Exercício 6 - *Radial Basis Function*

Frederico Schardong

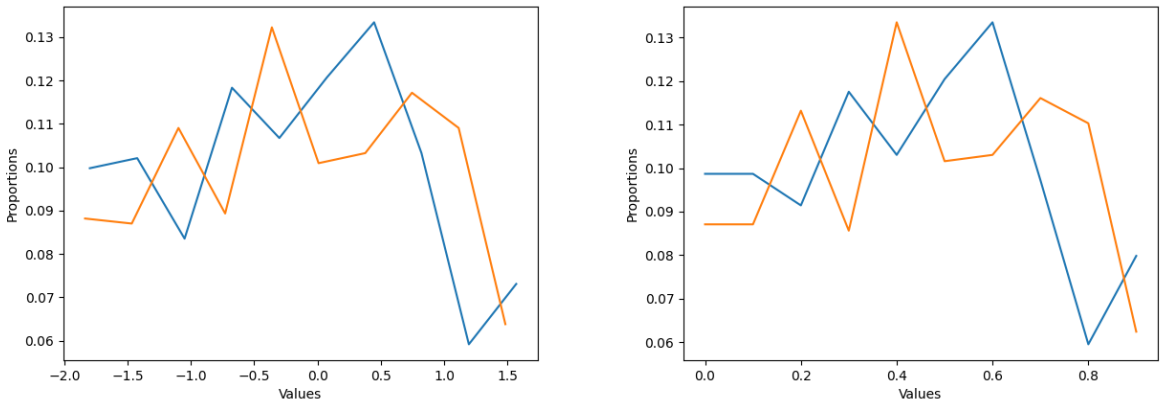
1 Enunciado do trabalho

O enunciado do exercício propõem a criação de uma rede neural do tipo *Radial Basis Function*(RBF) para classificar um dataset com 863 instâncias de 2 dimensões e 2 possíveis classes. A camada intermediária da rede RBF deve ser treinada, bem como o perceptron de saída da rede.

2 Resolução do Trabalho

Em todos os experimentos conduzidos neste trabalho, 80% das instâncias, ou seja, 690 instâncias, foram aleatoriamente selecionadas para treinamento da rede, e as restantes 173 para teste. Por fim, para treinar e medir o desepenho de todas as redes testadas, a métrica `f1-score`, que é a média harmônica da acurácia e da precisão, foi utilizada.

A distribuição dos dados deste dataset são simétricas, como pode ser visto na Figura 1a. Consequentemente, o normalizador `min max` foi utilizado apenas para transportar os dados de entrada para o intervalo $[0; 1]$, como pode ser visto na Figura 1b. Os valores mínimo, máximo, médio e desvio padrão dos dados, bem como após a aplicação da técnica de normalização são listados na Tabela 1.



(a) Sem normalização.

(b) Com normalização `min max`.

Figure 1: Histograma dos dados antes e depois de serem normalizados. Cada linha representa uma dimensão diferente dos dados de entrada.

Os parâmetros ajustaveis de uma rede RBF são: (i) x: posição do centro da função radial; e (ii) σ: unidade de espaço que determina a distância em que a função radial terá influência. Como a função radial é uma função de densidade Gaussiana multivariada, a função de saída dos neurônios da camada escondida é dada por:

$$a_h(x) = e^{\frac{-||x-u||^2}{\sigma^2}} \tag{1}$$

	Dimensão	Mínimo	Máximo	Média	Desvio Padrão
Sem normalização	0	-1.80	1.94	0.00	1.00
Sem normalização	1	-1.84	1.85	0.00	1.00
`min max`	0	0.00	1.00	0.48	0.27
`min max`	1	0.00	1.00	0.50	0.27

Table 1: Características da distribuição antes e depois das normalizações.

Sendo o objetivo do projetista encontrar a quantidade de centros (*i.e.* neurônios na camada intermediária) e seus respectivos σ para o problema em questão. Existem também os pesos entre os neurônios da camada escondida e o neurônio da camada de saída. Como este neurônio possui uma função de ativação linear, os pesos que minimizam o erro quadrático médio são encontrados otimamente via fórmula analítica $((X^T X)^{-1}.X^T Y)$.

O desafio de utilizar uma rede RBF pode ser dividido em três sub-problemas:

- Determinar a quantidade de funções radiais (neurônios na camada intermediária);
- Determinar a posição das funções radiais em relação aos dados de entrada;
- Determinar σ para cada função radial.

Para resolver o segundo sub-problema (posição do centro das funções radiais), o algoritmo `k-means` foi utilizado.

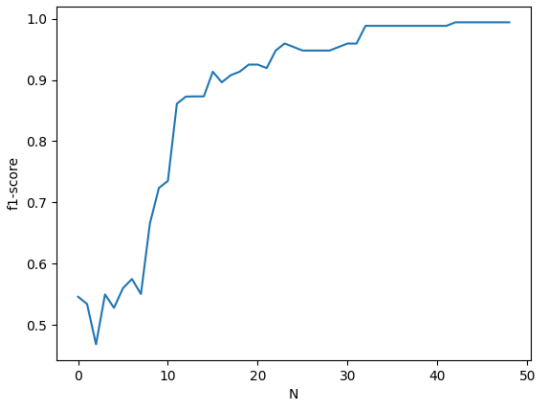
Em relação ao terceiro problema, um número alto de funções radiais foi utilizado (50) e três técnicas para determinar a melhor forma de calcular σ para cada um dos 50 neurônios foram testadas. Todas elas consistem em calcular a distância de cada centróide em relação aos outros, a distinção está na forma de seleção de quais distâncias serão usadas para compor σ:

- Selecionar N distâncias aleatoriamente;
- Selecionar N distâncias em ordem crescente;
- Selecionar N distâncias em ordem decrescente.

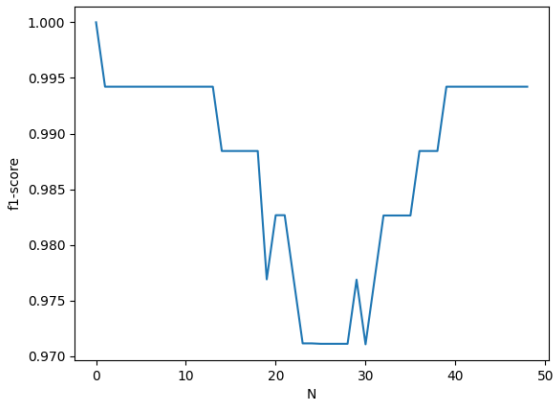
Para determinar a melhor solução, foram realizados experimentos onde N variou de 2 a 49 para cada uma das três opções listadas acima. A métrica `f1-score` foi utilizado para medir o desempenho de cada técnica e os resultados são reportados nas Figuras 2a, 2b e 2c.

Estes experimentos mostram que: (i) para 50 neurônios a métrica `f1-score` converge para 1 (melhor resultado possível) quando os $N > 10$ centróides que compõem cada σ são selecionados aleatoriamente; (ii) quando os N centróides mais próximos são selecionados, o `f1-score` se mantém próximo de 1, nunca sendo menor que 0.97; e (iii) quando N centróides mais distantes são selecionados, o desempenho é o mesmo para qualquer N, sempre acima de 0.99. Desta forma, o terceiro método para determinar σ foi escolhido para ser utilizado nos próximos experimentos.

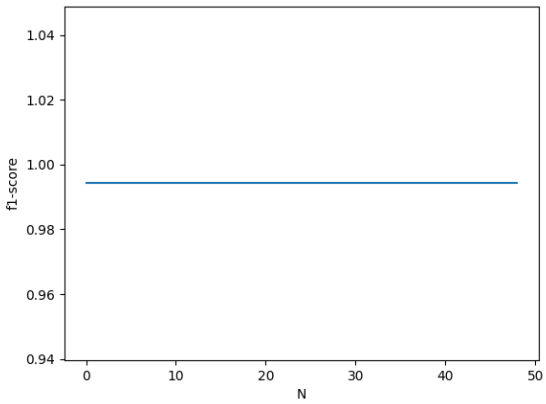
Finalmente apenas o sub-problema de determinar a quantidade de funções radiais (neurônios na camada intermediária) precisa ser atacado. Para entender o comportamento das diferentes quantidades de neurônios, de 2 a 50 neurônios foram utilizados em conjunto com a terceira técnica de cálculo de σ. A Figura 3 mostra o crescimento de `f1-score` a medida que mais neurônios são adicionados na camada intermediária. Como pode ser visto, a partir de cerca de 25 neurônios o `f1-score` se estabiliza próximo de 1. Na Figura 4a são mostrados todos os 863 dados de entrada divididos nas duas categorias. Os 25 clusters retornados pelo `k-means` pode ser visualizado na Figura 4b, bem como a classificação dos dados de treinamento na Figura 4c e o resultado da classificação do conjunto de testes na Figura 4d. A matriz de confusão desta configuração é exibida na Tabela 2.



(a) Selecionados aleatoriamente.



(b) Selecionados em ordem crescente.



(c) Selecionados em ordem decrescente.

Figure 2: N centróides selecionados de três formas diferentes.

3 Detalhes de implementação

O trabalho foi implementado em Python 3, e as seguintes bibliotecas foram utilizadas:

- `scikit-learn` [2] para: (i) normalizar os dados; e (ii) criar a rede neural, treiná-la e testá-la;
- `numpy` [3] para obter os valores mínimos, máximos, média e desvio padrão antes e depois das normalizações;
- `matplotlib` [1] para geração dos gráficos;

Todas as bibliotecas podem ser instaladas através do gerenciador de pacotes padrão do Python, o `pip`, através do comando:

```
$ pip install scikit-learn numpy matplotlib
```

O código fonte desta tarefa é publico[1] e possui a licensa MIT. Ele foi submetido junto com este relatório. Para reproduzir as imagens e tabelas listas neste relatório, basta executa o comando abaixo.

```
$ python3 main.py
```

[1] Código fonte disponível em `https://github.com/fredericoschardong/RBFNN`.

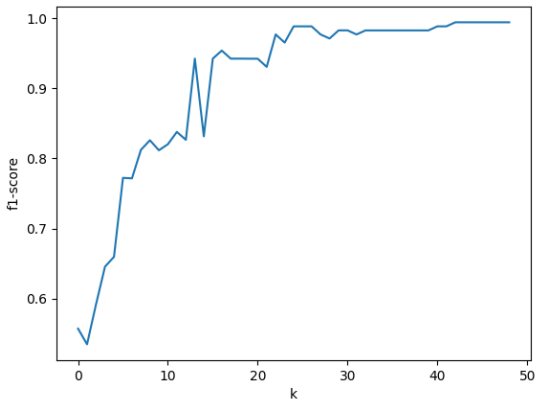


Figure 3: Evolução de `f1-score` para diferentes quantidades de neurônios na camada escondida.

		Valor Previsto	
		Positivo	Negativo
Valor Real	Positivo	71	4
	Negativo	2	96

Table 2: Matrix de confusão usando 25 neurônios na camada escondida.

4 Considerações Finais

Neste trabalho três métodos diferentes para o cálculo de σ para redes RBF foram avaliados. O que apresentou o melhor resultado foi utilizado para determinar o menor número de neurônios necessários na camada escondida para resolver o problema proposto. Para realizar todos os testes 80% dos dados foram usados para treinamento e 20% para teste.

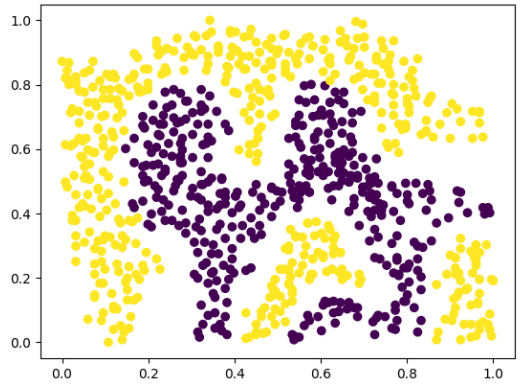
Com os testes executados é possível observar que após encontrar uma heurística de desempenho razoável para calcular σ, as redes RBF com diferentes quantidades de neurônios na camada escondida rapidamente (apenas 25 neurônios) convergiram para um ótimo desempenho (`f1-score` próximo de 1).

References

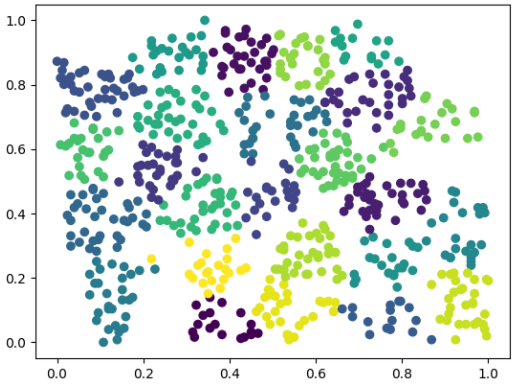
[1] John D Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in science & engineering* 9.3 (2007), pp. 90–95.

[2] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

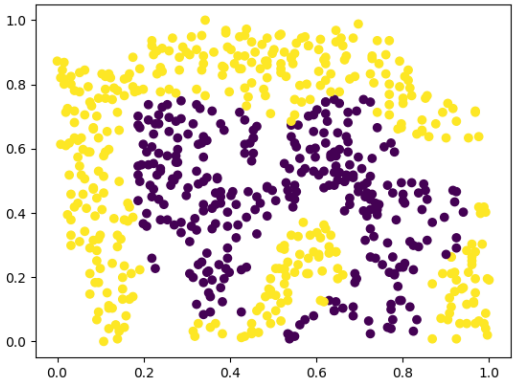
[3] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation". In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30.



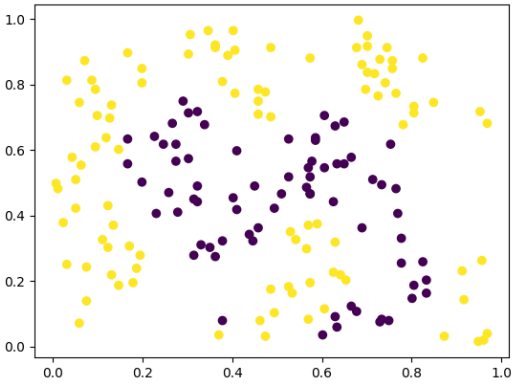
(a) Todos os dados (863 instâncias).



(b) 25 grupos retornados pelo `k-means`.



(c) Classificação do conjunto de treinamento.



(d) Classificação do conjunto de teste.

Figure 4: Conjunto de dados, e diferentes métricas usando 25 neurônios na camada escondida.