Universidade Federal de Pernambuco

Centro de Informática

Aprendizagem de Máquina

Projeto de Graduação, 2014-2

Eduardo M. B. de A. Tenório
Hélio de M. Lins Neto
Hugo B. Barbosa

4 de fevereiro de 2015

Introdução

O projeto consiste em gerar 300 amostras a partir de distribuições gaussianas bivariadas, formando 2 classes. A classe 1 é dividida nas subclasses 1-1 e 1-2, com 100 amostras cada, e a classe 2 com 100 amostras também. As funções geradoras possuem os seguintes parâmetros:

1-1: $\mu_1 = 60$, $\mu_2 = 30$, $\sigma_1^2 = 9$ e $\sigma_2^2 = 144$

1-2: $\mu_1 = 52$, $\mu_2 = 30$, $\sigma_1^2 = 9$ e $\sigma_2^2 = 9$

2: $\mu_1 = 45$, $\mu_2 = 22$, $\sigma_1^2 = 100$ e $\sigma_2^2 = 9$

Os dados podem ser gerados por uma das duas funções: `mvnrnd` do MATLAB ou `multivariate_normal` do pacote `random` de NumPy (biblioteca numérica para Python). São registrados nos arquivos texto *r11.txt*, *r12.txt* e *r2.txt*, com cada linha correspondendo a uma amostra, composta por dois números reais representando a primeira e a segunda variáveis, respectivamente.

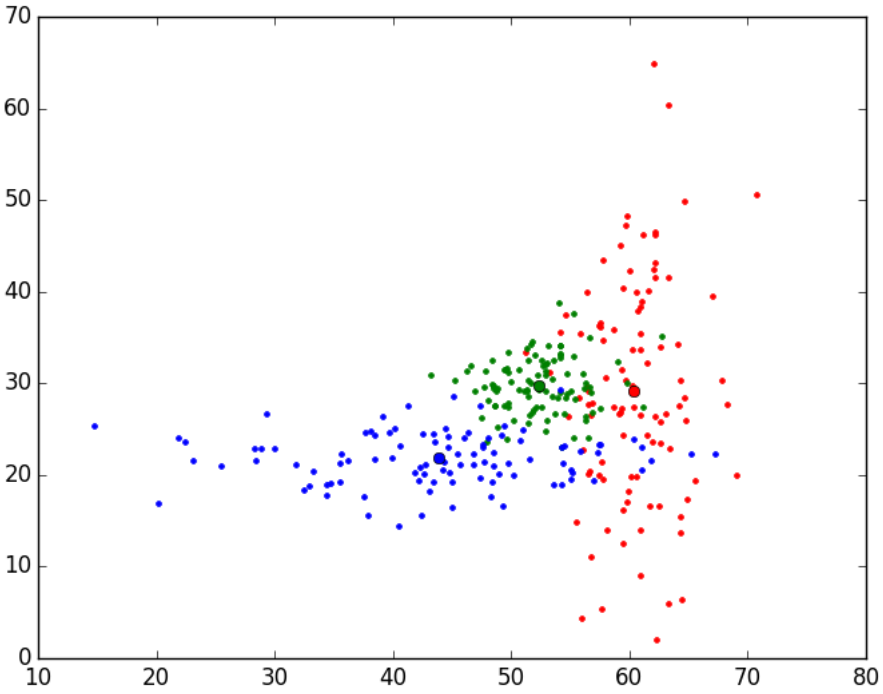

Figura 1: Classes 1 (1-1 em vermelho e 1-2 em verde) e 2 (azul).

Questão 1

O algoritmo FCM-DFCV foi retirado de [1], implementado utilizando Python3.4 com a biblioteca NumPy. A biblioteca matplotlib foi utilizada para gerar os gráficos necessários para a visualização dos dados e da classificação final.

Ocorreram poucas dificuldades, entre elas a falta de familiaridade com a biblioteca NumPy e o entendimento do algoritmo em si. A implementação alcançou protótipos muito parecidos com a média das distribuições 1-1, 1-2 e 2, obtendo um índice de Rand entre 55% e 70% dependendo da distribuição. Em grande parte das vezes o algoritmo converge antes de 150 iterações. Os dados descritos em Fig. 1 foram agrupados utilizando Fuzzy c-Means (FCM), como mostrado em Fig. 2.

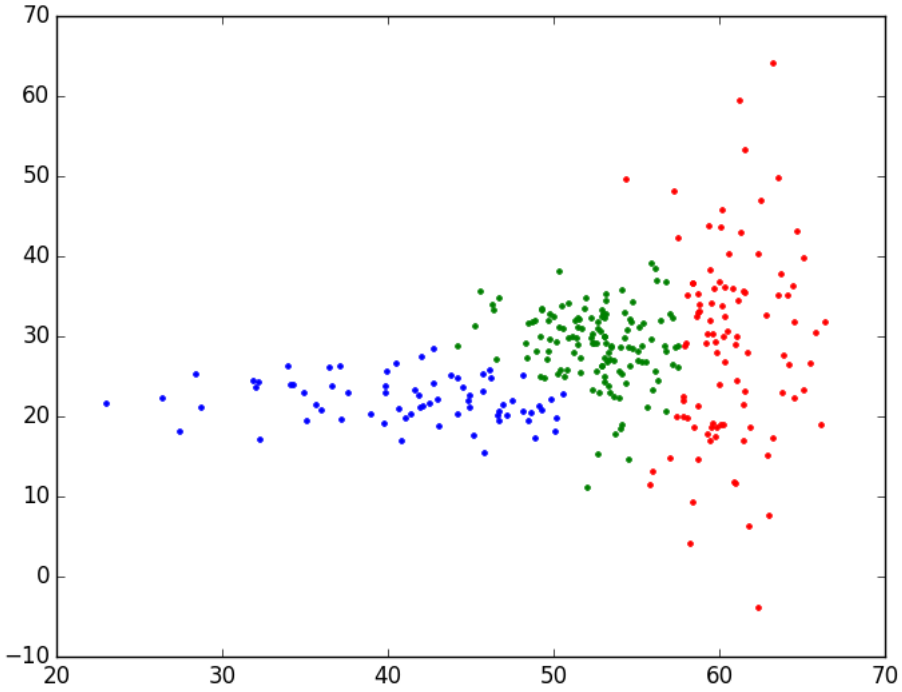

Figura 2: Classes 1 (1-1 em vermelho e 1-2 em verde) e 2 (azul).

Questão 2

As probabilidades a priori $P(\omega_1)$ e $P(\omega_2)$ são determinadas pela maximização da verossimilhança. Neste caso correspondem à razão entre a quantidade de objetos da classe e do conjunto universo:

$$\begin{aligned} P(\omega_1) &= \frac{200}{300} = \frac{2}{3} \\ P(\omega_2) &= \frac{100}{300} = \frac{1}{3} \end{aligned} \tag{1}$$

(A)

Para a gaussiana bivariada o método da Estimação da Máxima Verossimilhança (Maximum Likelihood Estimation, MLE) nos diz que a média e a matrix de covariância são dadas por

$$\mu = \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k \tag{2}$$

$$\mathbf{\Sigma} = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{x}_k - \mu)(\mathbf{x}_k - \mu)' \tag{3}$$

respectivamente, cuja implementação encontra-se em `segunda_a.m`. A densidade de probabilidade para a classe 2 encontra-se em Fig. 3. Para a classe 1 o algoritmo Expectation-Maximization (EM) estima um média e uma matriz de covariância para cada gaussiana da mistura. A descrição do processo encontra-se no arquivo `EM_GM.m`, com alguns resultados em Fig. 4 e Fig. 5.

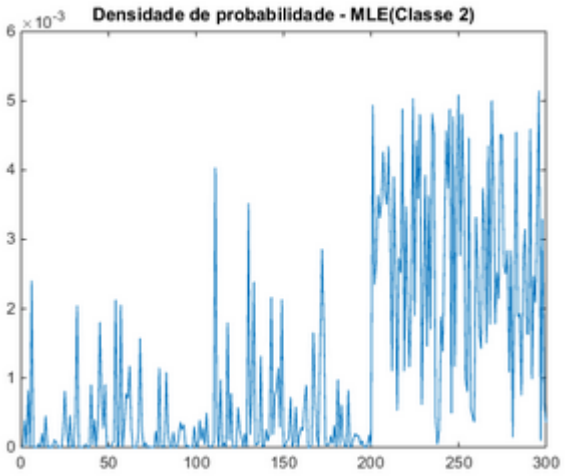

Figura 3: Densidade de probabilidade

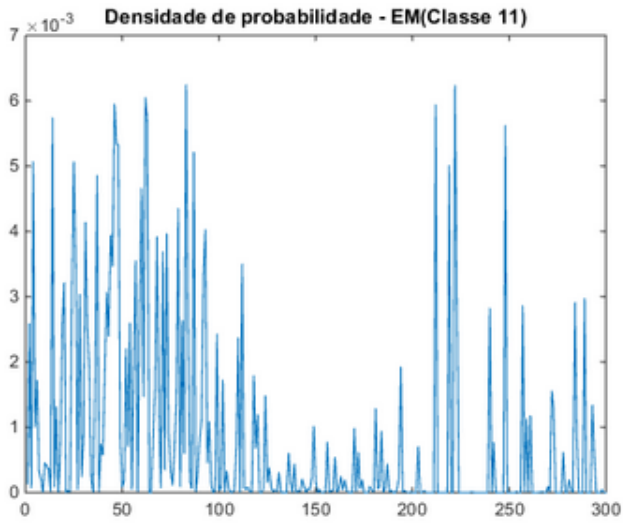

Figura 4:

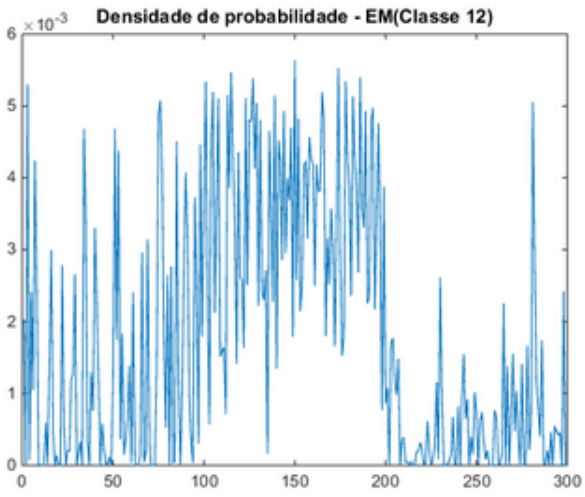

Figura 5:

Agora podemos calcular a densidade de probabilidade total da classe 1, ponderando os valores das subclasses (obtidos pelo EM) e misturando as gaussianas. O resultado é mostrado em Fig. 6.

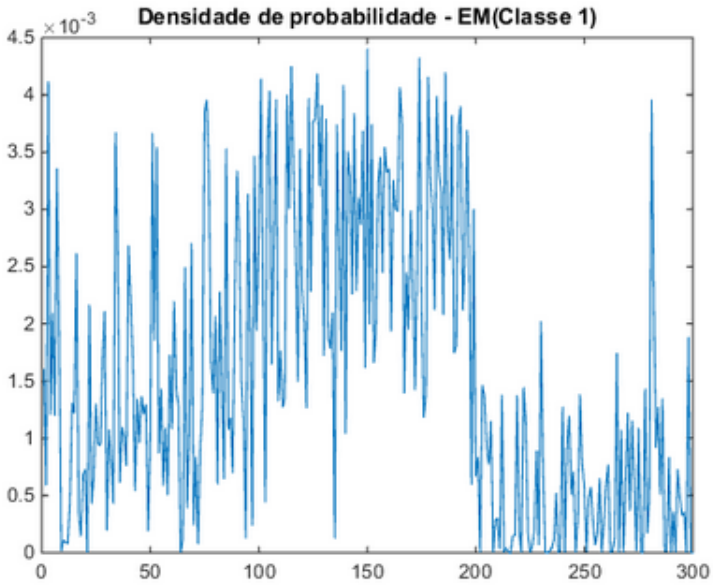

Figura 6:

Por fim, $P(\omega_i|\mathbf{x}_k)$ é calculado a partir dos parâmetros já encontrados, exibida em Fig. 7 e Fig. 8.

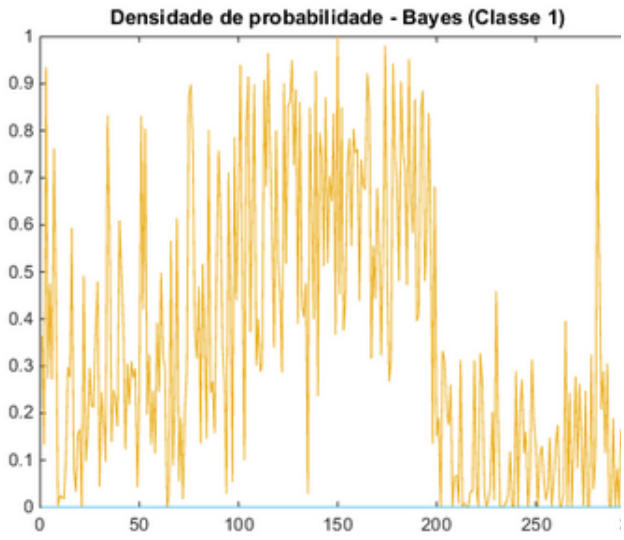

Figura 7: Densidade de probabilidade por Bayes, classe 1.

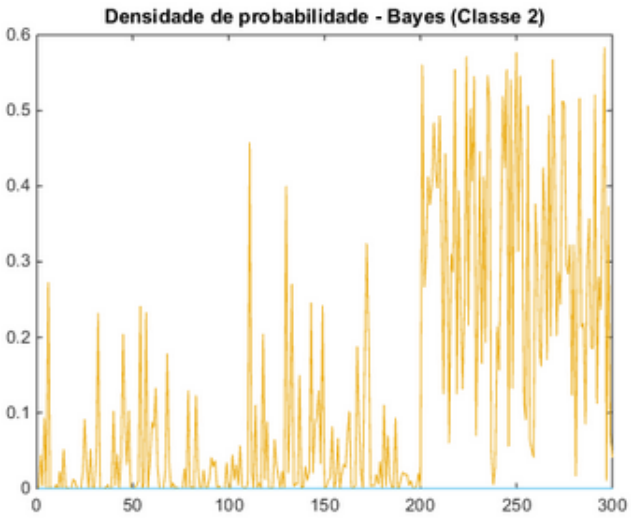

Figura 8: Densidade de probabilidade por Bayes, classe 2.

(B)

Utilizando a função de kernel bivariada, os códigos `parzen.m`, `bivar.m`, `multi.m` e `uni.m` implementam a Janela de Parzen e definem as densidades de probabilidade variando o valor de h. A partir destes resultados é possível calcular $P(\omega_i|\mathbf{x}_k)$, esboçado nas figuras abaixo.

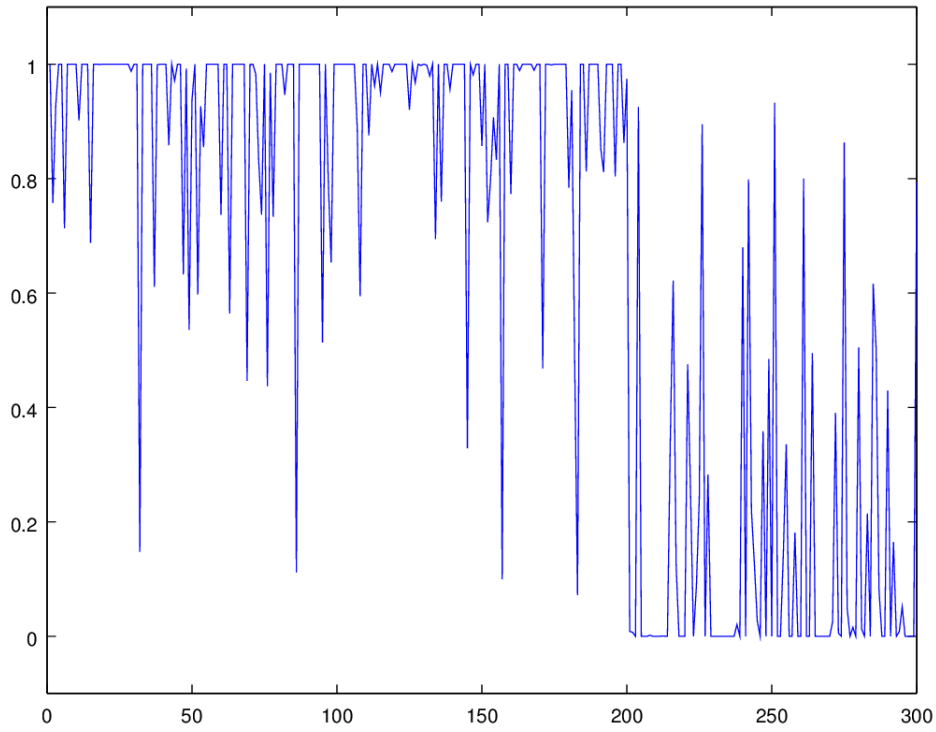

Figura 9: $P(\omega_1|\mathbf{x}_k)$ para k = 1.

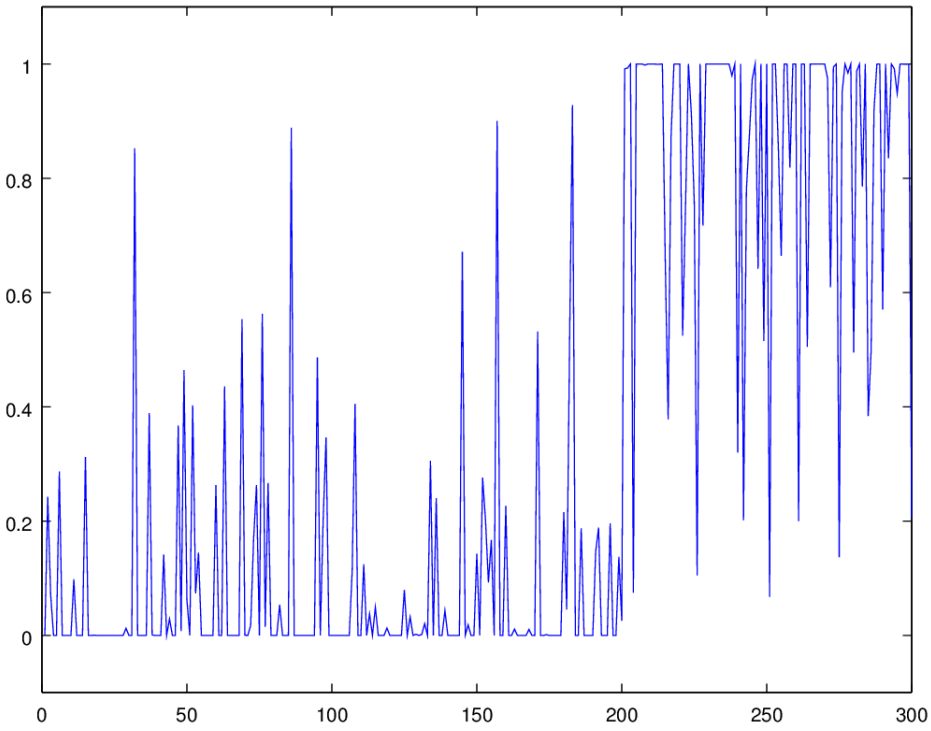

Figura 10: $P(\omega_2|\mathbf{x}_k)$ para k = 1.

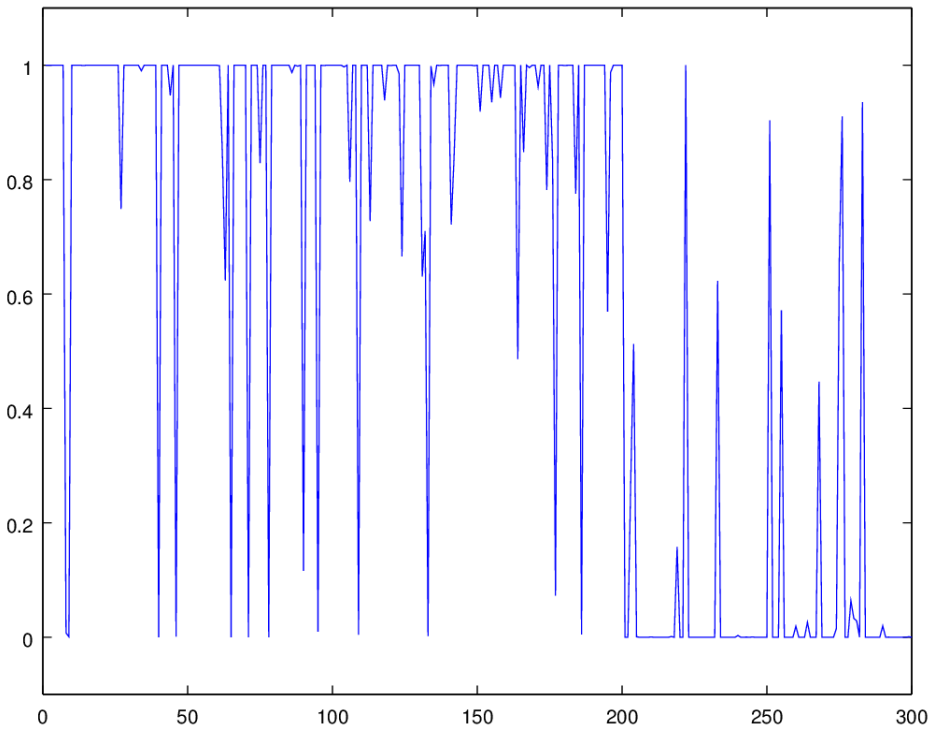

Figura 11: $P(\omega_1|\mathbf{x}_k)$ para k = 0.5.

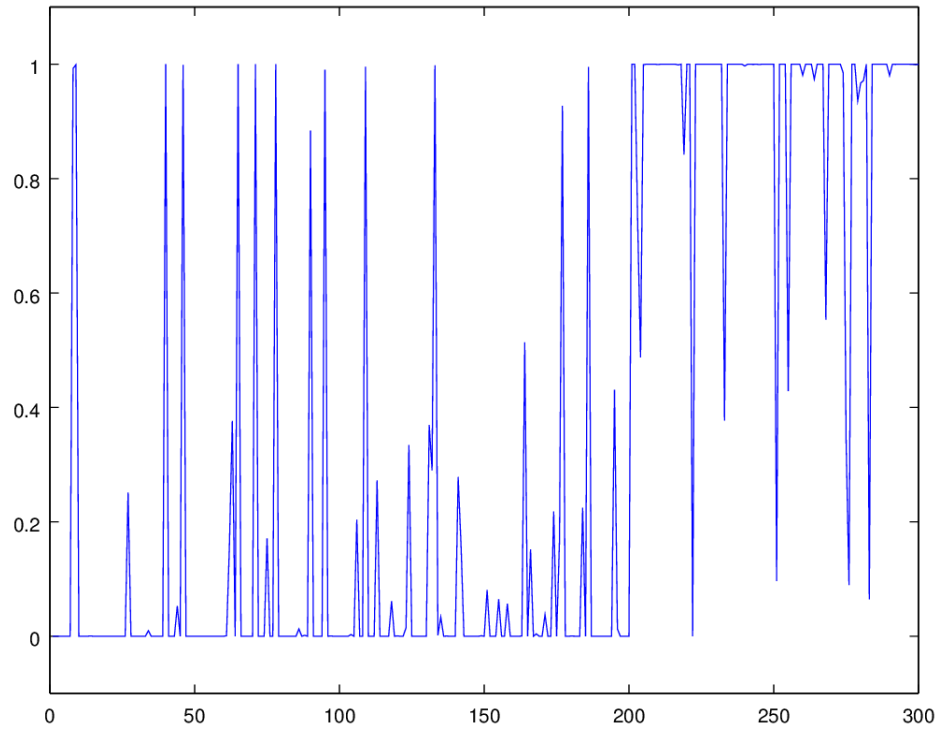

Figura 12: $P(\omega_2|\mathbf{x}_k)$ para k = 0.5.

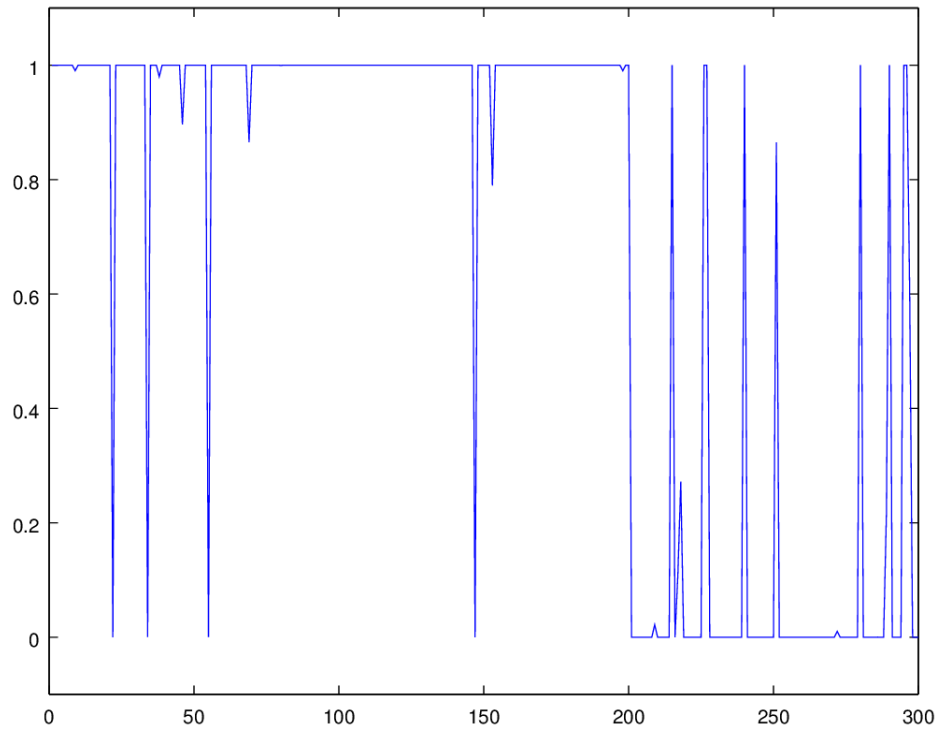

Figura 13: $P(\omega_1|\mathbf{x}_k)$ para k = 0.3.

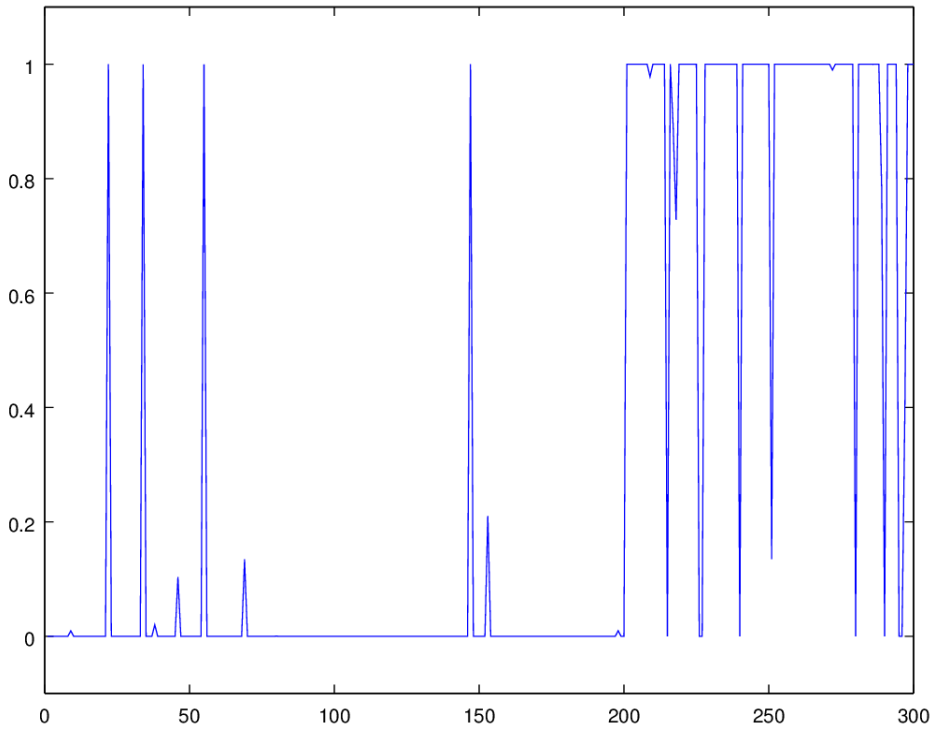

Figura 14: $P(\omega_2|\mathbf{x}_k)$ para k = 0.3.

(C)

O código `kNN.py` implementa a probabilidade *a posteriori* de ω_i em relação a uma amostra $\mathbf{x}$. Como demonstrado em [2], uma boa estimativa de $P(\omega_i|\mathbf{x})$ é

$$P_n(\omega_i|\mathbf{x}) = \frac{p_n(\mathbf{x}, \omega_i)}{\sum_{j=1}^{c} p_n(\mathbf{x}, \omega_j)} \tag{4}$$

e como

$$p_n(\mathbf{x}, \omega_i) = \frac{k_i/n}{V} \tag{5}$$

e

$$\sum_{j=1}^{c} k_j = k \tag{6}$$

então Eq. 4 torna-se

$$P_n(\omega_i|\mathbf{x}) = \frac{k_i}{k} \tag{7}$$

A equação descrita acima apenas demonstra que $P_n(\omega_i|\mathbf{x})$ tende a $P(\omega_i|\mathbf{x})$ quando k cresce. Logo, achando os k vizinhos mais próximos ao ponto $\mathbf{x}$ temos o valor estimado de sua probabilidade *a posteriori*. Fig. 15 e Fig. 16 mostram as estimações para $k = 9$. Note que o gráfico de ω_1 é o inverso do de ω_2. Isto ocorre porque

$$P_n(\omega_1|\mathbf{x}) + P_n(\omega_2|\mathbf{x}) = 1 \tag{8}$$

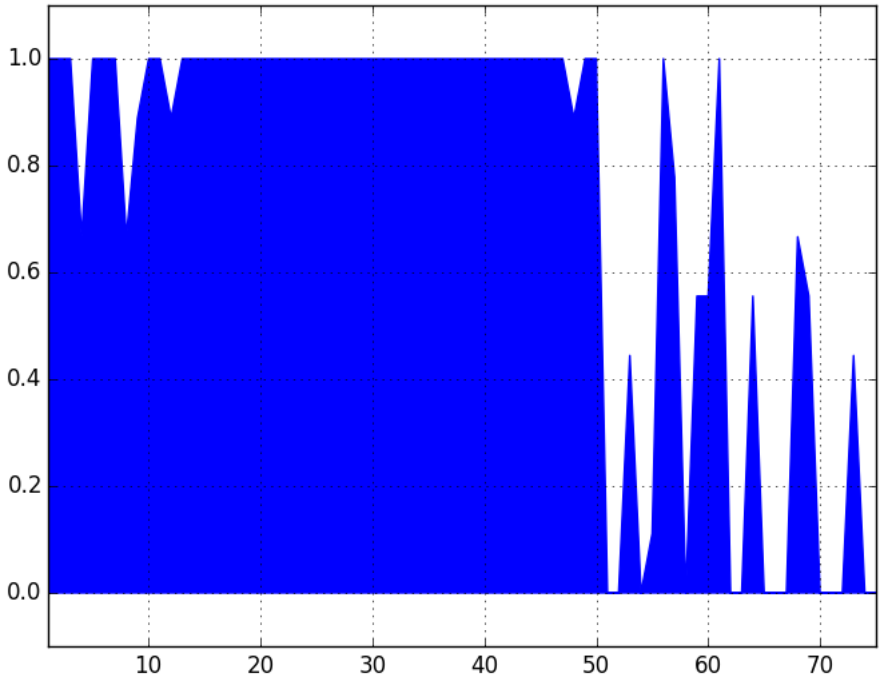

Figura 15: P($\omega_1|\mathbf{x}$), para k = 9.

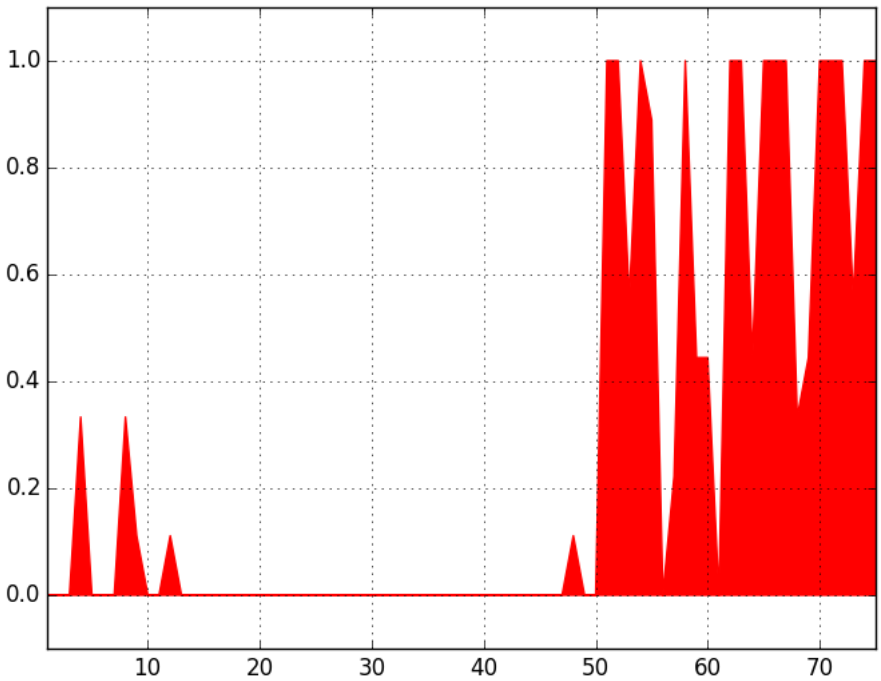

Figura 16: P($\omega_2|\mathbf{x}$), para k = 9.

(D)

Utilizando um MLP treinado com backpropagation obtemos através de `mlp.py` um erro médio quadrático convergente após 300 épocas. O algoritmo permanece em execução até 10000 épocas (Fig. 17). Foram usadas 3 camadas escondidas e o

resultado da classificação foi documentado em *mlpout.txt*, onde a primeira coluna corresponde aos objetos da classe 1 e 2, respectivamente, na mesma ordem das matrizes r11, r12 e r2. A segunda coluna corresponde a saída da rede neural e a terceira, ao erro. Considerou-se que a classe 2 deve ter resposta ideal zero e a classe 1, deverá ter resposta 1.

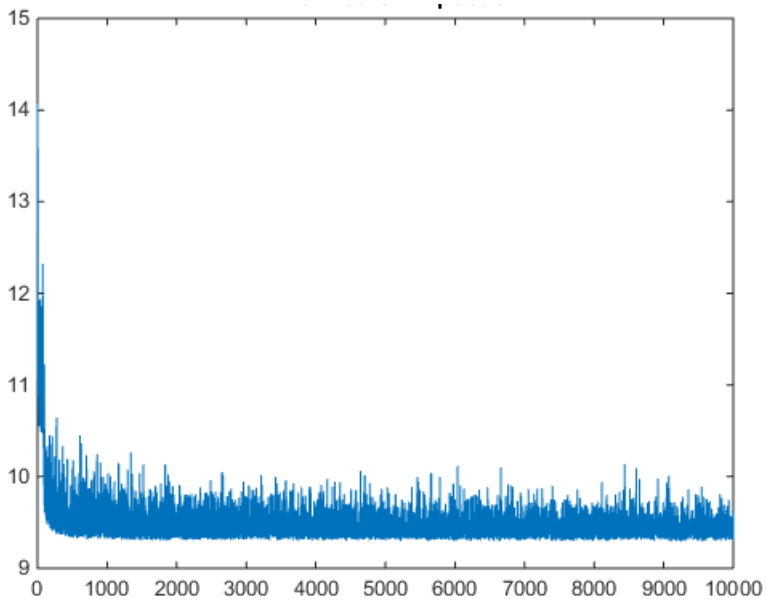

Figura 17: Erro médio x Épocas.

(E)

A regra da soma, implementada em `regra_soma.m`, para uma determinada classe ω_j é dada por

$$(1 - L)P(\omega_j) + \sum_{i=1}^{L} p(\omega_j|\mathbf{x_i}) \tag{9}$$

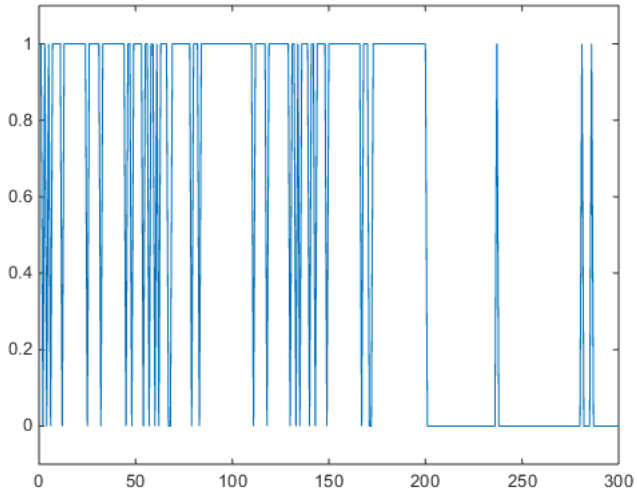

Figura 18:

Questão 3

Bibliografia

[1] Francisco de A.T. de Carvalho, Camilo P. Tenório e Nicomedes L. Cavalcanti Junior. "Partitional fuzzy clustering methods based on adaptive quadratic distances". Em: *Fuzzy Sets and Systems* 157 (2006), pp. 2833–2857.

[2] Richard O. Duda, Peter E. Hart e David G. Stork. *Pattern Classification*. second. Wiley-Interscience, 2000.