Exercício 5

Frederico Schardong

1 Enunciado do trabalho

O enunciado do exercício propõem a criação de uma rede neural com objetivo de aprender e classificar dígitos manuscritos. Os parâmetros da rede devem ser explorados pelo aluno. O conjunto de dados consiste em 5000 instâncias com 400 dimensões cada, onde cada instância é uma imagem de tamanho 20x20 representando os dígitos em escala de cinza e a respectiva categoria (números de 0 a 9, onde o 0 representa o número 10).

2 Resolução do Trabalho

Nesta seção são apresentados: (i) as diferentes formas de normalização dos dados testadas em redes rasas; (ii) redes profundas; e (iii) detalhes de implementação para reproduzir os experimentos aqui reportados.

Em todos os experimentos conduzidos neste trabalho, 80% das instâncias, ou seja, 4000 imagens, foram aleatoriamente selecionadas para treinamento da rede, e as restantes 1000 para teste. Por fim, para treinar e medir o desempenho de todas as redes, foram utilizadas a métrica `f1-score`, que é a média harmônica da acurácia e da precisão, e 5-*fold cross-validation* para evitar bias durante o treinamento.

2.1 Redes rasas

A distribuição dos valores para todas as imagens utilizadas no treinamento (80% do total, ou seja, 4000 imagens) são assimétricas, como pode ser visto na Figura 1. Com o objetivo de selecionar as melhores funções de normalização para o conjunto de treinamento em mãos, diferentes métodos de normalização foram experimentados: (i) `min max` transforma os dados de entrada para o intervalo $[0; 1]$, considerando os valores mínimo e máximo, mas não modifica a distribuição os dados (Figura 2a); (ii) `max abs` transforma os dados de entrada para o intervalo $[-1; 1]$, considerando apenas o valor máximo, também não alterando a distribuição os dados (Figura 2b); (iii) `l2-normalization` altera cada entrada tal que a distância euclidiana do valor normalizado e do valor original seja igual a 1 (Figura 2c); (iv) `(x - u) / s` transforma cada entrada x em $(x - u)/s$ onde u é a média dos 400 valores da instância e s o desvio padrão (Figura 2d); e por fim duas técnicas de normalização que utilizam quartil com o objetivo de (vi) uniformizar ou (vii) normalizar a distribuição de valores (Figuras 2e e 2f). O valor mínimo, máximo, médio e desvio padrão dos dados, bem como após as diferentes técnicas de normalização testadas são listados na Tabela 1.

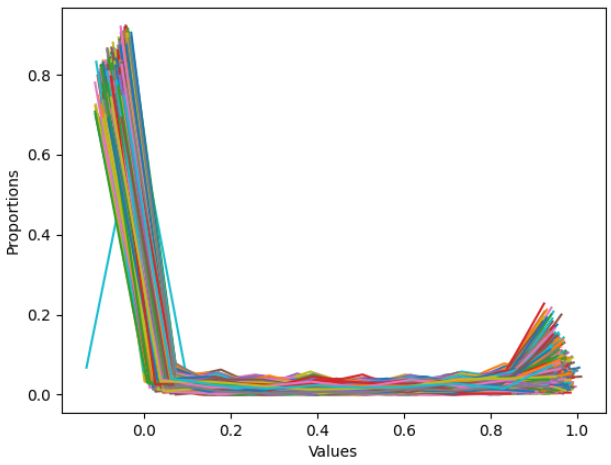
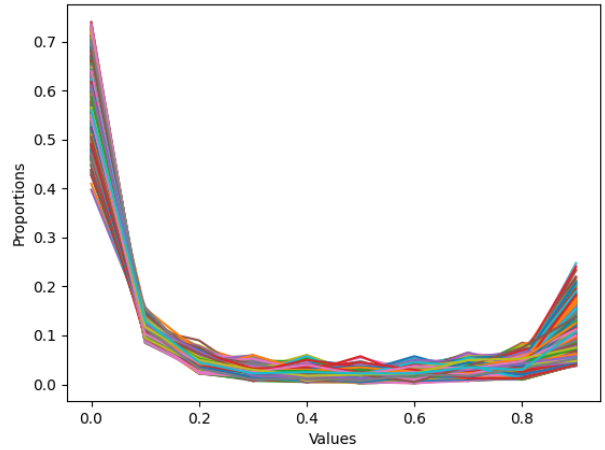
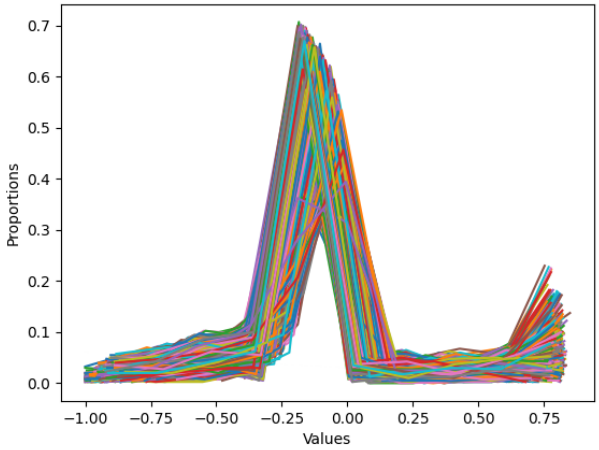


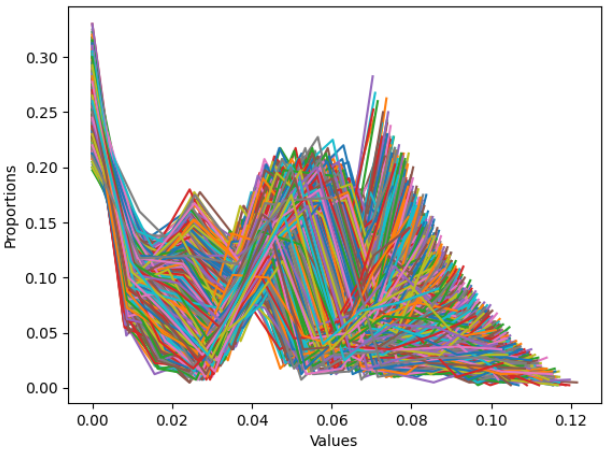
Figure 1: Histograma dos valores de treinamento antes da normalização.



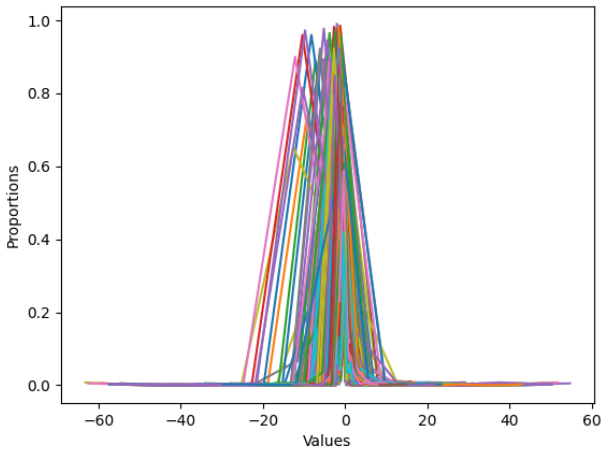
(a) Normalização `min max`.



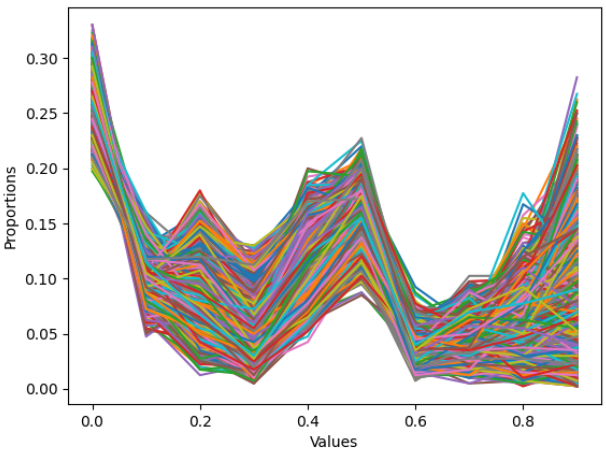
(b) Normalização `max abs`.



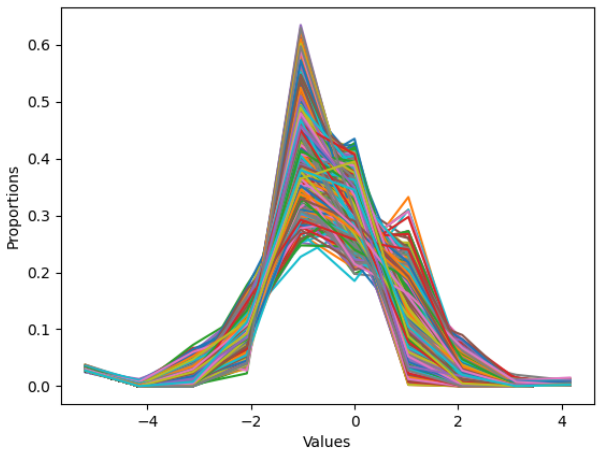
(c) Normalização `l2-normalization`.



(d) Normalização `(x - u) / s`.



(e) Normalização `quantil-uniform`.



(f) Normalização `quantil-normal`.

Figure 2: Histograma dos dados normalizados através de diferentes métodos.

Para descartar as técnias de normalização que produzem resultados insatisfatórios, os dados de treinamento normalizados com cada método supracitado foram aplicados à várias redes razas (1 camada com 100 neurônios) com diferentes configurações, utilizando 5-*fold cross-validation*. A biblioteca `scikit-learn` [3], utilizada para implementar a rede, fornece

	Mínimo	Máximo	Média	Desvio Padrão
Sem normalização	-0.13	1.13	0.13	0.29
`min max`	0.00	1.00	0.24	0.31
`max abs`	-1.00	1.00	0.00	0.33
`l2-normalization`	0.00	0.14	0.04	0.03
`(x - u) / s`	-63.24	63.24	0.00	0.99
`quantil-uniform`	0.00	1.00	0.39	0.30
`quantil-normal`	-5.20	5.20	-0.13	1.22

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

a classe `GridSearchCV` que recebe como parâmetro as possíveis configurações de uma rede neural e executa cada possível combinação de configuração, utilizando nesta implementação a métrica `f1` para comparar o desempenho das diferentes configurações. Esta métrica foi escolhida pois é a média harmônica da precisão e acurácia [1], produzindo valores no intervalo $[0; 1]$. A permutação de todos os parâmetros detalhados na Tabela 2 foram executados para cada normalizador supracitado.

Parâmetro	**Possíveis Valores**
Função de ativação	Tangente Hiperbólica (tanh), Unidade Linear Retificada (relu), Sigmóide, Linear
Solucionador/Algorítmo	Gradiente Descendente (sgd), Adam, lbfgs
Taxa de aprendizado	0.00001, 0.0001, 0.001, 0.01
Limite de Épocas	100
Neurônios na camada escondida	100

Table 2: Todas as configurações da rede testadas.

O melhor resultado encontrado para cada normalizador usando todas as possíveis configurações listadas na Tabela 2 são listados na Tabela 3. Os normalizadores `min max` e `max abs` foram selecionados para serem utilizados nas próximas etapas (criação de rede profunda) pois produziram os maiores `f1-score`. Os outros normalizadores foram descartados.

Normalizador	`f1-score`	**Configuração**
`min max`	0.94	Função de ativação = sigmóide, $\alpha = 0.01$, Algorítmo: lbfgs
`(x - u) / s`	0.93	Função de ativação = relu, $\alpha = 0.01$, Algorítmo: adam
`max abs`	0.94	Função de ativação = relu, $\alpha = 0.01$, Algorítmo: adam
`quantil-normal`	0.91	Função de ativação = relu, $\alpha = 0.0001$, Algorítmo: adam
`quantil-uniform`	0.92	Função de ativação = sigmóide, $\alpha = 0.01$, Algorítmo: lbfgs
`l2-normalization`	0.89	Função de ativação = sigmóide, $\alpha = 0.00001$, Algorítmo: lbfgs

Table 3: Melhor resultado de cada normalizador e a respectiva configuração da rede.

2.2 Redes profundas

Os normalizadores `min max` e `max abs` foram aplicados a diferentes configurações de redes profundas completamente conectadas com a mesma quantidade de neurônios que o experimento anterior (100 neurônios) Foram testadas as configurações: (i) 10 camadas de 10 neurônios; (ii) 5 camadas de 20 neurônios e; (iii) 4 camadas de 25 neurônios. Cada configuração supracitada foi aplicada aos três algoritmos de treinamento usados anteriormente (`lbfgs`, `sgd`, `adam`). Por fim, diferentes taxas de aprendizado (0.00001, 0.0001, 0.001 e 0.01) e funções de ativação (sigmóide e tangente hiperbólica) foram testadas para cada configuração mencionada e o melhor resultado (maior `f1-score`) é apresentado na Tabela 4.

Como os dois normalizadores apresentaram desempenho igual para 17 dos 18 testes, apenas o `min max` foi utilizado nos próximos experimentos. Considerando a insignificante diferença entre 5 camadas de 20 neurônios e 4 camadas de 25 neurônios, o próximo esperimento consistiu em testar os três algorítmos de treinamento em 7 redes com 1, 2, ..., 7 camadas de 25 neurônios cada. Como pode ser visto na Tabela 5, o `f1-score` de todos os algorítmos permanece virtualmente inalterado independente da profundidade da rede.

Por fim, a matriz de confusão do melhor resultado das redes profundas (`f1-score` de 3 camadas com 25 neurônios usando `adam`, $\alpha = 0.01$, função de ativação = tangente hiperbólica) é mostrado na Figura 3.

Camadas \ Algorítmo	lbfgs		sgd		adam	
	min max	max abs	min max	max abs	min max	max abs
10 camadas de 10 neurônios	0.45	0.46	0.74	0.74	0.87	0.87
5 camadas de 20 neurônios	0.90	0.90	0.85	0.85	0.92	0.92
4 camadas de 25 neurônios	0.90	0.90	0.87	0.87	0.91	0.91

Table 4: `f1-score` de cada rede para os dois normalizadores previamente selecionados.

Camadas \ Algorítmo	lbfgs	sgd	adam
1 camada	0.90	0.87	0.92
2 camadas	0.90	0.87	0.92
3 camadas	0.92	0.88	0.93
4 camadas	0.90	0.87	0.91
5 camadas	0.90	0.85	0.92
6 camadas	0.93	0.86	0.91
7 camadas	0.89	0.85	0.91

Table 5: `f1-score` de cada rede para os dois normalizadores previamente selecionados.

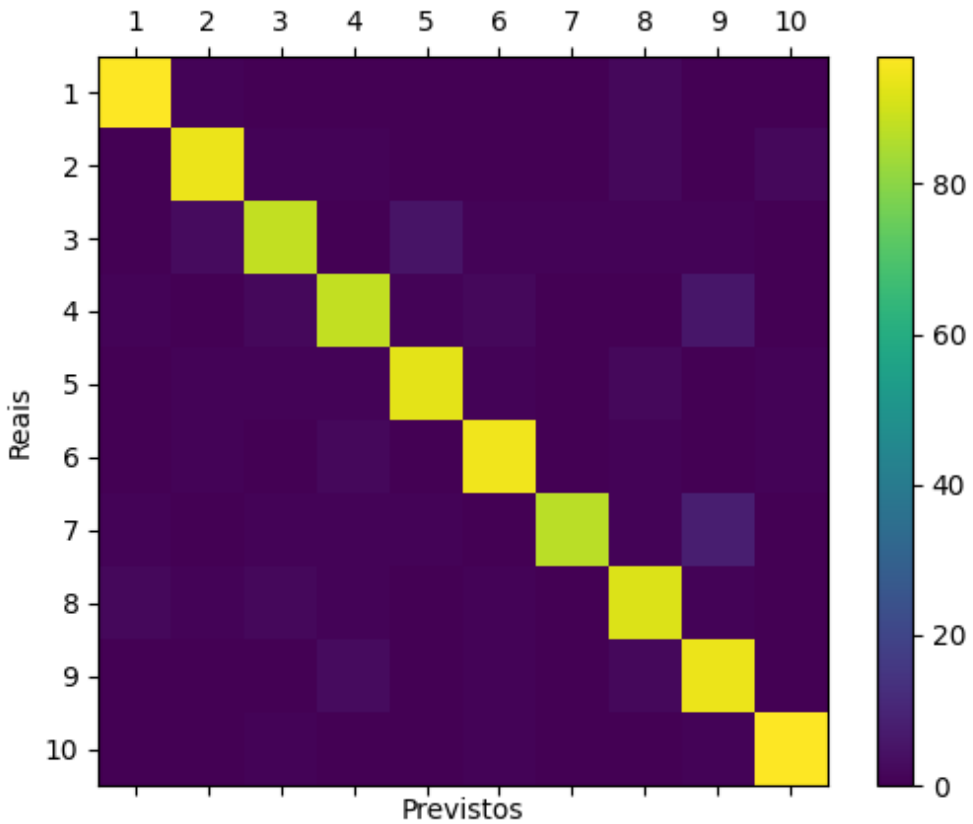


Figure 3: Matriz de confusão do melhor resultado das redes profundas.

2.3 Detalhes de implementação

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

- `scipy` [5] para carregar o dataset do arquivo `exdata.mat`;
- `scikit-learn` [3] para: (i) normalizar os dados; (ii) gerar a permutação de todas as configurações de teste; e (iii) criar a rede neural, treiná-la e testá-la;
- `numpy` [4] para obter os valores mínimos, máximos, média e desvio padrão antes e depois das normalizações;
- `matplotlib` [2] para geração dos gráficos dos histogramas e matriz de confusão;

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

```
$ pip install scipy 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
```

3 Considerações Finais

Dezenas de diferentes parâmetros foram utilizados para classificar instâncias do conjunto de dígitos manuscritos do `MNIST`, usando 80% dos dados para treinamento e 20% para teste. Para cada configuração da rede, os 80% dos dados usado no treinamento foram subdivididos em 5 para que cada teste fosse realizado usando 5-*fold cross-validation*.

Com os testes executados em redes rasas, é possível observar que as diferentes funções de normalização não tiveram grande impacto neste conjunto de dados. Em relação às redes de múltiplas camadas foi observado que uma rede de 10 camadas de 10 neurônios teve desempenho inferior a uma rede de 4 camadas de 25 neurônios, bem como diferentes quantidades de camadas de 25 neurônios tiveram desempenho semelhante.

References

[1] Tsong Yueh Chen, Fei-Ching Kuo, and Robert Merkel. "On the statistical properties of the f-measure". In: *Fourth International Conference onQuality Software, 2004. QSIC 2004. Proceedings.* IEEE. 2004, pp. 146–153.

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

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

[4] 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.

[5] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: `https://doi.org/10.1038/s41592-019-0686-2`.

[1] Código fonte disponível em `https://github.com/fredericoschardong/programming-exercise-5-MNIST-database-hyper-parameterization`.