# Trabalho 3 - MC886 Aprendizado de Máquina

Leo Yuuki Omori Omi
138684
leoyuuki@gmail.com

João Pedro Ramos Lopes
139546
jpedrorl@gmail.com

## I. Introdução

Neste trabalho, tivemos que agrupar vários textos utilizando utilizando técnicas de Aprendizado não-supervisionado. Utilizando dados em formato *bag-of-words* utilizamos algumas técnicas para tentar achar agrupamentos que fizessem algum sentido.

### A. Soluções Propostas

*1) DBSCAN:* Tentamos utilizar a técninca Density-Based Spatial Clustering of Applications with Noise (DBSCAN), já implementado na biblioteca Scikit Learn. Tentamos variar os parâmetros *eps* e *min_samples*, no entanto, inicialmente, não conseguimos achar uma configuração que desse um resultado que parecia coerente, portanto desistimos de continuar experimentos com esta técnica. Possivelmente, a distribuição dos dados não favorecia o DBSCAN ou mesmo foi uma escolha infortúnia do parêmtros.

*2) Hierarchical Clustering:*

*3) K-means:*

### B. Experimentos

*1) Modelos iniciais:* Para analisar o número de grupos presentes no problema, utilizamos como modelo base o Elbow Method, variando esse número no intervalo [10, 1000] com passo 10 encontramos o gráfico disposto na figura 1.

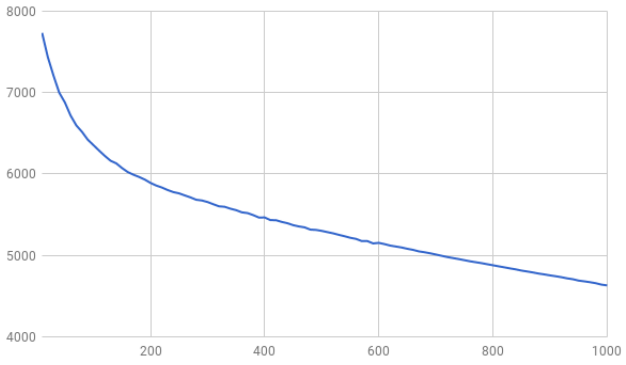


Figure 1. Número de Clusters x Erro médio

Utilizando o modelo de k-médias com 3 reinícios, 10 iterações e inicialização aleatória *k-means++*. Com base no gráfico gerado, escolhemos o range de classes [50, 400] a ser submetido ao método de silhueta. Utilizando passo 50 encontramos o gráfico da figura 2, que representa a pontuação para tais valores de k.

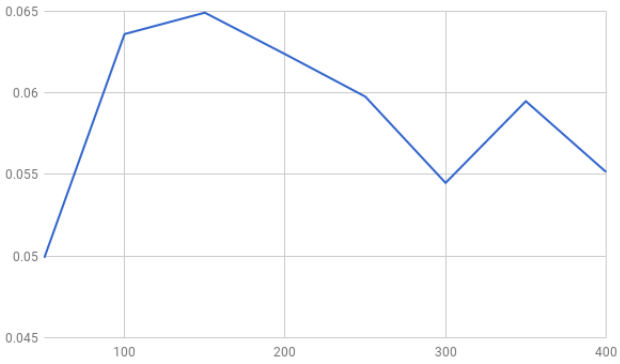


Figure 2. Silhueta no intervalo [50, 400] com passo 50

Finalmente, para determinar o número de classes a ser utilizado pelo modelo, foi percorrido o intervalo [125, 175] e desenhado o gráfico da pontuação encontrada pelo método de silhueta, representado na figura 3. Dessa forma, foram emscolhidas 151 clusters para o problema.

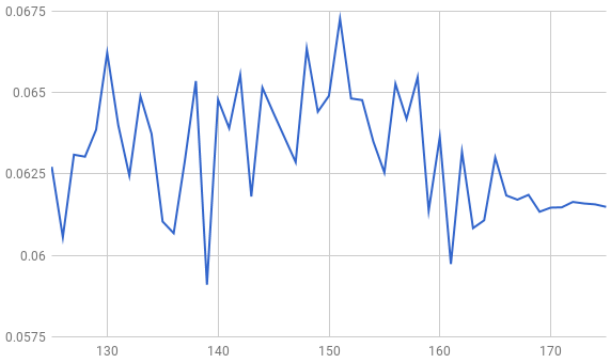


Figure 3. Silhueta no intervalo [125, 175]

*2) Solução Base K-Means:* Para analisar o modelo final do k-means, utilizou-se a mesma configuração que anteriormente, com máximo de 300 iterações e $k = 151$. Para checar os resultados encontrados, escolheu-se um agrupamento aleatório e ordenou-se os pontos pertencentes a ele pela distância euclideana. De forma amostral, checamos alguns

clusters e percebemos que o assunto geral dos documentos mais próximos foram:

- Cluster 0: pedidos de ajuda e sugestões/dicas em diversos tópicos (carros, hardware
- Cluster 17: discussões relacionadas a *Hockey*
- Cluster 31: 3 grupos principais: carros, esportes e eletrônicos
- Cluster 42: Uma longa discussão sobre o mesmo assunto: *Who's next? Mormons and Jews?*
- Cluster 67: Questões relacionadas a hardware/software da Microsoft/Apple
- Cluster 123: Alguns documentos são relacionados a vendas, outros relacionados a direção de veículos

*3) Solução Base PCA:* A solução utilizando o PCA, deu resultados bons. Foi escrito um programa rodando em um laço com valores crescentes para o número de compenentes, começando com 15 componentes e aumentando em 16 em cada iteração. A execução do problema é lenta, mas nos trouxe resultados que puderam ser aproveitados. Por volta de 40 componentes, obtemos uma acurácia de 30% tanto para o conjunto de treino como o conjunto de validação, o que foi o melhor resultado observado utilizando a regressão logística. Como obtemos uma acurácia semelhante entre os dois conjuntos, poderíamos dizer que não está ocorrendo *overfitting*, diferente das soluções anteriores. Além disso, como diminuímos a dimensionalidade do modelo, obtemos uma solução computacionalmente mais leve.

Table I
TABELA DE ACURÁCIA POR NÚMERO DE COMPONENTES

| Componentes | Acurácia Treino | Acurácia Validação |
|---|---|---|
| 15 | 0.29 | 0.29 |
| 31 | 0.3 | 0.3 |
| 47 | 0.3 | 0.3 |
| 63 | 0.31 | 0.3 |
| 79 | 0.31 | 0.3 |
| 95 | 0.31 | 0.3 |
| 111 | 0.31 | 0.3 |
| 127 | 0.31 | 0.3 |
| 143 | 0.32 | 0.3 |
| 159 | 0.32 | 0.3 |
| 175 | 0.32 | 0.3 |
| 191 | 0.32 | 0.3 |
| 207 | 0.32 | 0.3 |
| 223 | 0.32 | 0.3 |
| 239 | 0.32 | 0.3 |
| 255 | 0.32 | 0.3 |
| 271 | 0.32 | 0.29 |
| 287 | 0.32 | 0.29 |
| 303 | 0.32 | 0.29 |
| 319 | 0.32 | 0.29 |
| 335 | 0.32 | 0.29 |
| 351 | 0.33 | 0.29 |
| 367 | 0.33 | 0.29 |
| 383 | 0.33 | 0.29 |
| 399 | 0.33 | 0.29 |
| 415 | 0.33 | 0.29 |

*4) Solução Base (Multinomial):* Para esta solução, usando a regressão multinomial com PCA, obtivemos também uma acurácia de 30% para o conjunto de testes e 30% para o de validação. Um resultado semelhantes ao do One vs All, não obtendo uma melhora o resultado em comparação com a solução anterior.

## II. REDES NEURAIS

Redes Neurais agrupam métodos de aprendizado de máquina que foram originalmente baseados no comportamento do cérebro, onde neurônios fazem operações simples e a complexidade do sistema está na iteração entre os diversos neurônios. Apesar de os conceitos aplicados atualmente estarem distantes dos modelos cerebrais, Redes Neurais estão no Estado da Arte na solução de diversos problemas complexos.

Uma Rede Neural é dividida em 3 camadas, cada uma composta por uma matriz de pesos e uma função de ativação: *Input Layer*, *Hidden Layer* e *Output Layer*. A entrada é multiplicada pela primeira camada e submetida a função de ativação, onde sua saída é multiplicada pela matriz subsequente e assim por diante.

Existem diversas operações que podem substituir a multiplicação de matrizes, como uma operação Convolucional 2D (utilizado particularmente em problemas relacionados a Visão Computacional) em que filtros são aplicados de maneira convolucional a entrada.

*1) Soluções Propostas:* Para construir diversos modelos, consideramos os operadores *Dense(*$X$*)* que realiza a operação $output = dot(input, kernel) + bias$ utilizando $X$ neurônios; *Conv2D(32, (3, 3))*, que utiliza 32 filtros de aprendizado de tamanho $3x3$. Como operador de ativação, foi considerado o operador ReLU (*Rectified Linear Unit*) que aplica a operação $f(x) = max(x, 0)$.

Na camada de entrada e de saída foram utilizados os operadores *Dense(512* e *Dense(10)*, respectivamente. A saída foi ligada a um operador *softmax*, a fim de classificar as imagens nas 10 classes do problema. Na *Hidden Layer* foram feitas as composições de operadores sequenciais a seguir, cujos resultados encontram-se nas imagens indicadas. As duas últimas se tratam de camadas compostas.

- Dense(512) - Figura 4
- Conv2D(32, (3, 3)) - Figura 5
- Dense(512) & Conv2D(32, (3, 3)) - Figura 6
- Conv2D(32, (3, 3)) & Dense(512) - Figura 7

Por conta da dimensão do conjunto de treino, foram utilizadas apenas 10% das entradas. Então, o conjunto de treino foi dividido em treino e validação (utilizando uma proporção de 0.7) a fim de validar os modelos propostos. Os melhores resultados do conjunto de validação para cada modelo foram:

- Dense(512) - $0.4327$
- Conv2D(32, (3, 3)) - $0.4607$
- Dense(512) & Conv2D(32, (3, 3)) - $0.4433$
- Conv2D(32, (3, 3)) & Dense(512) - $0.4587$

Aplicando a rede neural encontrada nas melhores épocas para o modelo *Conv2D(32, (3, 3))* e para o modelo *Conv2D(32, (3, 3)) & Dense(512)* utilizando o conjunto de testes, encontrou-se uma acuracia de $0.4673$ e $0.4374$, respectivamente.

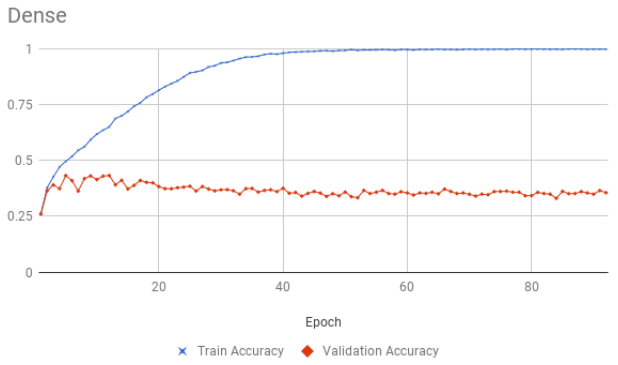


Figure 4. Acuracia para treino e validação

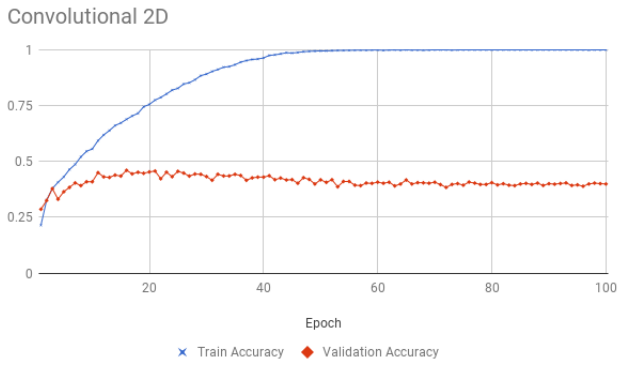


Figure 5. Acuracia para treino e validação

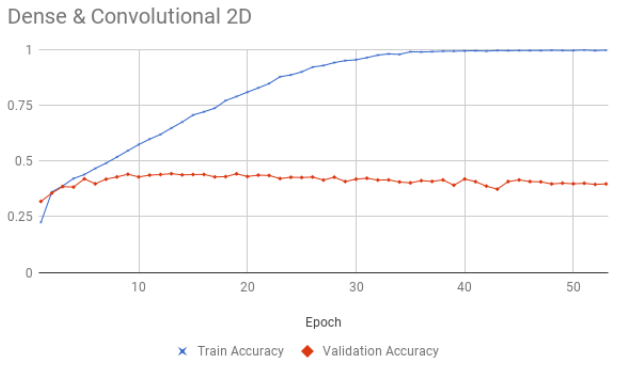


Figure 6. Acuracia para treino e validação

A maior acuracia encontrada pela rede *Conv2D(32, (3, 3))* pode ser atribuida a quantidade reduzida de dados utilizados, visto que por ser uma rede mais simples necessita de menor número de treinos para calibrar seus valores.

Finalmente, a fim de comparar a Ativação utilizando *ReLU*, executou-se o modelo *Conv2D(32, (3, 3)) & Dense(512)* utilizando como método de ativação a Tangente Hiperbólica: $f(x) = htan(x)$. A maior acurácia encontrada para o conjunto de validação foi de $0.3320$, apontando uma vantagem para a utilização do *ReLU*.

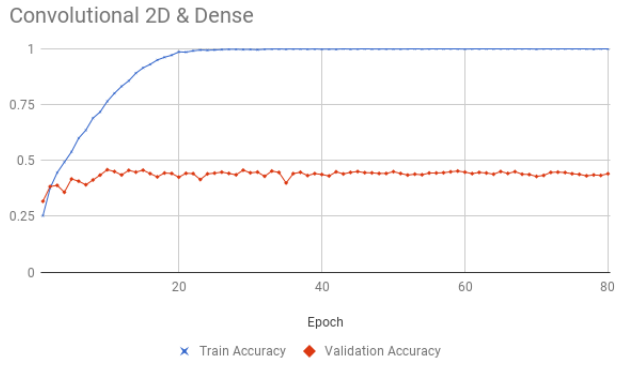


Figure 7. Acuracia para treino e validação

## III. Conclusões

Sobre as soluções propostas, aquelas que se utilizaram de processamento de imagem não trouxeram melhoras para o modelo. No entanto, a solução utilizando PCA trouxe melhoras tanto para o resultado quanto para tempo de computação do modelo, demonstrando a vantagem de usar a redução de dimensionalidade por este método.

Com relação a modelagem utilizando Redes neurais, poderia-se estudar a acuracia para um número maior de camadas, além da utilização mais estensiva de operadores convolucionais. Outra analise que pode ser feita em estudos posteriores é a utilização de Dropout, que utiliza da desativação de neurônios aleatórios de uma camada, evitando overfit e gerando redundâncias na rede neural.

## References

[1] Montgomery, D.C., Peck, E.A. and Vining, G.G., 2015. Introduction to linear regression analysis. John Wiley & Sons.

[2] Gendreau, M. and Potvin, J.Y., 2010. Handbook of metaheuristics (Vol. 2). New York: Springer.

[3] Haupt, R.L. and Haupt, S.E., 2004. Practical genetic algorithms. John Wiley & Sons.

[4] Djurisic, A.B., Elazar, J.M. and Rakic, A.D., 1997. Genetic algorithms for continuous optimization problems-a concept of parameter-space size adjustment. Journal of Physics A: Mathematical and General, 30(22), p.7849.