Relatório Trabalho 3 - Redução de dimensionalidade e Clustering

Gabriel Marinho da Costa
RA 172269
g172269@dac.unicamp.br

Giovanni Bertão
RA 173325
g173325@dac.unicamp.br

I. Introdução

Atualmente há uma quantidade exorbitante de dados disponíveis para serem coletados e utilizados, estes, provenientes de múltiplas fontes, muitas vezes são difíceis de serem interpretados. A partir deste empecilho, surgem técnicas específicas para solucionar o problema, duas delas serão tratadas neste trabalho. Uma delas se trata da redução de dimensionalidade, técnica que consiste em modificar os dados de forma que seja possível eliminar partes que não façam com que os dados percam sua característica. Desta forma, é possível ganhar tempo e espaço computacional e muitas vezes conseguir visualizar os dados, dado que o limite humano para visualização são 3 dimensões. Já a técnica de *Clustering* tem foco na busca de padrões nos dados, o que permite que os dados sejam reorganizados em classes, interpretados de uma maneira que faça sentido ou então representados de uma forma diferente. Ambas técnicas possuem diversas aplicações e no caso deste trabalho, focaremos na utilização combinada com redes neurais no conjunto Fashion-MNIST [1].

II. Atividades

Geralmente técnicas de *Clustering* são utilizadas quando não temos os dados anotados, entretanto, o conjunto de dados utilizado possui anotação, portanto iremos verificar o desempenho dos métodos utilizados comparando a classificação obtida com a real. Dados os métodos de redução de dimensionalidade vistos em aula [2], procuramos utilizar as versões já prontas do PCA [3] e de um Autoencoder [4] do Keras, já para a classificação, seguimos uma implementação do K-Means [5] e MiniBatch K-Means.

Notamos que há classes de imagem que podemos facilmente acabar confundindo, por exemplo alguns tipos de calçados ou então vestes de mangas compridas. Desta forma, não esperamos que a acurácia do modelo de classificação seja tão alta, mas procuramos verificar o desempenho de cada uma das técnicas separadamente antes de propor uma solução estratégica combinando-as.

III. Dataset

Fashion-MNIST [1] é uma base de dados de imagens de roupas, sendo 60 mil imagens para treino e 10 mil para teste, cada imagem possui dimensões 28x28 em escala de cinza e está associada à uma dentre 10 classes diferentes, que são enumeradas de 0 à 9 (t-shirt, trouser, pullover, dress e etc). A base de dados foi convertida em CSV, sendo que cada linha consiste da classe à que a imagem pertence, seguida por 784 valores entre 1 e 255, que correspondem aos pixels da imagem.

IV. Solução Proposta

Utilizando a mesma rede neural [6] como base para o teste das técnicas de redução de dimensionalidade, a solução consiste na tentativa de obter a melhor combinação utilizando uma das duas técnicas de classificação com uma das duas técnicas de redução de dimensão, para isso detalharemos a seguir as técnicas empregadas:

A. Rede Neural

A rede neural utilizada foi a convencional, temos apenas a camada *fully conected*, com função de ativação *ReLu* e na camada de saída função de ativação *Softmax*, que nos retorna a probabilidade atrelada à cada uma das 10 classes possíveis. O modelo pode ser conferido no sumário da Figura 1.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 128)               100480
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

Figura 1. Sumário do modelo utilizado na rede neural.

B. PCA

O método de análise de componente principal (PCA), trata-se de uma técnica para redução de dimensionalidade. Esta técnica consiste em detectar as características que mais descrevem o dado e desta forma manter apenas as K características mais descritivas. Exemplificando, suponhamos que um dado tenha 5 *features* e que selecionando apenas 2 delas, consigamos 90 por cento do resultado que obtivemos utilizando todas elas. Neste caso, estaríamos reduzindo a dimensão de 5 para 2 e ainda assim mantendo um bom resultado. Para isso 5 passos são realizados:

- Centralizar e normalizar os dados: transformar os valores de modo que fiquem na mesma escala e que os números estejam próximos à 0.
- Computar a matriz de covariância.
- Encontrar todos os autovalores e autovetores.
- Ordenar os autovetores e escolher os K primeiros.
- Projetar os dados nos K autovetores.

C. K-Means

Este método de aprendizado não supervisionado baseia-se em K conjuntos, previamente definidos, para separar os dados. Um centróide é um ponto que será colocado no espaço do conjunto de dados, colocaremos K pontos neste espaço e assim cada ponto do conjunto de dados será atribuído a um grupo (centróide) até que se obtenha a classificação. O algoritmo pode ser resumido nos seguintes passos:

- Definir os K centróides que serão utilizados.
- Para cada ponto do conjunto de dados, encontrar o centróide mais próximo e atualizar a classe correspondente.
- Mover os centróides para o centro do conjunto de pontos pertencentes à sua classe.
- Repetir os dois últimos passos até que o centróide não esteja se movendo significativamente.

V. Experimentação e Discussão

A. *Baseline*

Utilizamos a rede neural citada anteriormente como nosso resultado base para este experimento. Nosso conjunto de treino possuía 50 mil imagens e o de validação 10 mil imagens, rodamos a rede por 10 épocas, a *Loss* pode ser conferida na Figura 2 e a acurácia foi 0.9074 no conjunto de treino e 0.8816 no conjunto de validação.

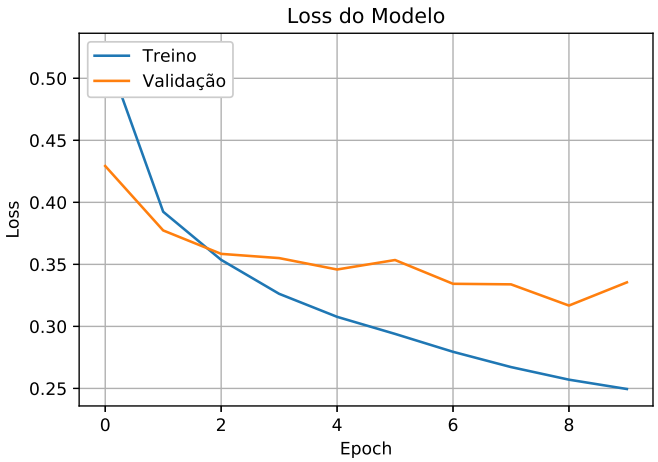

Figura 2. *Loss* e cada época iterada do baseline.

B. *PCA*

Primeiramente analisamos a variância de acordo com o número de componentes rodando o algoritmo, que pode ser conferida na Figura 3.

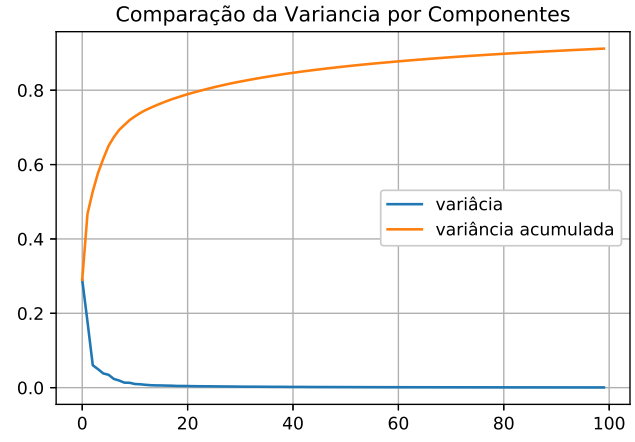

Figura 3. Variância de acordo com a quantidade de componentes para 100 componentes.

Sendo assim, aparentemente se utilizarmos entre 20 e 40 componentes, teremos um bom resultado mesmo com poucas componentes. Então realizamos o experimento com 30 componentes, executamos a mesma rede neural do *Baseline* e a acurácia final foi de 0.8851 no conjunto de treino e 0.4548 no conjunto de validação. Claramente ocorreu *overfitting* no modelo, aliás, ao final da primeira época já havia ocorrido, pois a *Loss* do conjunto de validação já estava muito alta e crescendo enquanto a do conjunto de treino baixa. Fizemos o mesmo teste para números menores de componentes e descobrimos que até com 3 componentes, um número muito baixo, ocorre *overfitting*.

Portanto repetimos o experimento com as mesmas configurações do *Baseline* utilizando apenas 2 componentes e obtivemos acurácia de 0.5513 no conjunto de treino e 0.5504 no conjunto de validação. A *Loss* pode ser conferida na Figura 4 e a acurácia na Figura 5.

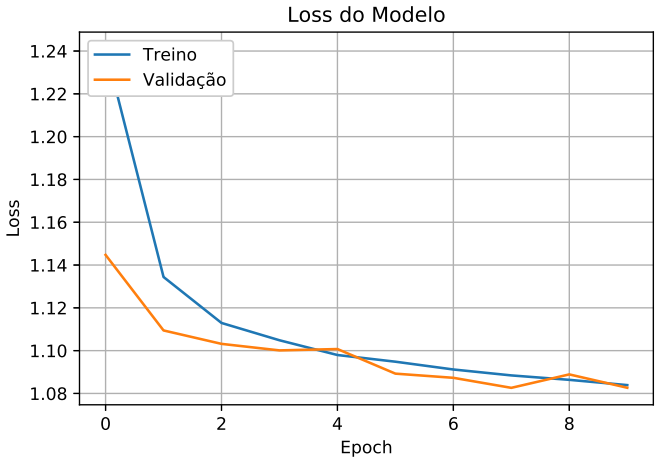

Figura 4. *Loss* ao longo das épocas utilizando PCA com 2 componentes.

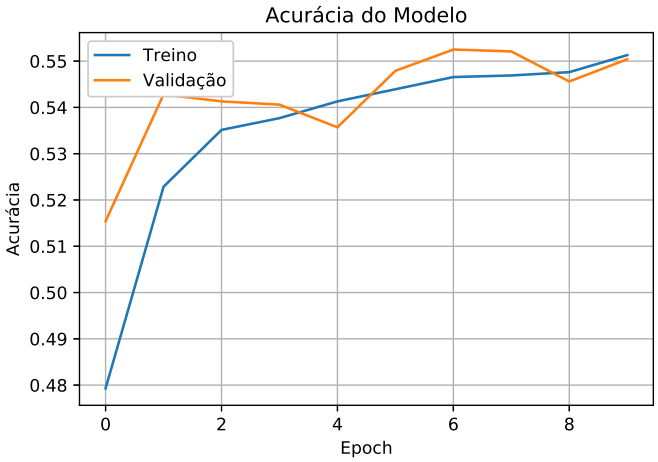

Figura 5. Acurácia ao longo das épocas utilizando PCA com 2 componentes.

C. *Auto-Encoder*

As técnicas não supervisionadas de auto-encoder consistem em aprender uma representação da entrada contendo um número reduzido de features e ser capaz de reconstruir a entrada da versão reduzida. No nosso trabalho aplicamos 2 varições de auto-encoder, uma com um vetor de tamanho 2 e outro de tamanho 20. Inicialmente quisemos gerar no máximo 2 features para poder ter uma melhor visualização dos dados. Em seguida, quisemos verificar a eficacia da rede ao multiplicarmos em 10 vezes o tamanho do vetor. Uma motivação de preparar uma rede com auto-encoder é

poder alimentar o baseline com o resultado reduzido e analisar o que foi aprendido. Ao alimentar o baseline com o resultado do auto-encoder de vetor de tamanho 2, obtivemos uma acurácia de 48%. Já para o vetor de tamanho 20, como já era esperado, foi obtido uma acurácia de 80%. Podemos observar a curva de aprendizado do processo usando o vetor de tamanho 2 e do processo usando o vetor de tamanho 20 nas Figuras 6 e 7, respectivamente. Além disso, podemos observar a evolução da acurácia dos dois processos nas Figuras 8 e 9.

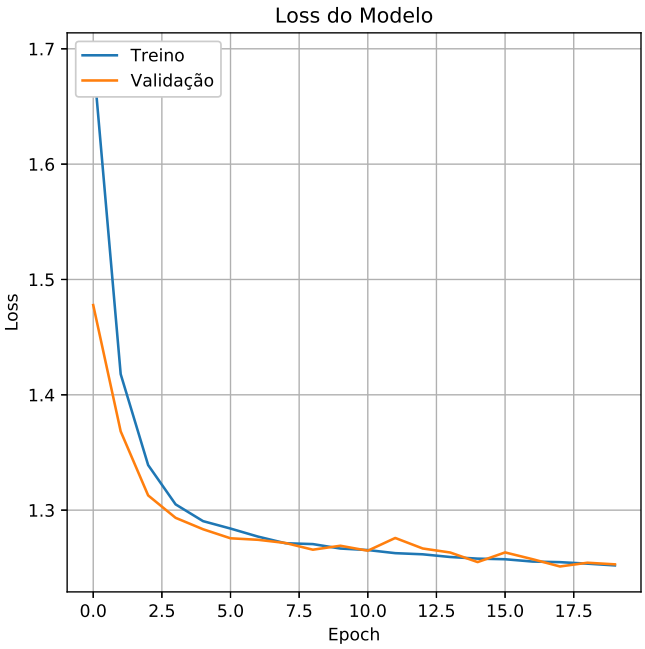

Figura 6. Loss por época ao usar o auto-encoder para gerar um conjunto reduzido de features em um vetor de tamanho 20.

D. K-Means

Como o conjunto original possui 10 classes, executamos o algoritmo com 10 clusters, utilizamos, primeiramente, o conjunto de dados sem aplicar *PCA* ou *Autoencoder*. Com isso, obtivemos no melhor caso uma acurácia de 22%, entretanto, a pior foi de 3%, uma variação grande e um resultado bem ruim. A matriz de confusão do melhor caso pode ser conferida na Figura 10.

Conhecendo o funcionamento do algoritmo, sabemos que ele calcula a distância dos pontos aos centroides. Sendo assim, pode ser que este algoritmo não seja o ideal para classificar peças de roupas, intuitivamente podemos pensar que ele irá conseguir diferenciar bem calçados de camisetas, vestidos e coisas do tipo, pois os pontos estão localizados em partes diferentes da imagem, entretanto, a diferença de distância dos pontos em algumas classes específicas (*coat* e *pullover* por exemplo) pode ser muito pequena para que se identifique uma diferença que possibilite a classificação.

E. Mini-Batch KMeans

Outra forma de aplicar o KMeans é utilizando o formato de mini-batch onde limitamos o número de entradas por iteração a fim de ganharmos mais tempo e velocidade para processar o dataset. Contudo, com a possível perda de acurácia. Para tanto, executamos o processo de clusterização utilizando um batch de tamanho 100. E como esperávamos, obtivemos um resultado de 9%, no melhor

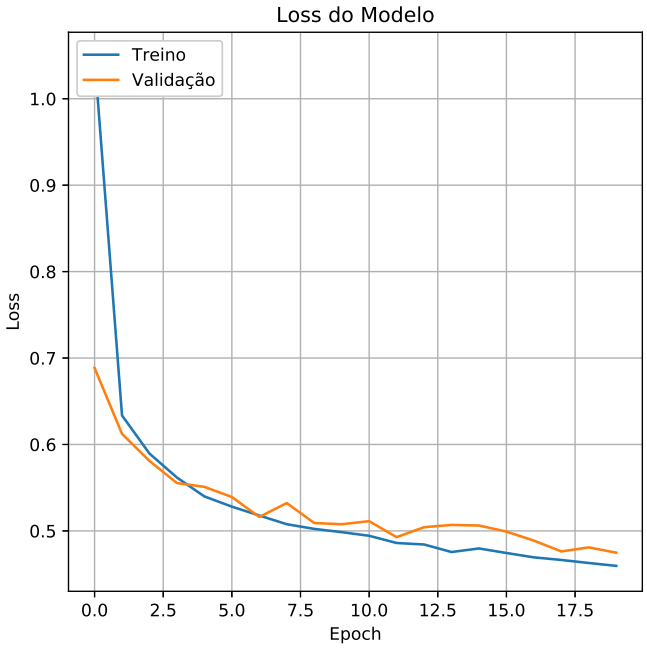

Figura 7. Loss por época ao usar o auto-encoder para gerar um conjunto reduzido de features em um vetor de tamanho 20.

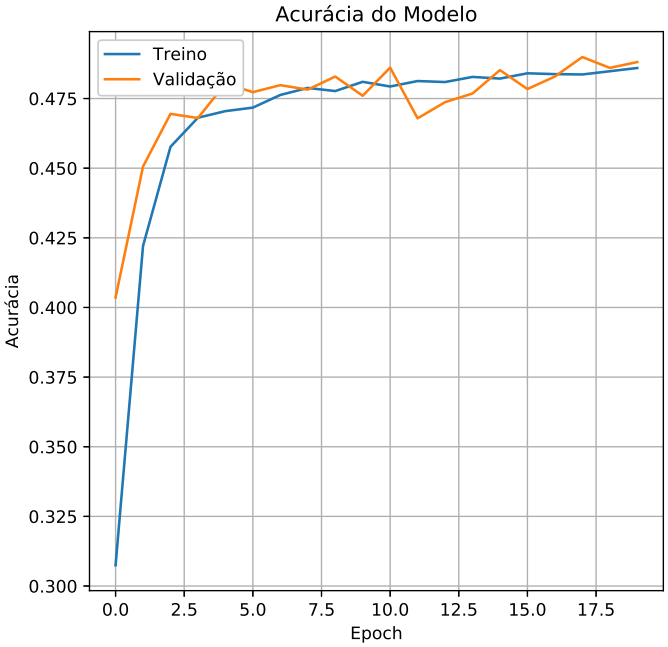

Figura 8. Acurácia por época ao usar o auto-encoder para gerar um conjunto reduzido de features em um vetor de tamanho 2.

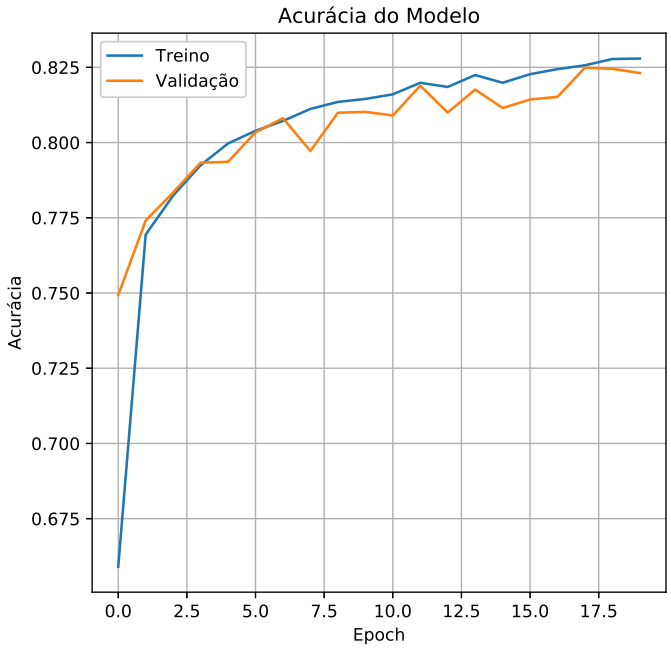

Figura 9. Acurácia por época ao usar o auto-encoder para gerar um conjunto reduzido de features em um vetor de tamanho 20.

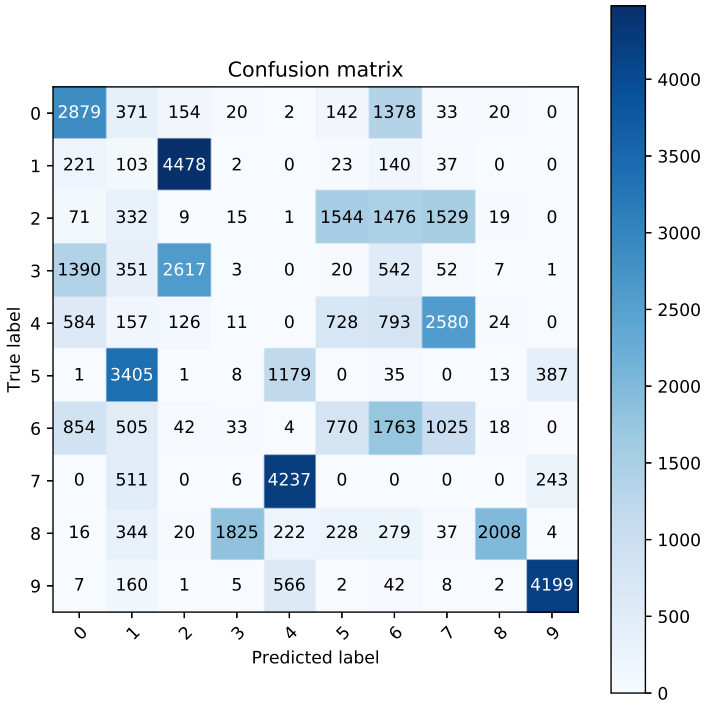

Figura 10. Matriz de confusão ao rodar K-Means com 10 componentes.

caso. Podemos observar da matriz de confusão na Figura 11, que as classificações tiveram um aumento muito significativo na precisão, principalmente ao comparar com a matriz anterior (Figura 10).

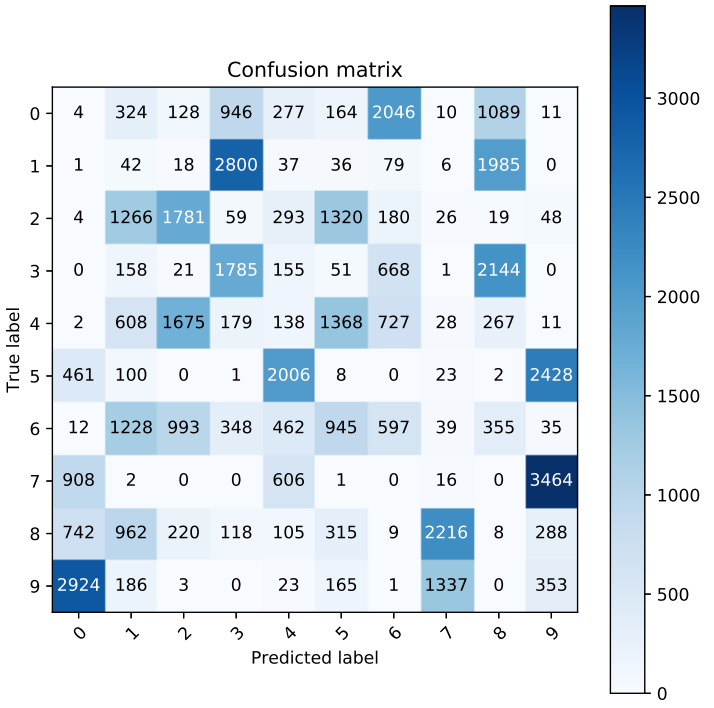

Figura 11. Matriz de confusão ao rodar Mini Batch K-Means com 10 componentes e batches de tamanho 100.

VI. Desempenho no Conjunto de Teste

De todos os nossos modelos o de melhor acurácia foi a rede neural definida no Baseline. Logo, executamos o conjunto de teste nela e obtivemos um acurácia de 88.5%.

VII. Conclusão e Trabalhos Futuros

Nesse trabalho, aplicamos as técnicas de aprendizado não supervisionado. Concluímos que por mais que a melhoria no desempenho impacta negativamente a acurácia do modelo em geral. Porém, não significa que de todo modo as melhorias sejam ruins. Delas obtemos interpretações ao reduzir o número de feature para 2 e treinamos a mesma rede em um menor tempo. Por fim, podemos aplicar as técnicas aprendidas nesse trabalho para estender o mesmo, sendo possível explorar novos tamanhos de vetores e algorítimos de clusterização afim de melhorar tanto desempenho quanto resultado.

Referências

[1] Zalando, "Fashion-mnist." https://www.dropbox.com/s/qawunrav8ri0sp4/fashion-mnist-dataset.zip.

[2] S. Avila, "Classes." http://www.ic.unicamp.br/~sandra/teaching/2019-2-mc886/.

[3] A. Bilogur, "Dimensionality reduction and pca for fashion mnist." https://www.kaggle.com/residentmario/dimensionality-reduction-and-pca-for-fashion-mnist.

[4] F. Chollet, "Building autoencoders in keras." https://blog.keras.io/building-autoencoders-in-keras.html.

[5] P. SHARMA, "The most comprehensive guide to k-means clustering you'll ever need." https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/.

[6] Tensorflow, "Treine sua primeira rede neural: classificação básica." https://www.tensorflow.org/tutorials/keras/classification?hl=pt-br.