Bruno Bacelar Abe - 9292858
Exercício 3 - Comparação RBF e MLP

Foram comparados os resultados entre o MLP e RBF para a execução do exercício. O algoritmo do MLP foi reutilizado (o entregue no exercício, não o do trabalho), enquanto que o RBF foi feito “na mão”.
Quanto a implementação do RBF, utilizamos uma estratégia de sempre colocar os centros na média (como sugerido no trabalho). Como há apenas 3 classes, criamos 3 vetores (que foram colocados juntos, formando apenas um) um para cada classe.

```
sigma = (1/len(t1))*np.sum(np.sqrt(np.sum((t1 - center1) **2, axis=1)))
beta = 1/(2*(sigma**2))
self.beta.append(beta)
sigma = 0
beta = 0
```

(Fig. 1: Betas e Sigmas)

```
for i in range(len(t1)):
    for j in range(len(t1[0])):
        center1[j] += t1[i][j]

center1 = center1/len(t1)
```

(Fig. 2: Centros)

Nas imagens acima, podemos ver como encontramos os centros, betas e sigmas. Uma vez que os centros foram encontrados, fazemos uma alimentação e multiplicação simples com os pesos.
Como função de ativação, que era “livre”, escolhi utilizar a seguinte:

```
def activationFunciton(self, inputs, center, beta):
    return (np.exp(-beta*np.sum((inputs-center)**2, axis=1)))
```

(Fig. 3: Função de ativação)

Quanto aos resultados obtidos, o MLP obteve um desempenho melhor do que o RBF, numa média.

```
C:\Users\Abe\Desktop\RBF\Teste1>python rbf.py
normalizando
normalizando
Resultado: RBF
0.9253348302838106
C:\Users\Abe\Desktop\RBF\Teste1>PYTHON mlPerceptron.py
normalizando
Resultado: MLP
resp:  0.99929187805503
```

(Fig. 4: Resultados)

O resultado de cima mostra que o MLP é mais eficiente do que o RBF, na média (quando falamos de resultado médio). Quanto a velocidade de execução, ambos executaram em velocidades parecidas e com 1000 iterações cada.
Ambos foram testados e treinados com as mesmas bases de dados, e todos os códigos, arquivos de texto e afins estão anexados na entrega.