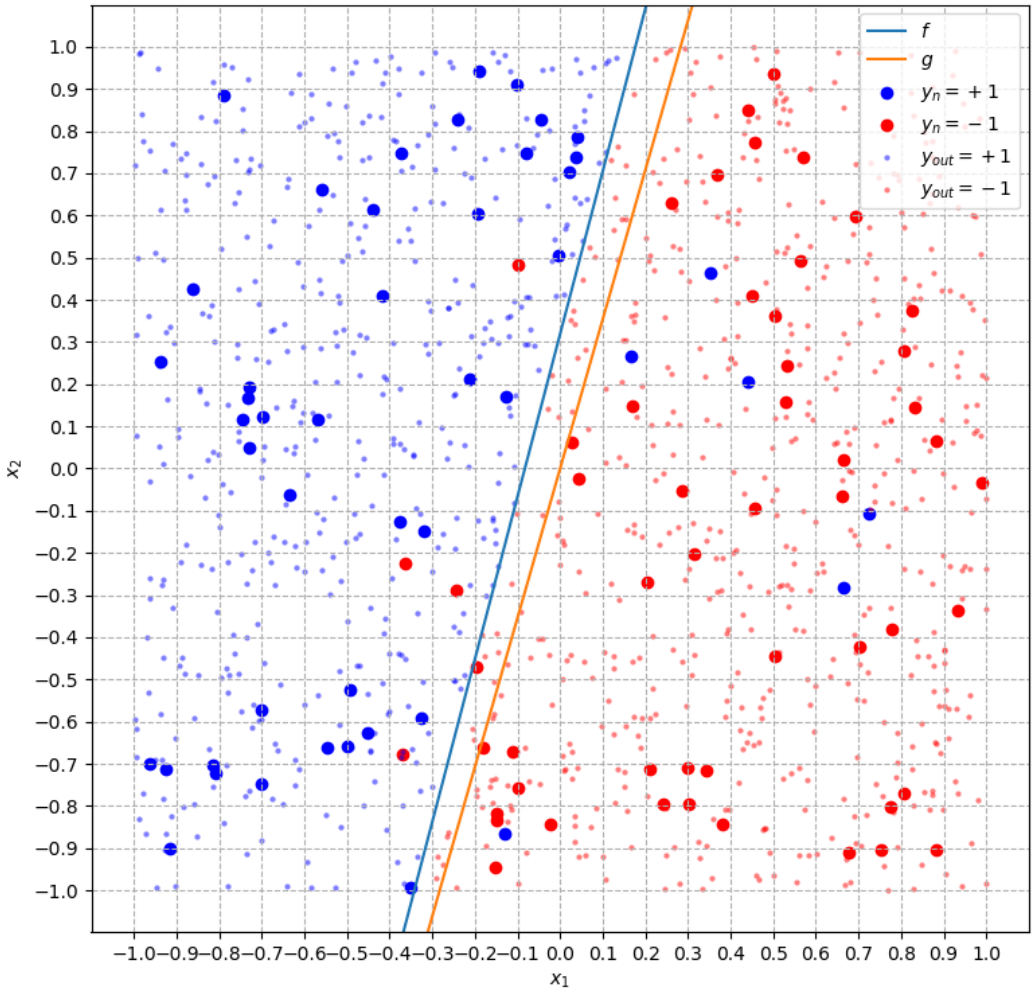

Figura 5: Gráfico da questão 9(a)

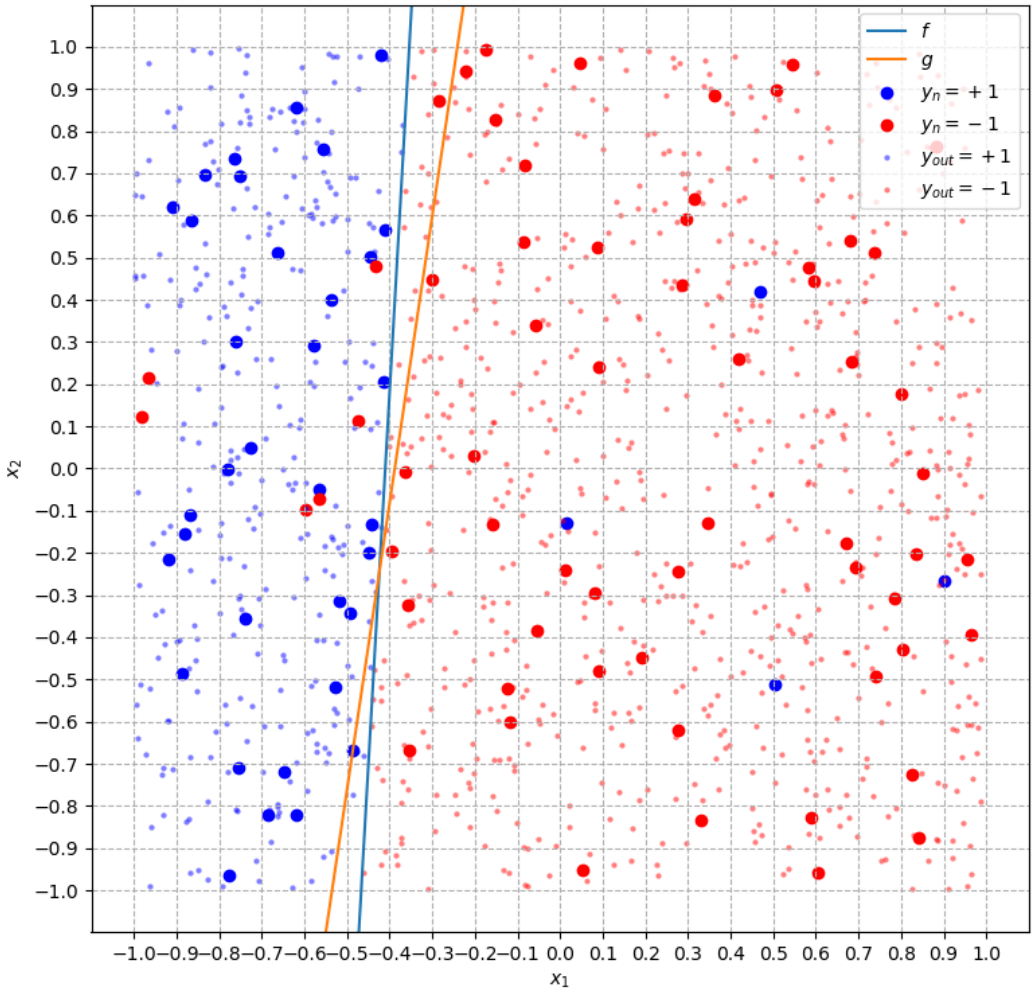

Figura 6: Gráfico da questão 9(b)

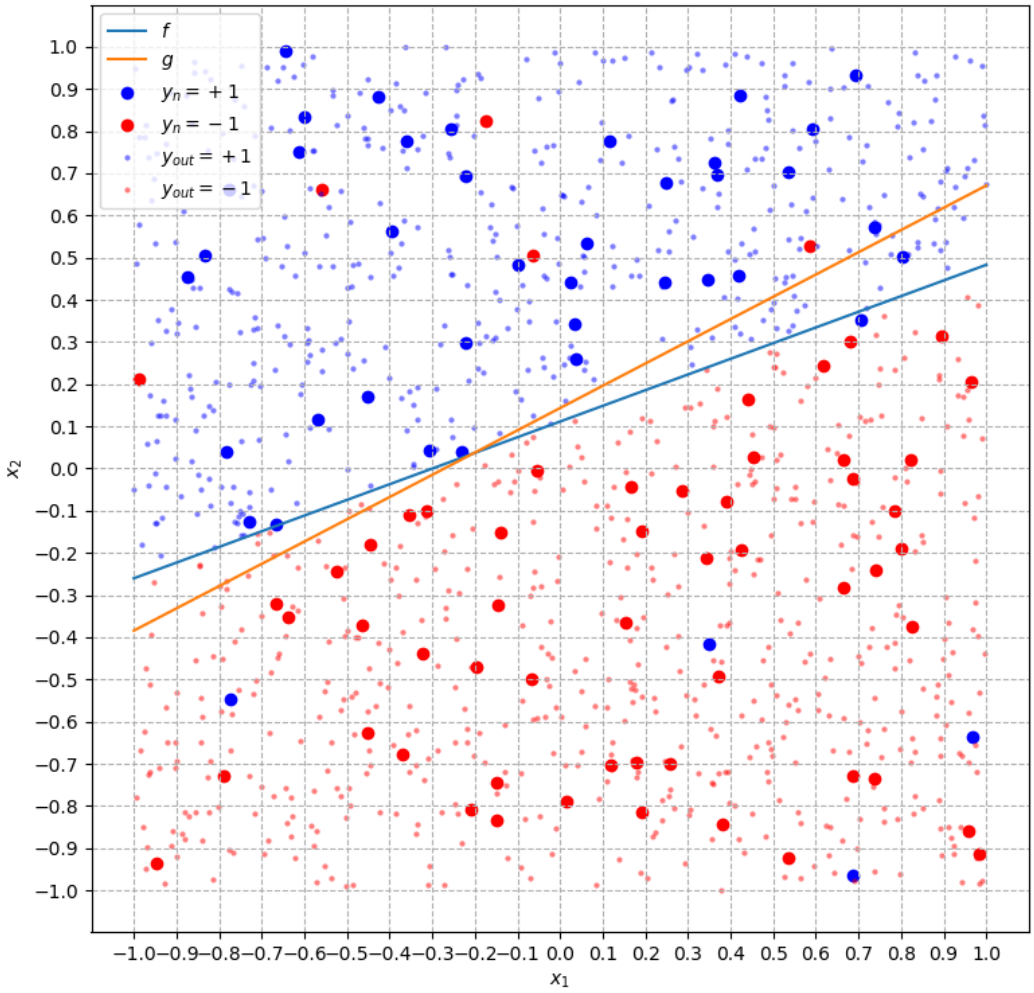

Figura 7: Gráfico da questão 9(c)

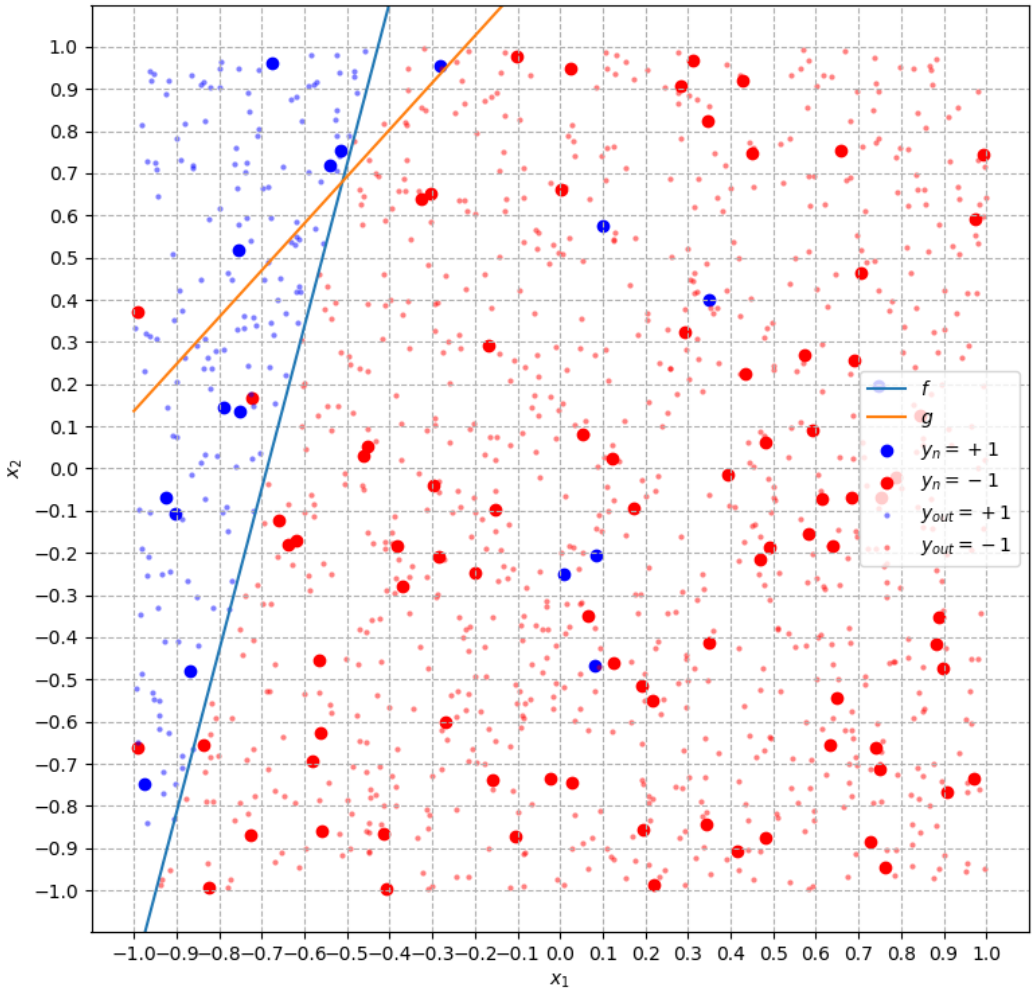

Figura 8: Gráfico da questão 9(d)

3.2 Resultados

Questão 10

Classificação por regressão linear, função-alvo não-linear (sem transformação).

- Função-alvo: $f(x_1, x_2) = \text{sign}(x_1^2 + x_2^2 - 0.6)$;
- $N = 1.000$ dados de treinamento.
- 1.000 execuções;

O erro dentro da amostra encontrado foi:

- $\overline{E}_{in}$: 0.5.

O gráfico figura 9 mostra os dados deste cenário;
Resposta: **d**

Questão 11

Regressão com transformação não-linear:

- Função-alvo: $f(x_1, x_2) = \text{sign}(x_1^2 + x_2^2 - 0.6)$;
- $N = 1.000$ dados de treinamento;
- Ruído em 10% dos dados de treinamento;
- 1.000 execuções;
- Vetor de atributos: $(1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$.

O vetor encontrado $\tilde{\mathbf{w}}$, com o o valor médio dos componentes avaliados em 1.000 execuções, foi:

$$\tilde{\mathbf{w}} = [-1, -0.003 + 0.0008 + 0.004 + 1.6 + 1.6]$$

A figura 10 mostra o gráfico gerado para o último experimento simulado.
Dentre os itens da resposta, a que mais se aproxima é:

$$\tilde{\mathbf{w}} = [-1, -0.05, +0.08, +0.13, +1.5, +1.5]$$

Resposta: **a**

Questão 12

- Função-alvo: $f(x_1, x_2) = \text{sign}(x_1^2 + x_2^2 - 0.6)$;
- $N = 1.000$ dados de treinamento.
- Ruído em 10% dos dados de treinamento;
- 1.000 execuções;
- Vetor de atributos: $(1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$.

- 1.000 pontos fora da amostra para cálculo de Eout;
- Ruído em 10% dos dados de deste.

Nesta questão, foram utilizados os mesmos parâmetros e função-alvo do execício anterior. O médio erro fora da amostra encontrado foi:

- $\overline{E}_{out} \approxeq 0.1$.

A figura 10 mostra o os dados do último experimento simulado.
Resposta: **b**

3.3 Análise e Conclusões

Primeiramente, observou-se na questão 10 que, se a função-alvo for não-linear e utilizar-se o modelo linear sem o artifício de transformação, os resultados encontrados podem ser bastante ruins.

Por outro lado, nas questões 11 e 12 verificou-se que, se for possível utilizar uma função de transformação não-linear que consiga separar os dados de modo satisfatório no espaço $\mathcal{X}$, então é possível obter excelentes resultados usando o modelo linear (mesmo que a função alvo seja não-linear e mesmo na presença de ruído nos dados de treinamento), o que é um resultado bastante interessante.

Observou-se ainda, na questão 12, qu o ruído nos dados de teste impactou diretamente a estimativa de E_{out}.

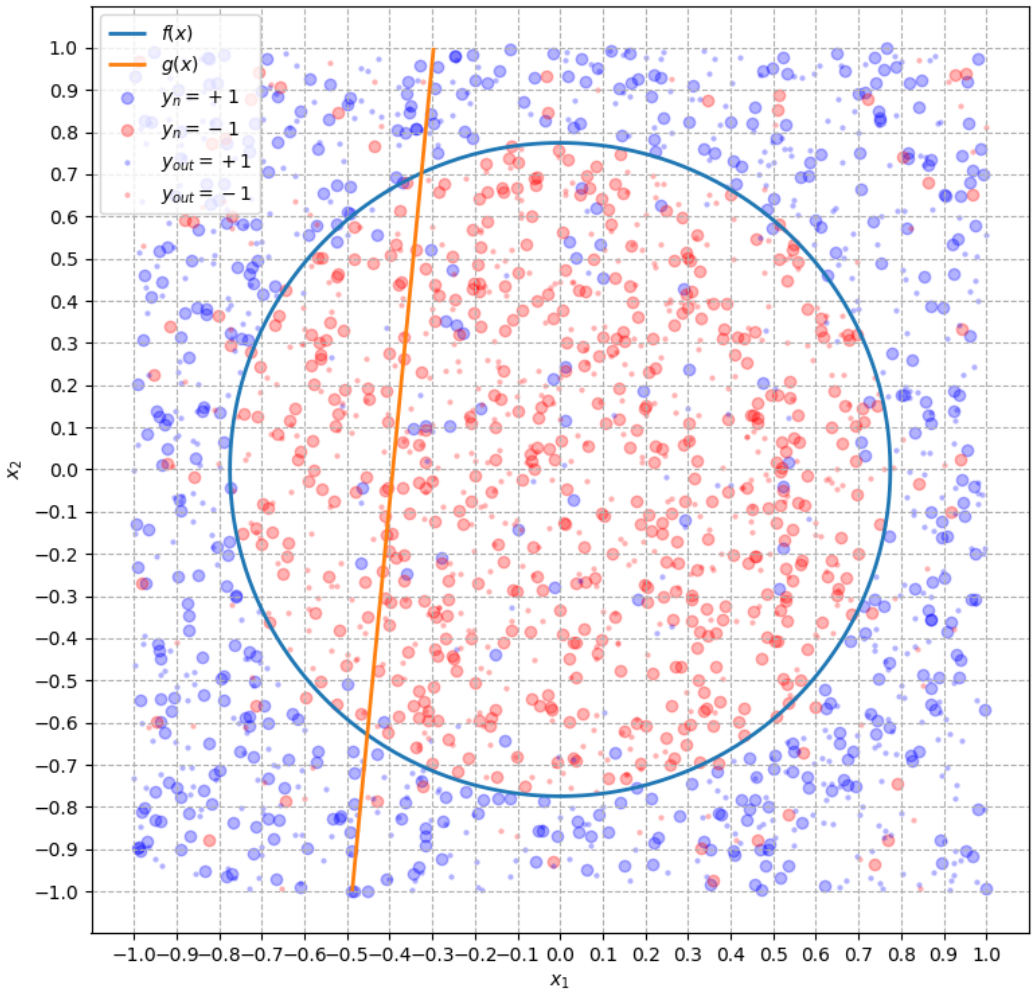

Figura 9: Gráfico da questão 10

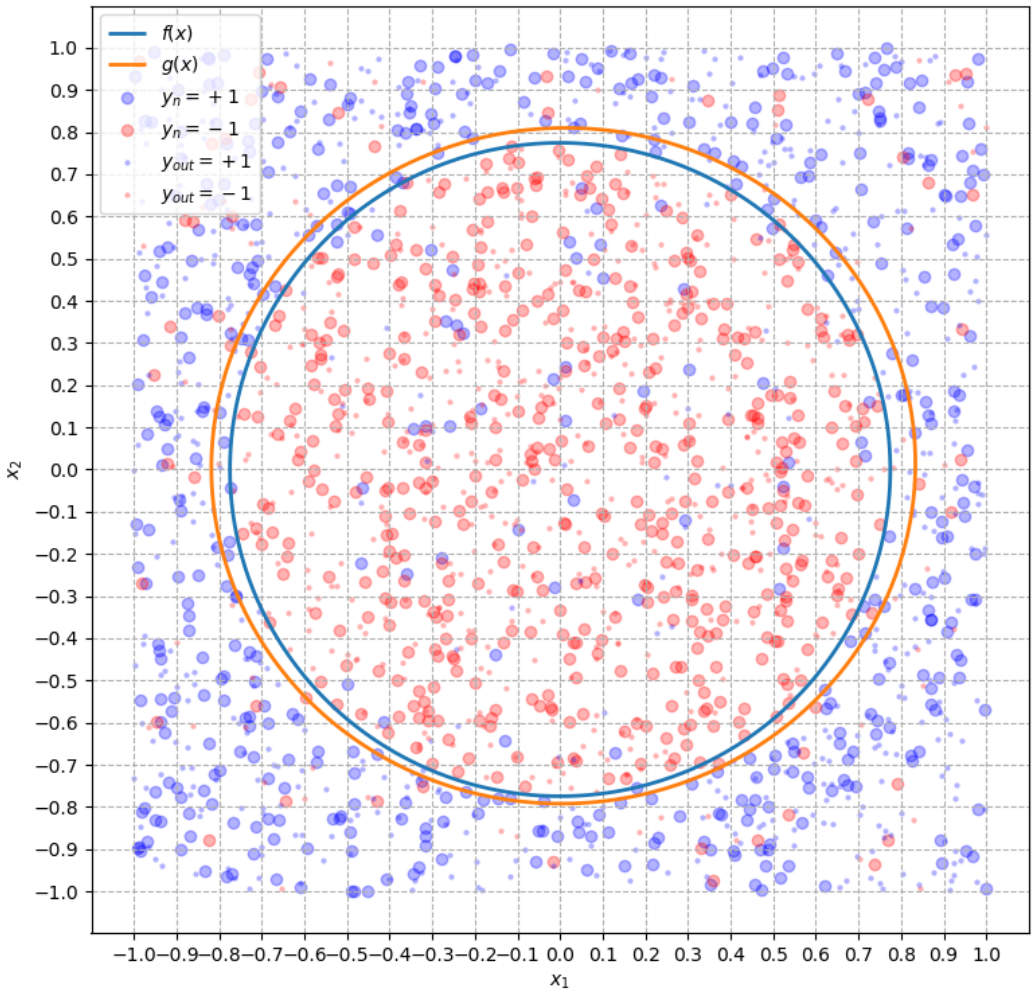

Figura 10: Gráfico das questões 11 e 12

A Implementação

Os algoritmos deste trabalho foram implementado na linguagem *Python* 3 utilizando como base os seguintes pacotes principais:

- *numpy*: para manipulação de vetores e matrizes;
- *matplotlib*: para geração de gráficos;
- *scipy.linalg*: para cálculo de matriz pseudo-inversa.

Para caso de necessidade de depuração ou reprodução, todos os experimentos e gráficos foram gerados com $seed = 3727339038$.

Os algoritmos foram implementados na relação de arquivos abaixo, anexados a este trabalho.

- PLA inicializado ou não com regressão:
  - Questões: 1, 2, 3, 4, 5, 8;
  - Arquivo: `pla.py`
- *Pocket* PLA inicializado ou não com regressão:
  - Questões: 9;
  - Arquivo: `pocket_pla.py`
- Classificação por regressão linear:
  - Questões: 6, 7;
  - Arquivo: `classif_reg_linear.py`
- Classificação por regressão com transforação não-linear:
  - Questões: 10, 11 12;
  - Arquivo: `classif_reg_nao_linear.py`

Referências

[1] Yaser S. Abu-Mostafa, Malik Magdon-Ismail e Hsuan-Tien Lin. *Learning from Data - A short Course*. AMLbook.com, 2012.