Lista 3 - Fundamentos em Redes Neurais e Aprendizagem Estatística

Lorran de Araújo Durães Soares*

2024

Introdução

Este documento refere-se à elaboração da terceira lista de exercícios da disciplina de Fundamentos de Redes Neurais e Aprendizagem Estatística, promovida pelo Laboratório Nacional de Computação Científica (LNCC) sob a orientação do professor Gilson Antonio Giraldi. A lista consiste em três questões, que serão apresentadas neste formato de artigo, detalhando o passo a passo necessário para a resolução e implementação de cada uma delas. Todas as questões foram implementadas na linguagem Python, utilizando notebooks do tipo iPynb e empregando bibliotecas como `Keras` (CHOLLET et al., 2024), `Pandas` (TEAM, 2024b), `Matplotlib` (TEAM, 2024a), `Numpy` (DEVELOPERS, 2024a), `Scikit-learn` (DEVELOPERS, 2024b), `OpenCV` (OpenCV Team, 2024), `Seaborn` (WASKOM et al., 2024) e `PIL` (CONTRIBUTORS, 2024). As implementações de todas as questões serão disponibilizadas ao final da explicação de cada uma delas neste documento.

Para a realização de todas as questões, foi utilizada a base de imagens CIFAR-10, fornecida pelo `Keras`. Esse conjunto de dados é composto por 60.000 imagens coloridas de dimensão 32x32 pixels, distribuídas em 10 classes. Entretanto, foi realizada uma filtragem para trabalhar com a classificação de apenas duas classes de imagens, aviões e carros, reduzindo então o conjunto para 12.000 imagens. Algumas dessas imagens são ilustradas na Figura 1. Além disso, devido a limitações de hardware, nos exercícios 1 e 3, foram utilizados apenas 1.200 das 12.000 imagens.

*lorranspbr@gmail.com

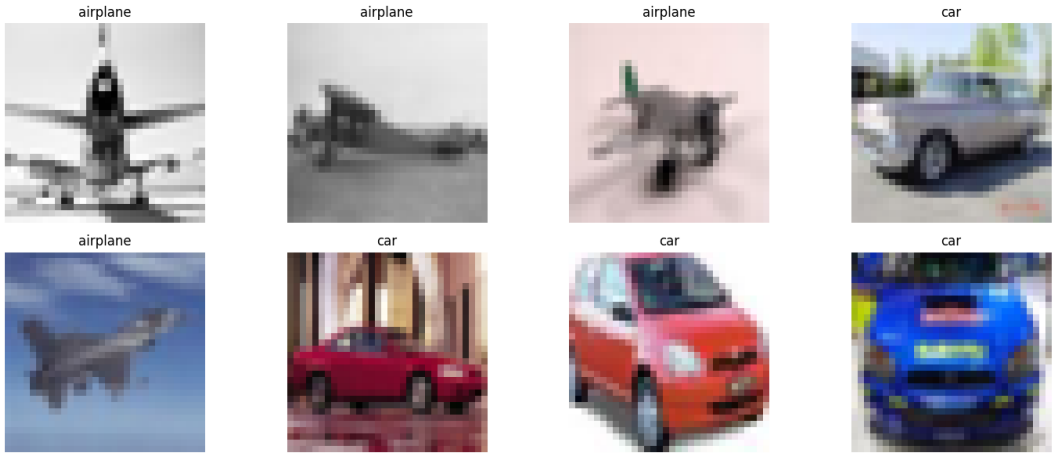

Figura 1 – Amostra do banco de imagens

Esse banco de imagens foi importado da classe `datasets.cifar10` da biblioteca `Keras` através da função `load_data()`. A escolha de usar um mesmo banco de dados em todas as questões se objetivou em realizar uma comparação entre as diferentes estratégias de classificação que cada exercício pede.

Pré-processamento

Para a realização dos exercícios 1 e 3, foi realizado um pré-processamento dos dados, necessário para a execução das operações nessas questões e para obter uma melhor acurácia nos resultados.

As seguintes operações foram realizadas:

- Conversão para escala de cinza: realizada através da biblioteca `OpenCV`, utilizando a função `cvtColor`;
- Vetorização das imagens: realizada através do método `reshape` da biblioteca `Numpy`;
- Normalização das features: realizada através da ferramenta `StandardScaler` da biblioteca `Scikit-Learn`. O `StandardScaler` ajusta cada feature do conjunto de dados de acordo com a fórmula:

$$z = \frac{x - \mu}{\sigma}$$

  Onde:

  - x é o valor original da feature.
  - μ é a média dos valores da feature.
  - σ é o desvio padrão da feature.

Após isso, através da ferramenta `train_test_split` da biblioteca `Scikit-Learn`, o conjunto de dados foi separado com proporção de 70% para treinamento e 30% para teste. Além disso, com o intuito de realizar a comparação sob o mesmo conjunto de dados para todas as questões, foi selecionada a opção `random_state` = 42 na função utilizada para a separação dos dados, fixando a semente de sorteio dos dados. Na implementação de cada uma das questões foi realizado o cálculo da quantidade de imagens

de cada classe no conjunto de treinamento, com o intuito de observar possíveis desbalanceamentos de dados. Como mostra a Figura 2, as duas classes tinham números de dados iguais, não sendo necessário, portanto, a exclusão ou adição de novos dados no conjunto de treinamento para a realização dos exercícios.

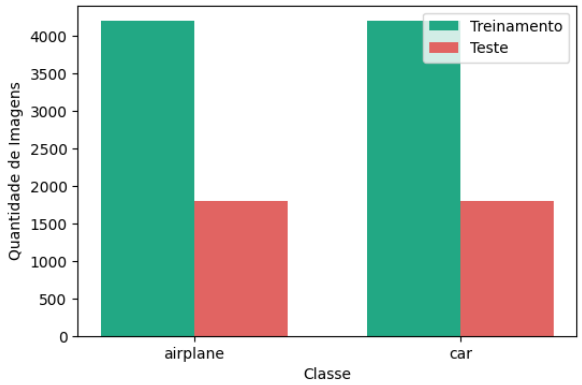

Figura 2 – Histograma da quantidade de imagens por classe para treinamento e teste

Questão 1

1. *Considere um banco de dados de imagens e um problema de classificação. Aplique validação cruzada leave-one-out multi-fold explicada na seção 8.5 de (GIRALDI, 2024), com $K = 4$, e SVM como segue:*

   a) SVM Linear não separável com espaço de características obtido através do KPCA.

   b) SVM Kernel não separável com espaço de características obtido através do PCA.

   c) Compare os resultados dos itens (a) e (b).

Para a realização deste exercício, especificamente para o proposto na letra (a), foi inicialmente realizado o cálculo do KPCA através da biblioteca `Scikit-Learn` com a função `KernelPCA`. Foi escolhido um kernel polinomial para o cálculo da matriz KPCA com todas as componentes, para posteriormente selecionar as principais componentes. A projeção do conjunto nas duas primeiras componentes do KPCA está ilustrada na Figura 3.

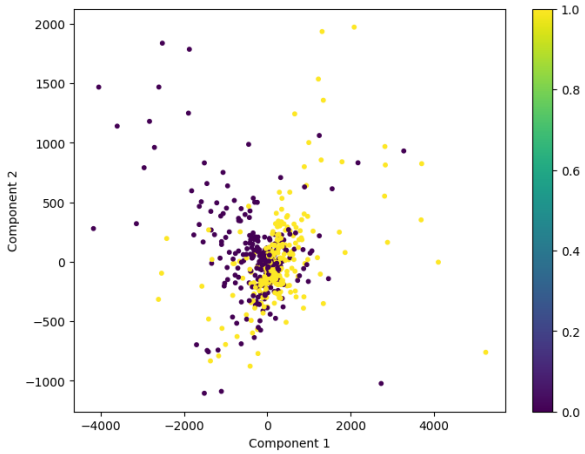

Figura 3 – Projeção do conjunto de treinamento na base obtida pelo KPCA

Após o cálculo, foi plotado o gráfico da variância explicada cumulativa, mostrado na Figura 4. Para a escolha das componentes principais, utilizando a biblioteca `Numpy` com a função `argmax`, foi calculado o número de componentes necessário para explicar no mínimo 95% da variância dos dados, resultando em 435 componentes. A partir disso, a matriz KPCA calculada anteriormente foi reduzida para este número de componentes.

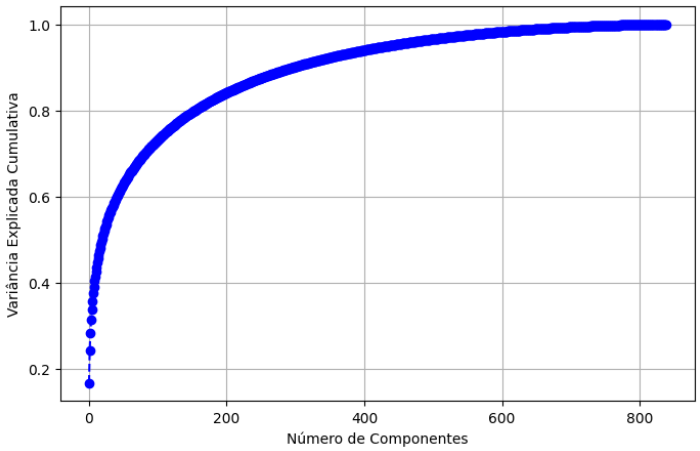

Figura 4 – Gráfico da variância explicada cumulativa por componentes KPCA

Utilizando então um `K-fold` = 4 (explicar melhor), através da função `SVC` da biblioteca `Scikit-Learn`, foi construído o SVM, com o objetivo de performar a classificação dos dados. O resultado da acurácia para cada fold sobre os conjuntos de validação e de testes está presente na Tabela 1. Em média, pode-se observar que o classificador construído obteve maior acurácia no conjunto de testes do que no de validação.

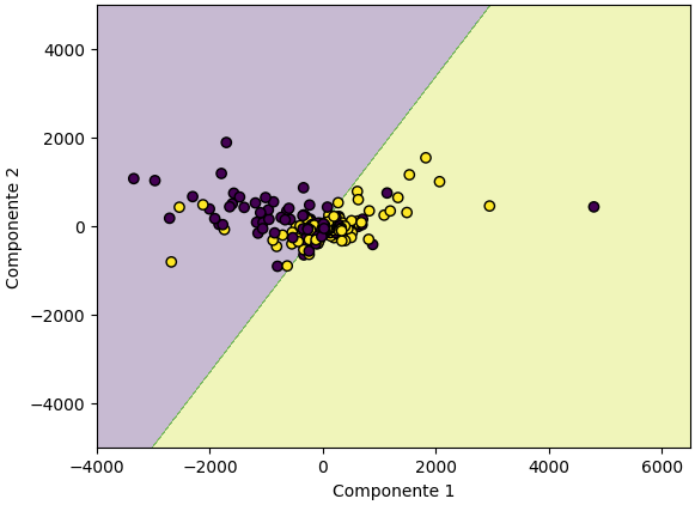

Figura 5 – Reta de decisão do SVM Linear com KPCA

Conjunto	Fold 1	Fold 2	Fold 3	Fold 4	Média
Validação	0.60	0.69	0.57	0.60	0.615
Teste	0.72	0.74	0.71	0.72	0.7225

Tabela 1 – Acurácia da classificação do SVM Linear com o KPCA

A fim de promover uma visualização gráfica para exemplificar como seria o plano de separação para apenas as duas primeiras componentes obtidas pelo cálculo do KPCA, foi realizado o treinamento de um novo SVM linear, obtendo a reta de separação mostrada na Figura 5, juntamente com a projeção do conjunto de treinamento nessas duas direções. Neste caso, a acurácia média obtida no conjunto de testes foi de 61%, inferior à de 72% obtida anteriormente com todas as componentes principais, evidenciando que a redução de dimensionalidade para apenas duas componentes principais impactou na redução da performance do classificador.

De maneira parecida como feito anteriormente para o item a, agora especificamente referente ao item (b), foi inicialmente realizado o cálculo do PCA utilizando a biblioteca `Scikit-Learn` com a função `PCA`. Neste primeiro passo, foram calculadas todas as componentes principais, permitindo posteriormente a seleção das mais relevantes. A projeção do conjunto de dados na base PCA está representada na Figura 6.

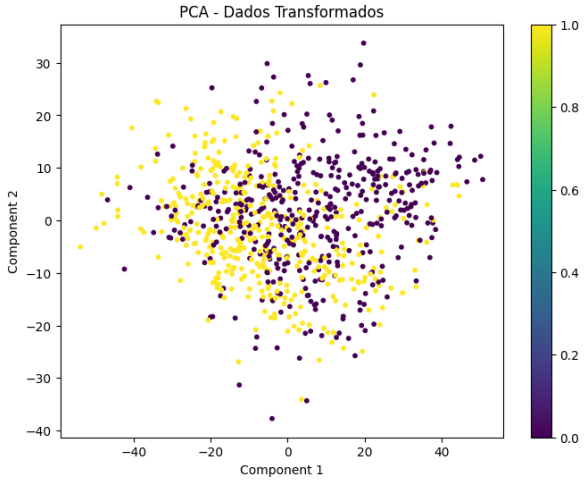

Figura 6 – Projeção do conjunto de treinamento na base PCA

Após o cálculo das componentes principais, foi gerado o gráfico da variância explicada cumulativa, fornecido pelo próprio cálculo do PCA através do método `explained_variance_ratio_`. O resultado obtido está ilustrado na Figura 7. Para identificar as componentes principais, foi utilizada a função `argmax` da biblioteca `Numpy`, determinando o número de componentes necessário para explicar pelo menos 95% da variância dos dados, resultando em 123 componentes. Com isso, a matriz PCA foi reduzida para conter apenas essas componentes.

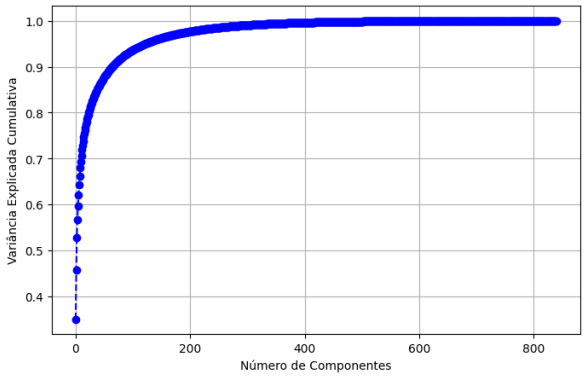

Figura 7 – Gráfico da variância explicada cumulativa para as componentes do PCA

Em seguida, foi aplicado um `K-fold` com valor 4 (explicar melhor), e, utilizando a função `SVC` da biblioteca `Scikit-Learn`, foi construído um SVM com kernel polinomial, com o objetivo de performar a classificação dos dados, obtendo as acurácias mostradas na Tabela 2 para os conjuntos de validação e de teste.

Conjunto	Fold 1	Fold 2	Fold 3	Fold 4	Média
Validação	0.60	0.69	0.57	0.60	0.615
Teste	0.72	0.74	0.71	0.72	0.7225

Tabela 2 – Acurácia da classificação do SVM Polinomial com o PCA

Com o intuito de visualizar o plano de separação utilizando apenas as duas primeiras componentes obtidas pelo PCA, foi treinado um novo SVM polinomial considerando apenas as duas primeiras componentes principais obtidas pelo PCA, e a curva de separação foi gerada, como mostrado na Figura 8. Essa figura também mostra a projeção do conjunto de treinamento nas duas primeiras direções principais. Nesse cenário, a acurácia média alcançada foi de 61%.

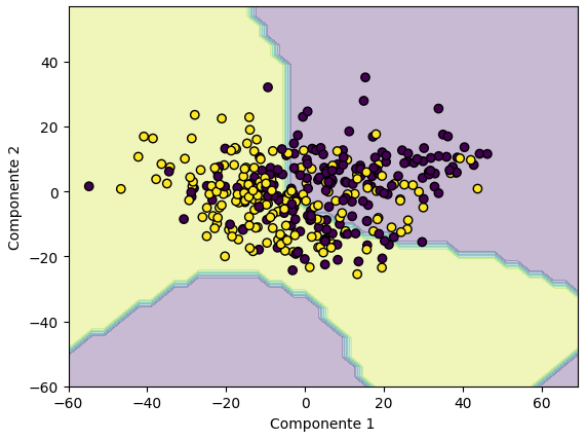

Figura 8 – Curva de decisão do SVM Polinomial com PCA

Falta falar da comparação entre elas.

Questão 2

2. Considere um banco de dados e um problema de classificação. Aplique a validação cruzada multi-fold leave-one-out explicada na seção 8.5 de (GIRALDI, 2024), com $K = 5$, para um modelo de CNN. Utilize as facilidades disponíveis em bibliotecas para a implementação de redes neurais, como Keras, TensorFlow, etc.

(a) Mostre a representação gráfica da evolução das etapas de treinamento e validação (ver Figura 8.8 da monografia do curso).

(b) Realize uma análise estatística do desempenho (seção 8.6) dos cinco modelos aplicados sobre a $\mathbb{D}_{te}$.

Para a realização deste exercício, foram utilizadas todas as 12.000 imagens da base de dados CIFAR-10 referentes às classes aviões e carros, com o objetivo ter mais dados para o treinamento.

A rede CNN construída para a realização deste exercício teve arquitetura inspirada na rede LeNet-5, desenvolvida para a classificação de imagens, como dígitos numéricos (LECUN et al., 1998).

Como mostra a tabela **??**, a rede construída possui 3 camadas convulucionais, 2 de pooling, 1 de vetorização e 2 camadas MLP densas para a realização da classificação.

Além disso, para a determinação dos hiperparâmetros do otimizador, foi realizado um *grid search* por meio da biblioteca `Scikit-Learn` utilizando a ferramenta `GridSearchCV`. Os hiperparâmetros considerados estão presentes na Tabela **??**.

Após este processo, foi determinado que os melhores hiperparâmetros são:

Para o treinamento do modelo, utilizando a biblioteca `Keras`, foi considerada uma estratégia de *early stopping* da classe `callbacks` da função `fit` para evitar o *overfitting*. Considerando a métrica de *val_accuracy* como a quantidade a ser monitorada no *early stopping*, temos que os seguinte parâmetros para este critério de parada foram considerados:

- $min\_delta = 0.01$: quantidade mínima para ser considerada como melhoria;
- $patience = 4$: número de épocas sem melhorias a quais o treinamento será interrompido;
- $restore\_best\_weights$ = True: restaura para a rede os pesos referentes à epoca com melhor resultado na métrica escolhida.

Então, foi realizado o treinamento do modelo usando $K - Fold = 5$, como determinado pela questão. A evolução da acurácia e da loss, para o treinamento e a validação, estão presentes nos gráficos das figuras **??**. Através do gráfico, podemos perceber que a estratégia do *early stopping* funcionou em evitar o *overfiiting*.

Com as ferramentas `confusion_matrix` e `classification_report`, presentes na biblioteca `Scikit-Learn`, podemos então calcular as métricas estátisticas descritas na seção 8.6 de (GIRALDI, 2024), obtendo então os resultados, sobre o conjunto de teste, presentes na tabela tal.

Como pode ser observado, o modelo de $K - fold$ igual a tal teve o melhor resultado, mas em média, os modelos tiveram acurácias parecidas.

Questão 3

Considere um banco de dados de imagens e um problema de classificação.

(a) *Aplique a validação cruzada multi-fold leave-one-out explicada na seção 8.5 de (GIRALDI, 2024), com $K = 4$ usando LDA no espaço reduzido de PCA e realize a classificação sobre o conjunto de teste. Analise os resultados.*

(b) *Aplique a validação cruzada multi-fold leave-one-out explicada na seção 8.5 de (GIRALDI, 2024), com $K = 4$ e SVM Kernel não separável com espaço de características obtido através da Análise Discriminante de Componentes Principais (DPCA).*a

(c) *Compare os resultados obtidos nos itens (a) e (b) acima.*

Para a realização da letra (a), foi então calculado o PCA e efetuada a redução de dimensionalidade de forma análoga à realizada anteriormente na questão 1, na letra (c). Mas, ao invés de usar um Kernel SVM para realizar a classificação dos dados, foi realizado o treinamento do LDA, com K-fold = 4, com o número de componentes iguais a 1 (pois temos apenas duas classes) sobre o conjunto obtido, através da biblioteca `Scikit-Learn`. A acurácia obtida na classificação do conjunto de validação e de teste através do LDA está descrita na tabela 3.

Conjunto	Fold 1	Fold 2	Fold 3	Fold 4	Média
Validação	0.74	0.74	0.64	0.75	0.7175
Teste	0.75	0.71	0.76	0.75	0.7425

Tabela 3 – Acurácia da classificação do LDA com o PCA

Referências

CHOLLET, F. et al. *Keras documentation*. 2024. <https://keras.io/>. Accessed: 2024-08-22. Citado na página 1.

CONTRIBUTORS, P. *Pillow documentation*. 2024. <https://pillow.readthedocs.io/>. Accessed: 2024-08-22. Citado na página 1.

DEVELOPERS, N. *NumPy documentation*. 2024. <https://numpy.org/doc/>. Accessed: 2024-08-22. Citado na página 1.

DEVELOPERS, S. learn. *Scikit-learn documentation*. 2024. <https://scikit-learn.org/stable/>. Accessed: 2024-08-22. Citado na página 1.

GIRALDI, G. A. Fundamentals of neural networks and statistical learning. 2024. Citado 3 vezes nas páginas 3, 7 e 8.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Ieee, v. 86, n. 11, p. 2278–2324, 1998. Citado na página 7.

OpenCV Team. *OpenCV documentation*. 2024. <https://opencv.org/>. Accessed: 2024-08-22. Citado na página 1.

TEAM, M. D. *Matplotlib documentation*. 2024. <https://matplotlib.org/stable/contents.html>. Accessed: 2024-08-22. Citado na página 1.

TEAM, P. D. *Pandas documentation*. 2024. <https://pandas.pydata.org/docs/>. Accessed: 2024-08-22. Citado na página 1.

WASKOM, M. et al. *Seaborn: statistical data visualization*. 2024. <https://seaborn.pydata.org/>. Accessed: 2024-08-22. Citado na página 1.