# Aprendizado de Máquina
## Lista 02

Ricardo Marra - 19/0137576

### I. Questão 01

Assume you want to design a classifier which receives univariate data as input and must choose between class +1 or class -1. The data associated with class "+1" follow a gaussian density with mean 2 and unit variance. On the other hand, the data associated with class "-1" follow a uniform density between -2 and 2. The prior probabilities are $P(C_{+1}) = 0.6$ and $P(C_{-1}) = 0.4$.

*A. Apply Bayes' methodology to obtain the optimal classification model and show, using a diagram, the model's architecture (with clear indication of the discriminant function(s)).*

Para o Método de Maximização a Posteriori, as funções discriminantes são definidas da seguinte maneira:

$$g_k(x) = p(x|C_k)P(C_k) \tag{1}$$

Para este caso, tem-se que:

$$p(x|C_{-1}) = \begin{cases} \frac{1}{4}, & \text{se } |x| \leq 2 \\ 0, & \text{c.c} \end{cases} \tag{2}$$

$$p(x|C_{+1}) = \mathcal{N}(2, 1) \tag{3}$$

Aplicando (2) e (3) em (1) tem-se as seguintes funções discriminantes:

$$g_{-1}(x) = \begin{cases} 0.1, & \text{se } |x| \leq 2 \\ 0, & \text{c.c} \end{cases} \tag{4}$$

$$g_{+1}(x) = 0.6 \times \mathcal{N}(2, 1) \tag{5}$$

As funções discriminantes podem ser observadas na Fig. 1.

*B. Calculate the model's decision boundary. Analytically calculate the classifier error probability (the integrals can be numerically solved).*

Para encontrar a região de decisão se faz:

$$g(x) = g_{-1}(x) - g_{+1}(x) \tag{6}$$

Tal que:

$$C_k = \begin{cases} C_{-1}, & \text{se } g(x) > 0 \\ C_{+1}, & \text{se } g(x) < 0 \end{cases} \tag{7}$$

Porém, os intervalos de decisão devem ser analisados devido ao fato de $g_{-1}(x)$ existir apenas no intervalo $-2 \leq x \leq 2$. Portanto, ao se aplicar (4) e (5) em (6), tem-se que:

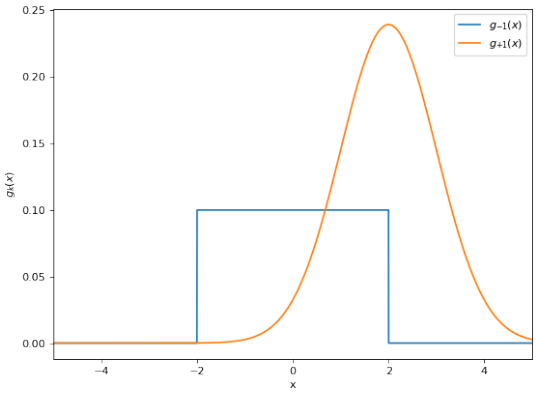


Fig. 1. Curvas das funções $g_{-1}(x)$ e $g_{+1}(x)$

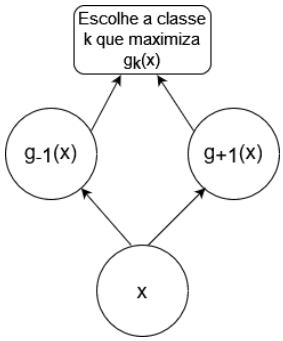


Fig. 2. Diagrama da arquitetura da Máquina de Bayes univariável.

$$g(x) = \begin{cases} 0.1 - 0.6\frac{e^{-\frac{1}{2}(x-2)^2}}{\sqrt{2\pi}}, & \text{se } |x| < 2 \\ -0.6\frac{e^{-\frac{1}{2}(x-2)^2}}{\sqrt{2\pi}}, & \text{c.c} \end{cases} \tag{8}$$

Dentro da região $|x| \leq 2$ existe uma transição entre as classes, onde $g_{+1}(x) > g_{-1}(x)$. Como é possível se observar na Fig. 1. Logo, dentro deste intervalo observa-se outra região de decisão, e para encontrá-la, é feito $g(x) = 0$. Fazendo isso e aplicando-se o logaritmo, tem-se:

$$-\frac{(x-2)^2}{2} = \log\left(\frac{\sqrt{2\pi}}{6}\right) \tag{9}$$

Após as devidas manipulações chega-se em:

$$x = 2 \pm \sqrt{-2\log\left(\frac{\sqrt{2\pi}}{6}\right)} \tag{10}$$

Assim, se determina as duas regiões de classificação

$$\begin{cases} x_1 = 3.321 \\ x_2 = 0.6788 \end{cases} \tag{11}$$

Como $p(x|C_{-1})$ é 0 fora do intervalo definido, caso $x > 2$, sempre será escolhida a classe $C_{+1}$. Portanto, tem-se definido que a região de decisão é separada pela linha em $x = 0.6788$.

Dessa forma, conclui-se que:

$$C_k = \begin{cases} C_{-1}, & \text{se } -2 \leq x \leq 0.6788 \\ C_{+1}, & \text{c.c} \end{cases} \tag{12}$$

Uma representação gráfica da região de decisão pode ser vista na Fig. 3, vendo tanto o comportamento das curvas das posterioris (13) das classes quanto o comportamento da curva $g(x)$.

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)} \tag{13}$$

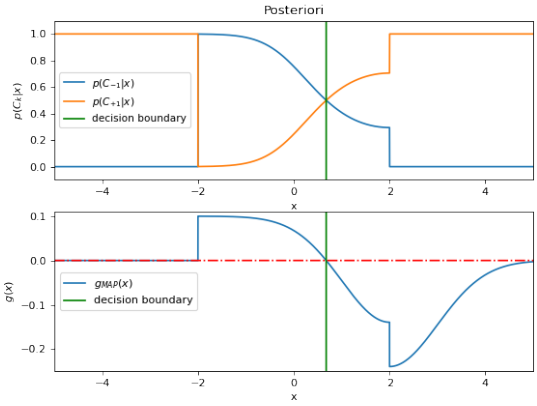


Fig. 3. Curvas das funções $p(C_k|x)$ (cima) e $g(x)$ (baixo).

Para calcular o erro total do classificador temos que calcular o erro de cada classe. Este erro acontece quando se escolher $C_{-1}$ na região de $C_{+1}$ e vice-versa.

$$P(erro|x) = \begin{cases} P(C_{-1}|x), \text{se decidido por } C_{+1} \\ P(C_{+1}|x), \text{se decidido por } C_{-1} \end{cases} \tag{14}$$

Portanto, o erro total é dado por:

$$\begin{aligned} P_{erro} = &\int_{x \in C_{-1}} p(x|C_{+1})P(C_{+1})dx \\ &+ \int_{x \in C_{+1}} p(x|C_{-1})P(C_{-1})dx \end{aligned} \tag{15}$$

Usando os intervalos obtidos por (12), os erros são dados por:

$$\begin{aligned} P_{erro} = &\int_{-\infty}^{-2} p(x|C_{-1})P(C_{-1})dx \\ &+ \int_{-2}^{0.6788} p(x|C_{+1})P(C_{+1})dx \\ &+ \int_{0.6788}^{\infty} p(x|C_{-1})P(C_{-1})dx \end{aligned} \tag{16}$$

Como $p(x|C_{-1}) = 0$ fora do intervalo $|x| \leq 2$, as integrais se tornam:

$$P_{erro} = \int_{-2}^{0.6788} 0.6\frac{e^{-\frac{1}{2}(x-2)^2}}{\sqrt{2\pi}}dx + \int_{0.6788}^{2} \frac{0.4}{4}dx \tag{17}$$

Por fim, tem-se que:

$$P_{erro} = 0.188$$

*C. Recalculate the model's decision boundary if the Maximum-Likelihood criterion is adopted, this time. Analytically calculate the classifier error probability (the integrals can be numerically solved).*

Utilizando o critério da Máxima Verossimilhança, as funções discriminantes passam a ser a verossimilhança, ou seja, a própria função de densidade de probabilidade $p(x|C_k)$.

Assim, tem-se que:

$$g_{-1}(x) = \frac{1}{4} \tag{18}$$

$$g_{+1}(x) = \mathcal{N}(2, 1) \tag{19}$$

O resto do processo é análogo ao critério de Máxima Posteriori, com a diferença das funções discriminantes. Logo:

$$g(x) = \frac{1}{4} - \frac{e^{-\frac{1}{2}(x-2)^2}}{\sqrt{2\pi}} \tag{20}$$

Após as manipulações, chega-se em:

$$x = 2 \pm \sqrt{-2\log\left(\frac{\sqrt{2\pi}}{4}\right)} \tag{21}$$

Assim, se determina as duas regiões de classificação

$$\begin{cases} x_1 = 2.966 \\ x_2 = 1.033 \end{cases} \tag{22}$$

Análogo ao critério de Máxima Posteriori temos:

$$C_k = \begin{cases} C_{-1}, & \text{se } -2 \leq x \leq 1.033 \\ C_{+1}, & \text{c.c} \end{cases} \tag{23}$$

Uma representação gráfica da região de decisão pode ser vista na Fig. 4, observando tanto o comportamento das curvas das verossimilhanças das classes quanto o comportamento da curva $g(x)$.

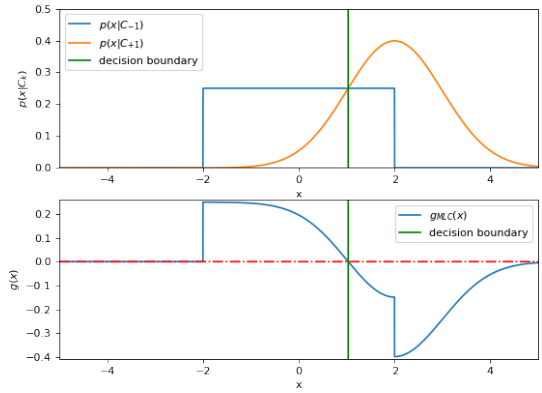


Fig. 4. Curvas das funções $p(x|C_k)$ (cima) e $g(x)$ (baixo).

O erro também é calculado da mesma forma, a partir de (16), porém com os intervalos definidos em (23).

$$P_{erro} = \int_{-2}^{1.033} 0.6 \frac{e^{-\frac{1}{2}(x-2)^2}}{\sqrt{2\pi}} dx + \int_{1.033}^{2} \frac{0.4}{4} dx \tag{24}$$

Por fim, tem-se que:

$$P_{erro} = 0.1967$$

*D. Compare the performances of both methodologies.*

Observando apenas o erro das duas metodologias podemos concluir que o critério de Máxima Posteriori consegue um erro menor que o critério da Máxima Verossimilhança. Isso acontece pelo fato do critério de verossimilhança não levar em consideração a probabilidade a priori no cálculo das funções discriminantes.

Desconsiderando essa informação, a probabilidade do erro aumenta, visto que a probabilidade a priori serve como um peso na função discriminante, fazendo com que a classificação seja mais acurata.

## II. Questão 02

Consider the dataset of 3000 bivariate, labeled samples in dados.csv file.

*A. If the samples labeled with "+1" are drawn by a bivariate gaussian density with* $\mu_{+1} = [6\ -4]^T$, $\Sigma_{+1} = \left[\begin{smallmatrix} 1 & 1 \\ 1 & 4 \end{smallmatrix}\right]$, *the samples labeled with "-1" are drawn by another gaussian density with* $\mu_{-1} = [0.5\ 2.5]^T$, $\Sigma_{-1} = \left[\begin{smallmatrix} 4 & -2.4 \\ -2.4 & 9 \end{smallmatrix}\right]$ *and the prior probabilities are* $P(C_{+1}) = 0.7$ *e* $P(C_{-1}) = 0.3$, *present the discriminant functions of the Bayes classifier and evaluate its performance via the error rate over the dataset.*

A distribuição normal para mais de uma variável é dada por:

$$\mathcal{N}(\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma|^{\frac{1}{2}}} e^{\left[-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right]} \tag{25}$$

A função discriminante é análoga ao caso univariável.

$$g_k(x) = p(x|C_k)P(C_k) \tag{26}$$

Desta forma, as funções discriminantes de cada classe são:

$$g_{-1} = 0.3 \times \mathcal{N}(\mu_{-1}, \Sigma_{-1}) \tag{27}$$

$$g_{+1} = 0.7 \times \mathcal{N}(\mu_{+1}, \Sigma_{+1}) \tag{28}$$

Aplicando as funções discriminantes nas variáveis fornecidas pela fonte de dados obtém-se um erro em 22 pontos. Como o classificador errou apenas 22 vezes em meio as 3000 amostras, pode-se calcular a taxa de erro como:

$$error_{Bayes} = \frac{22}{3000} \tag{29}$$

$$error_{Bayes} = 0.0073 \tag{30}$$

Os erros podem ser visualizados na Fig. 5, pelos pontos em **x**.

Os pontos **x** em azul são onde o classificador de Bayes classificou como $C_{-1}$ e na verdade a classe era $C_{+1}$ e os pontos **x** em vermelho, o classificador de Bayes classificou como $C_{+1}$ e na verdade a classe era $C_{-1}$.

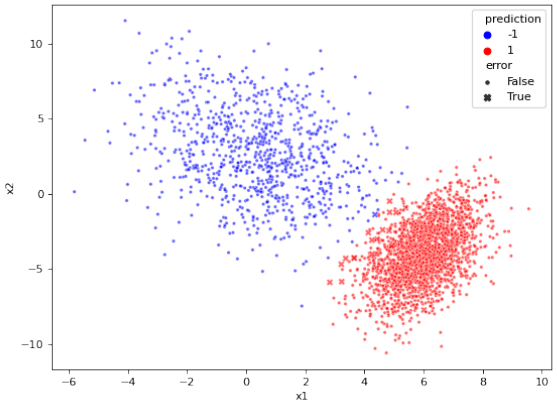


Fig. 5. Gráfico de dispersão das variáveis com suas classificações pelo classificador de Bayes.

*B. Calculate the decision boundary and plot it on a graph with the dataset samples. Comment your results.*

O cálculo da região de decisão é análogo ao caso de uma variável, ou seja a partir da Eq. (4) e caso $g(x)$ for maior ou menor que 0 decidir a classe.

Para encontrar a região de decisão, neste caso deve-se aplicar $\log(.)$ nas funções discriminantes. Assim, tem-se que:

$$g_k(\mathbf{x}) = \log(p(x|C_k)) + \log(P(C_k)) \tag{31}$$

$$g_k(\mathbf{x}) = \log(P(C_k)) - \frac{d}{2}\log(2\pi) - \log(|\Sigma_k|) \\ -\frac{1}{2}(\mathbf{x}-\mu_k)^T\Sigma_k^{-1}(\mathbf{x}-\mu_k) \tag{32}$$

Fazendo todas as multiplicações de matrizes e todas as manipulações para a classe $C_{-1}$ chega-se em:

$$g_{-1}(\mathbf{x}) = -0.149\mathbf{x}_1^2 + 0.347\mathbf{x}_1 - 0.079\mathbf{x}_1\mathbf{x}_2 \\ -0.066\mathbf{x}_2^2 + 0.37\mathbf{x}_2 - 3.458 \tag{33}$$

Análogo, para a classe $C_{+1}$:

$$g_{+1}(\mathbf{x}) = -0.667(\mathbf{x}_1^2 - 14\mathbf{x}_1 - 0.5\mathbf{x}_1\mathbf{x}_2 \\ +0.25\mathbf{x}_2^2 + 5\mathbf{x}_2 + 53.36) \tag{34}$$

A região de decisão é dada pela diferença das funções discriminantes:

$$g(\mathbf{x}) = g_{-1}(\mathbf{x}) - g_{+1}(\mathbf{x}) \tag{35}$$

Portanto, tem-se que:

$$g(\mathbf{x}) = 0.518\mathbf{x}_1^2 - 9.68\mathbf{x}_1 - 0.413\mathbf{x}_1\mathbf{x}_2 \\ +0.101\mathbf{x}_2^2 + 3.7\mathbf{x}_2 + 32.114 \tag{36}$$

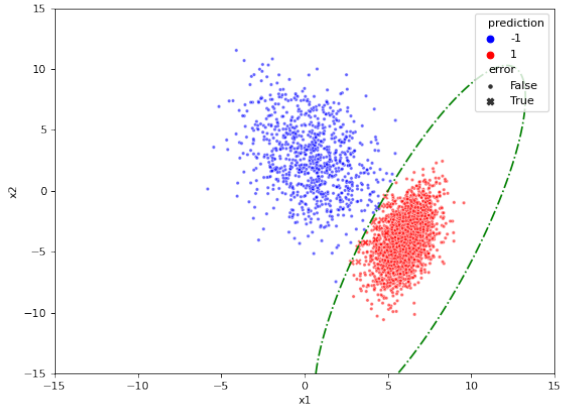


Fig. 6. Gráfico de dispersão das variáveis e sua região de decisão.

A região de decisão pode ser vista na Fig. 6, como a região é elipsoidal, é decidido pela classe $C_{-1}$ quando o ponto está fora da elipse, e $C_{+1}$ quando o ponto se encontra dentro da elipse.

Isso explica os erros encontrados na questão II-B. Na Fig. 6 é possível observar onde o classificador errou. Agora com sua região de decisão também plotada, observa-se que os pontos marcados em $\mathbf{x}$ onde eram pra ser classificados como $C_{-1}$ e foram classificados como $C_{+1}$ estão todos dentro da elipse, fazendo com o que o classificador errasse a classificação desses pontos.

## III. Questão 03

*A. Do your own implementation and train a univariate polynomial regression model for a given problem. Pick some dataset (suggestions: Kaggle, UCI Machine Learning Repository) and employ cross-validation to define the polynomial order that better suits your problem. Comment your results.*

O grupo de dados utilizado foi ”House Prices - Advanced Regression Techniques” do Kaggle, dataset clássico para estudo de regressão. Neste caso, a variável alvo é o preço em que a casa foi vendida enquanto o atributo utilizado foi a área do lote.

Para fazer a escolha da melhor ordem polinomial para regressão, foi aplicada validação cruzada, treinando o modelo para cada grau de regressão e avaliando sua performance por meio do erro quadrático médio no conjunto de validação e treinamento.

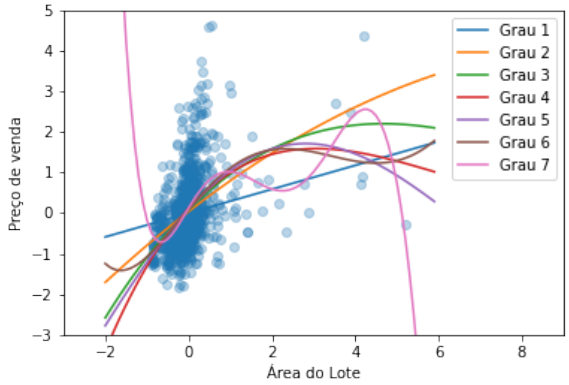


Fig. 7. Gráfico da regressão até o sétimo grau.

Observando a Fig. 8, é possível ver um grande aumento no erro de validação quando o grau polinomial é maior que 6. Este aumento pode caracterizar um ”overfitting” na regressão, visto que o o modelo não generaliza o suficiente, fazendo com que erre mais com novas observações.

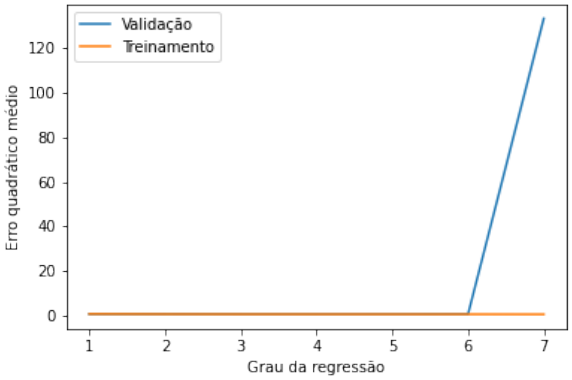


Fig. 8. Curvas dos erros de treinamento e validação.

Como a diferença entre os erros de treinamento e validação é muito pequena para os outros graus polinomiais. De acordo com a Fig. 9 escolhe-se o polinômio de terceiro grau, afim de evitar ”overfitting” utilizando um polinômio de maior grau.

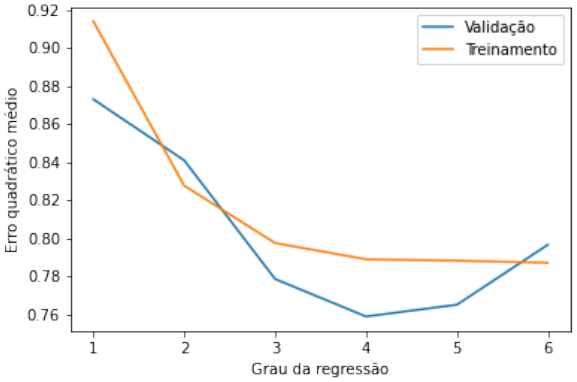


Fig. 9. Curvas dos erros de treinamento e validação até a regressão de sexto grau.

Avaliando a escolha no conjunto de teste, observa-se um erro quadrático médio de 0.801. O valor está perto do erro de treinamento e validação, indicando um pequeno ”overfit”, porém é uma estimativa razoável para a regressão, visto que o erro não foi muito diferente do que era esperado.

## IV. Apêndice

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

x = np.arange(-10, 10, 0.001)

pc0 = np.array([0.25 if np.abs(i) < 2 else 0 for i in x]) # p(x|C-1)
pc1 = np.exp((-(x-2)**2)/2)/np.sqrt((2*np.pi)) # p(x|C+1)
evidence = 0.4*pc0 + 0.6*pc1 # p(x)

g0 = 0.4*pc0 # g-1(x)
g1 = 0.6*pc1 # g+1(x)

posteriorC0 = 0.4*pc0/evidence # p(C-1|x)
posteriorC1 = 0.6*pc1/evidence # p(C+1|x)

likelihoodC0 = pc0
likelihoodC1 = pc1

g_map = 0.4*pc0 - 0.6*pc1 # g(x) - max posteriori
g_mec = pc0 - pc1 # g(x) - max likelihood

fig, ax = plt.subplots()
fig.set_size_inches(8,6)
fig.set_dpi(80)

plt.plot(x, g0, label='$g_{-1}(x)$')
plt.plot(x, g1, label='$g_{+1}(x)$')
plt.xlabel('x')
plt.ylabel('$g_{k}(x)$')
plt.xlim(-5, 5)
plt.legend()

plt.savefig('images/plotq1a')
plt.show()

plt.subplot(2,1,1)
plt.plot(x, posteriorC0, label='$p(C_{-1}|x)$')
plt.plot(x, posteriorC1, label='$p(C_{+1}|x)$')
plt.plot([x[posteriorC0 > posteriorC1][-1] for _ in x], x, label = 'decision boundary', color = 'g')
plt.xlabel('x')
plt.ylabel('$p(C_{k}|x)$')
plt.ylim(-0.1, 1.1)
plt.xlim(-5, 5)
plt.title('Posteriori')
plt.legend(loc = 'center left', prop={'size': 9.5})

plt.subplot(2, 1, 2)
plt.plot(x, g_map, label = '$g_{MAP}(x)$')
plt.plot([x[posteriorC0 > posteriorC1][-1] for _ in x], x, label = 'decision boundary', color = 'g')
plt.plot(x, [0 for _ in x], linestyle = '-.', color = 'r')
plt.ylim(np.min(g_map) - 0.01, np.max(g_map) + 0.01)
plt.xlabel('x')
plt.ylabel('$g(x)$')
plt.xlim(-5, 5)
plt.legend(loc = 'center left')

plt.savefig('images/plotq1b')
plt.show()

plt.subplot(2,1,1)
plt.plot(x, likelihoodC0, label='$p(x|C_{-1})$')
plt.plot(x, likelihoodC1, label='$p(x|C_{+1})$')
plt.plot([x[likelihoodC0 > likelihoodC1][-1] for _ in x], x, label = 'decision boundary', color = 'g')
plt.xlim(-5, 5)
```

```
plt.ylim(0, 0.5)
plt.xlabel('x')
plt.ylabel('$p(x|C_{k})$')
plt.legend()

plt.subplot(2, 1, 2)
plt.plot(x, g_mec, label = '$g_{MLC}(x)$')
plt.plot([x[likelihoodC0 > likelihoodC1][-1] for _ in x], x, label = 'decision boundary', color = 'g')
plt.plot(x, [0 for _ in x], linestyle = '-.', color = 'r')
plt.ylim(np.min(g_mec) - 0.01, np.max(g_mec) + 0.01)
plt.xlim(-5, 5)
plt.xlabel('x')
plt.ylabel('$g(x)$')
plt.legend()

plt.savefig('images/plotq1c')
plt.show()
```

Listing 1. Códigos referente a Questão 01

```
def predict(X, mean_C0, mean_C1, sigma_C0, sigma_C1):

    c0_inv = np.linalg.inv(sigma_C0)
    c1_inv = np.linalg.inv(sigma_C1)

    c0_det = np.linalg.det(sigma_C0)
    c1_det = np.linalg.det(sigma_C1)

    likelihoodC0 = np.zeros(X.shape[1])
    likelihoodC1 = np.zeros(X.shape[1])

    evidence = np.zeros(X.shape[1])

    posteriorC0 = np.zeros(X.shape[1])
    posteriorC1 = np.zeros(X.shape[1])

    pred_class = np.zeros(X.shape[1])
    priorC0 = 0.3
    priorC1 = 0.7

    for i in range(X.shape[1]):

        likelihoodC0[i] = (1/(np.sqrt(2*np.pi*c0_det)))*np.exp((-1/2)*np.dot(np.dot((X[:, i] - mean_C0).T, c0_inv), (X[:, i] - mean_C0)))
        likelihoodC1[i] = (1/(np.sqrt(2*np.pi*c1_det)))*np.exp((-1/2)*np.dot(np.dot((X[:, i] - mean_C1).T, c1_inv), (X[:, i] - mean_C1)))

        evidence = likelihoodC0[i]*priorC0 + likelihoodC1[i]*priorC1

        posteriorC0[i] = likelihoodC0[i]*priorC0/evidence
        posteriorC1[i] = likelihoodC1[i]*priorC1/evidence

        pred_class[i] = np.argmax([posteriorC0[i], posteriorC1[i]])

    return pred_class

def set_coordinates(d):

    x1 = np.linspace(-d, d, 1000)
    x2 = np.linspace(-d, d, 1000)
    X1, X2 = np.meshgrid(x1, x2)

    pos = np.empty(X1.shape + (2,))
    pos[:, :, 0] = X1
    pos[:, :, 1] = X2

    return X1, X2, pos

def exp2d(x, u, s):

    n = u.shape[0]
    sigma_det = np.linalg.det(s)
    sigma_inv = np.linalg.inv(s)
    N = np.sqrt((2*np.pi)**n * sigma_det)

    k = np.einsum('...k,kl,...l->...', x-u, sigma_inv, x-u)
    return np.exp(-k/2)/N

data = pd.read_csv('dados.csv', header = None, names = ['x1', 'x2', 'C'])

x1_C0 = data.loc[data['C'] == -1, 'x1']
x2_C0 = data.loc[data['C'] == -1, 'x2']

x1_C1 = data.loc[data['C'] == 1, 'x1']
x2_C1 = data.loc[data['C'] == 1, 'x2']

mean_C0 = np.array([x1_C0.mean(), x2_C0.mean()])
mean_C1 = np.array([x1_C1.mean(), x2_C1.mean()])

sigma_C0 = np.cov(x1_C0, x2_C0)
sigma_C1 = np.cov(x1_C1, x2_C1)

X = np.array(data[['x1', 'x2']]).T

data['prediction'] = predict(X, mean_C0, mean_C1, sigma_C0, sigma_C1)
data.loc[data['prediction'] == 0, 'prediction'] = -1
data['prediction'] = data['prediction'].astype(int)
data['error'] = data['prediction'] != data['C']
error_rate = data['error'].sum()/data.shape[0]
print(error_rate)

fig, ax = plt.subplots()
fig.set_size_inches(8,6)
fig.set_dpi(80)
sns.scatterplot(x = data['x1'], y = data['x2'], alpha = 0.6, hue = data['prediction'], style = data['error'], palette = 'bwr', markers = ['.', 'X'])
plt.savefig('images/plotq2a')
plt.show()

X1, X2, pos = set_coordinates(15)

ZC0 = 0.3*exp2d(pos, mean_C0, sigma_C0)
ZC1 = 0.7*exp2d(pos, mean_C1, sigma_C1)
Z_boundary = ZC0 - ZC1

ax.contour(X1, X2, Z_boundary, levels = 0, colors = 'g', linestyles = '-.')
sns.scatterplot(x = data['x1'], y = data['x2'], alpha = 0.6, hue = data['prediction'], style = data['error'], palette = 'bwr', markers = ['.', 'X'])
plt.xlabel('x1')
plt.ylabel('x2')
plt.savefig('images/plotq2b')
plt.show()
```

Listing 2. Códigos referente a Questão 02

```

def calc_weights(D, y):

    w = np.dot(np.dot(np.linalg.inv(np.dot(D.T, D)), D.T), y)
```

```
    return w

def degree_map(X, degree):

    D = np.zeros([X.shape[0], degree + 1])

    for i in range(degree + 1):
        D[:, i] = X**i

    return D

def mean_square_error(y_pred, y):

    return np.sum((y_pred - y)**2)/len(y)

def normalize(train, val, test):
    mu = np.mean(train)
    sigma = np.std(train)

    train_norm = (train - mu)/sigma
    val_norm = (val - mu)/sigma
    test_norm = (test - mu)/sigma

    return train_norm, val_norm, test_norm

houses = pd.read_csv('train.csv')[['LotArea', '
    SalePrice']]

train_data = houses.sample(frac = 0.6, random_state
    = 0)
test_validation = houses.drop(train_data.index)
validation_data = test_validation.sample(frac = 0.5,
     random_state = 0)
test_data = houses.drop(validation_data.index.union(
    train_data.index))

X_train, y_train = train_data['LotArea'], train_data
    ['SalePrice']
X_val, y_val = validation_data['LotArea'],
    validation_data['SalePrice']
X_test, y_test = test_data['LotArea'], test_data['
    SalePrice']

X_train, X_val, X_test = normalize(X_train, X_val,
    X_test)
y_train, y_val, y_test = normalize(y_train, y_val,
    y_test)

val_error = []
train_error = []
X_plot = np.arange(-2, 6, 0.1)

max_degree = 6
for degree in range(1, max_degree + 1):

    X_train_map = degree_map(X_train, degree)
    X_val_map = degree_map(X_val, degree)
    w = calc_weights(X_train_map, y_train)

    y_predTrain = np.dot(X_train_map, w)
    y_predVal = np.dot(X_val_map, w)

    val_error.append(mean_square_error(y_predVal,
    y_val))
    train_error.append(mean_square_error(y_predTrain
    , y_train))

    plt.plot(X_plot, np.dot(degree_map(X_plot,
    degree), w), label = f'Grau {degree}')

plt.scatter(X_train, y_train, alpha = 0.3)
plt.xlabel(' rea  do Lote')
plt.ylabel('Pre o de venda')
plt.xlim([-3, 9])
plt.ylim([-3, 5])
plt.legend()

plt.plot([d for d in range(1, degree+1)], val_error,
     label = 'Valida  o')
plt.plot([d for d in range(1, degree+1)],
    train_error, label = 'Treinamento')
plt.xlabel('Grau da regress o')
plt.ylabel('Erro quadr tico m dio')

plt.legend()
plt.savefig('images/plotq3c')
```

Listing 3. Códigos referente a Questão 03