Projeto 3 - MC886

Victor Ferreira Ferrari
RA 187890
vferrari@mpc.com.br

Vinícius Couto Espindola
RA 188115
vinicius.c.e@hotmail.com

I. Introdução

O trabalho tem como objetivo principal a exploração de técnicas de redução de dimensionalidade e aprendizado não-supervisionado (*clustering*), desta vez com o auxílio de bibliotecas especializadas que permitem a codificação em alto nível e, consequentemente, a implementação de mais métodos para comparação.

Redução de dimensionalidade é importante para melhorar o desempenho dos classificadores (tempo de execução), e, em diversos casos, pode melhorar o resultado fornecido (casos com número elevado de *features*). Métodos de *clustering* são cada vez mais prioridade em modelos práticos, pois nem sempre se tem muitos dados anotados, então é necessário um método **não-supervisionado** para resolver o problema.

II. Condições e Conjunto de Dados

Na implementação do modelo, foram utilizadas diversas bibliotecas externas da linguagem *Python*, sendo essas `NumPy`, para operações vetorizadas, `Pandas` para leitura de arquivo CSV, `MatPlotLib`, para criação de gráficos para visualização de resultados ou comparações, `Keras` para montagem de redes neurais em alto nível, e `Scikit-Learn` para funções alto-nível para PCA, algoritmos de *clustering* e algumas métricas de avaliação. Além disso, bibliotecas padrão da linguagem também foram utilizadas.

O conjunto de dados explorado é o Fashion-MNIST, famoso *dataset* que consiste em imagens monocromáticas 28x28 separadas em 10 classes de roupas. O conjunto de treinamento consiste de 60000 imagens, e o de teste consiste de outras 10000. O conjunto de validação está incluído no de treinamento.

III. Implementação

Os conjuntos de treino e teste podem ser lidos arquivos CSV facilmente com o auxílio da biblioteca `Pandas`, e em seguida transformados em vetores `NumPy`. Para a *feature scaling* dos dados foi utilizada a padronização (*standardization*). Nota-se que as estatísticas de normalização calculadas no conjunto de treino se mantêm a todos os outros, para preservar a normalização. As classes corretas de cada exemplo são recebidas em *label encoding*, e são transformadas em *One-Hot Encoding* via biblioteca `Keras`.

Procurando garantir que o modelo final é o melhor encontrado, o conjunto de coeficientes que fornece o menor custo de validação é o retornado ao final do treinamento. Para essa verificação e visualização do andamento, os valores de custo para conjuntos de treino e validação são calculados e salvos a cada época.

A. Exploração de Redes Neurais: Baseline

Como *baseline* do projeto, foram implementadas três diferentes arquiteturas de redes neurais densas. Para facilitar a elaboração das redes, criou-se um *wrapper* que define um arquitetura de rede densa a partir de um vetor de inteiros e da quantia inicial de features presentes nos dados.

Também utilizou-se um *wrapper* para a função de treino das redes, e neste implementou-se o método de *early stopping* através de *callbacks*, garantindo que o melhor resultado do treinamento fosse preservado e que não fosse gasto tempo treinando o modelo após este ter convergido. Para reduzir *overfitting*, os modelos foram treinados com o método de reguralização L2.

Arquiteturas de redes densas testadas:

1) 784, 300, 10
2) 784, 470, 100, 10
3) 784, 470, 284, 169, 100, 60, 30, 10

As redes foram treinadas com a função de ativação *ReLU* para as *hidden layers*, enquanto a *output layer* é obtida pela função de ativação *softmax*. A função de custo utilizada é a *categorical crossentropy*.

A seleção dos hiperparâmetros foi feita de maneira empírica, sendo que o otimizador utilizado foi o *Adam* e o *batch size* foi estabelecido em 1024 para todas as arquiteturas.

B. Redução de Dimensionalidade

A segunda etapa do projeto consiste na implementação e análise de dois métodos para redução de dimensionaldade: *Principal Component Analysis* (PCA) e *Autoencoders* (AE). A implementação do PCA utilizada é a presente na biblioteca `Scikit-Learn`, enquanto os Autoencoders foram feitos utilizando a biblioteca `Keras`.

O PCA é um método baseado em álgebra linear no qual se identifica o hiperplano mais próximo dos dados, e os projeta nele. Isso é feito por meio do cálculo dos k primeiros **autovetores**, sendo k o número final de *features* desejado. É possível, porém, em vez de saber *a priori* o número de componentes que o resultado deve ter, utilizar a **variância** como parâmetro. Quanto maior a conservação de variância do conjunto final, mais representativo ele é do conjunto completo de dados. Por isso, o PCA pode ser visto como um método que retira as *features* menos representativas do conjunto final.

Autoencoders (AE) são redes neurais que tentam "comprimir" as informações das variáveis de entrada em um espaço dimensional reduzido, e depois recriar o conjunto de entrada. O componente mais importante é a camada "**bottleneck**" (ou "gargalo"), que é onde a entrada está comprimida, antes do processo de recriação do conjunto original. A partir dessa camada, é possível extrair os dados reduzidos, da mesma maneira como os componentes principais do PCA. O principal tipo de rede testado foi a rede densa, mas também foi feito um AE **convolucional**.

Como parâmetro de redução, no PCA utilizou-se a conservação de variância, sendo que PCAXX% representa a redução por PCA que preserva XX% de variância. Para os AE, o parâmetro é o número de nós da camada de *bottleneck*, sendo que AE[X] representa a redução por AE que reduz as *features* da entrada utilizando uma arquitetura [X].

Considerando as definições, foram testados os seguintes parâmetros de redução:

Param	PCA	AE
1	95%	255
2	90%	128,64,32
3	85%	255,136,20

Foi testada também uma arquitetura convolucional, com até a camada de gargalo.

Input $\rightarrow$ Convolução com 16 filtros 3x3 $\rightarrow$ Max Pooling com janela 2x2 $\rightarrow$ Convolução com 8 filtros 3x3 $\rightarrow$ Max Pooling com janela 2x2 $\rightarrow$ Flatten

Depois, o oposto para reconstruir a imagem original. Observe que, para essa arquitetura funcionar, a imagem deve estar em formato tridimensional (altura, largura e profundidade).

C. Clusterização

A última parte do experimento foi focada em métodos de **clusterização**, técnica não-supervisionada que visa agrupar pontos no conjunto de dados a partir de similaridades entre os dados.

Dois métodos de *clustering* foram testados: **K-Means** e **Agglomerative Clustering**. O *K-Means* foi escolhido dado sua simplicidade e eficiência, tanto de memória quanto de tempo de execução. O *Agglomerative* foi selecionado por utilizar uma tática diferente no agrupamento, denominada *clustering* hierárquico. A implementação utilizada de ambas é a presente na biblioteca `Scikit-Learn`, que fornece classes prontas para a execução dos métodos.

O *K-Means* é um método baseado em distância, no qual os *clusters* são definidos pelo centroide mais próximo. Então, o centroide é deslocado ao centro do *cluster*, e o processo é repetido por um número de iterações. É um método muito utilizado e eficiente, se comparado a outros.

O *Agglomerative Clustering* constrói uma árvore (hierarquia) de forma que existam inicialmente apenas folhas e, conforme o algoritimo é executado, as subárvores sejam conectadas de acordo com a sua similaridade, parando apenas quando o número de árvores é o mesmo que o de classes. Nota-se que *clusterings* hierárquicos consomem muita memória e costumam demorar mais do que o **K-Means**.

Como, neste caso, os dados estão anotados, podemos facilmente associar os *clusters* à classes: cada um dos dez *clusters* são mapeados para exatamente uma classe, de forma que o *cluster* com maior número de acertos em alguma classe tem prioridade no mapeamento. Assim, são geradas as predições de classe para cada exemplo, utilizadas na **visualização** e nas **métricas de avaliação**.

Para avaliação dos resultados, foram utilizadas métricas presentes na biblioteca `Scikit-Learn`, supervisionadas e não-supervisionadas [1].

Como métodos de avaliação **supervisionados**, temos o *Adjusted Rand Index*, que mede a similaridade entre dois vetores (no caso, os reais e os agrupados. Uma métrica similar, mas não igual, é a *Adjusted Mutual Information*, que mede a concordância entre os *clusters* e os rótulos reais.

As últimas duas métricas são mais intuitivas, sendo a *V-Measure*, média harmônica entre a **homogeneidade** e a **completude** dos *clusters* em relação aos rótulos reais, e a *Acurácia*, obtida pela matriz de confusão, trivialmente construída a partir das previsões feitas anteriormente.

Para comparação, foi utilizada também uma famosa métrica de avaliação não-supervisionada, que utiliza a distância entre pontos do mesmo *cluster* e a distância entre pontos de *clusters* diferentes para estimar a qualidade dos mesmos sem conhecer os rótulos reais. É chamado de **Silhouette Coefficient**.

IV. Experimentos and Discussões

A. Redes Neurais

Para a primeira parte do experimento, testou-se as aquiteturas discutidas na implementação. Os resultados obtidos dada as condições descritas (*batch size*: 1024, *optmizer*: Adam e regularização L2) se encontram na tabela abaixo.

Arq.	Acurácia Treino	Acurácia Validação
1	89,52%	88,13%
2	90,19%	88,38%
3	91,43%	88,47%

Percebe-se que, apesar das variações de complexidade das arquiteturas, os resultados são próximos, o que indica certas limitações da rede neural densa. Essas pequenas mudanças são naturais da aleatoriedade das escolhas no treinamento.

Visando escolher a rede mais eficiente em relação a acurácia e complexidade, selecionamos a rede 2, já que esta apresenta uma complexidade intermediária e pode ser treinada mais rapidamente que a 3. A rede 1 poderia ser selecionada também, mas a escolha final foi a 2.

Para garantir que a arquitetura selecionada não apresentou *overfitting*, regulou-se o parâmetro de regularização L2 e a curva de aprendizado foi feita, vista na figura 1.

B. Reduções de Dimensionalidade

Com a arquitetura 2 selecionada, aplicamos as seis reduções de dimensionalidades mencionadas na seção III. Os resultados

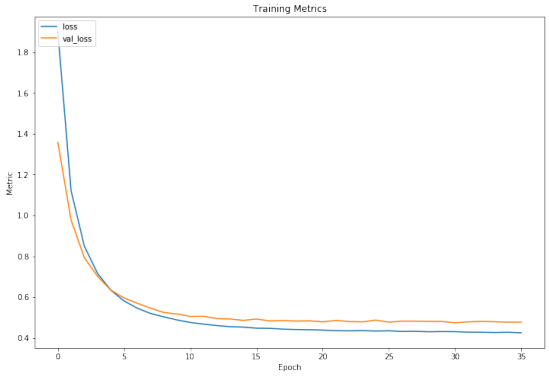

Figure 1. Curvas de aprendizado da arquitetura 2

da redução das **784 dimensões originais** encontram-se nas tabelas abaixo.

PCA	Dims	Tempo (s)	Treino	Validação
PCA95%	256	17.0	90,29%	88,80%
PCA90%	137	15.9	89,88%	88,50%
PCA85%	81	17.5	88,87%	88,06%

AE[X]	Dims	Tempo (s)	Treino	Validação
255	255	330.6	91,24%	88,83%
128,64,32	32	246.2	86,84%	86,23%
255,136,20	20	413.2	87,54%	86,43%

Nota-se que os treinos com PCA95% e o AE[255] apresentaram acurácias um pouco melhores do que com o *dataset* completo. Isso pode ocorrer pois as *features* reduzidas pelo PCA são pouco relevantes para o *dataset* e a redução realizada pelo AE é feita de maneira ponderada (treinada), neste caso, os maiores pesos são atribuídos para as features mais representativas. A eliminação de dados pouco representativos do conjunto, podem ajudar o treinamendo do modelo, uma vez que as *features* irrelevantes são ignoradas e o foco é voltado às *features* importantes do *dataset*. Ainda assim, a melhora foi marginal, e pode ser devido a aleatoriedades no treinamento.

Em relação a redução das dimensões, o PCA inicialmente aparenta ser mais eficiente, uma vez que consegue reduzir uma quantia significativa de features mantendo uma boa porcentagem de variabilidade. Todavia, o AE[255] reduz tanto quanto o PCA95% e produz resultados um pouco melhores. Esta observação pode estar vinculada ao fato que os AE, diferentemente do PCA que simplesmente descarta dados, tenta comprimir as informações originais em um número menor de dimensões evitando ao máximo perda de informação, logo, em reduções onde o PCA perde muita variabilidade para pruduzir uma redução significativa, os AE podem ser ferramentas mais adequadas.

Quanto ao tempo de execução, enquanto o PCA necessita poucos segundos para realizar a redução dos três parâmetros descritos, em contrapartida, os *autoencoders* são significativamente mais lentos para completar a redução, uma vez que estes têm que ser treinados. Pelo seu bom resultado e baixo tempo de execução, o PCA95% foi a redução escolhida para dar continuidade ao experimento.

O AE convolucional utilizado retornou o pior resultado dentre todos os utilizados, com acurácia de 87,75% com 392 *features*, e um tempo de execução altíssimo. Por isso, não foram testadas outras arquiteturas ou muito priorizados.

Assim, foi testado o conjunto de **teste** do *dataset* com a rede final: arquitetura 2 com PCA95%. Treinando pela última vez o modelo, temos acurácia de validação de 89,02%. Foi aplicado o mesmo modelo PCA treinado anteriormente no conjunto de teste. Utilizando essa última rede para prever o conjunto de teste, criamos a matriz de confusão e obtemos acurácia de 89,35%, ou seja, o modelo generalizou bem.

C. Clusterização

A fim de visualizar o agrupamento do conjunto de dados, foi feito um *scatter plot* utilizando a redução por PCA e selecionando apenas as duas componentes principais (que preservam mais variabilidade). Observando a **Figura 2** temos que, apesar da visualização ter uma boa representatividade (36,45%), há muitos conjuntos com agrupamentos quase que indistinguíveis. Classes similares, como *sandals* (marrom) e *sneakers* (cinza), por exemplo, estão sobrepostas e dependem da variabilidade remanescente (63,55%) para serem propriamente distinguidas.

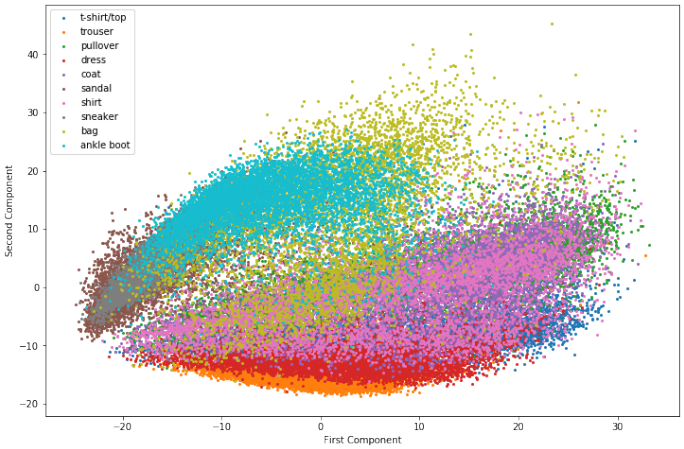

Figure 2. Classes ilustradas pelas duas features mais representativas (36,45% de variabilidade).

Treinando os os métodos de agrupamento e associando os *clusters* resultantes as classes do *dataset*, geramos as matrizes de confusão para ambos os métodos testados.

Analisando as matrizes de confusão (**Figura 3** e **Figura 4**), percebe-se algumas limitações dos métodos sobre o *dataset*. Como previsto pelo *scatter plot* da **Figura 2**, certas classes não foram bem distinguidas.

No caso do **K-Means**, a classe *sandal* teve quase metade de suas amostras agrupadas sobre o cluster *sneaker*, implicando que a classe *sandal* possui cerca de 34% de chance ser rotulada como *sneaker*. Todavia, o caso mais discrepante são as classes *pullover* e *dress*, que tiveram quase que 100% de erro em suas predições: a primeira é agrupada pelos clusters *coat* e *sneaker* a segunda se encontra nos clusters *t-shirt* e *trouser*.

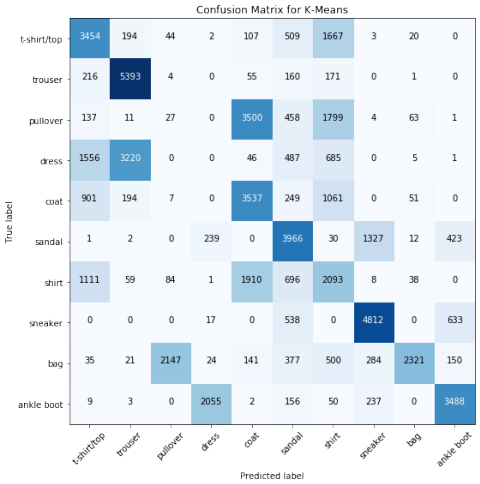

Figure 3. Matriz de Confusão do K-Means (Acc. 48,49%).

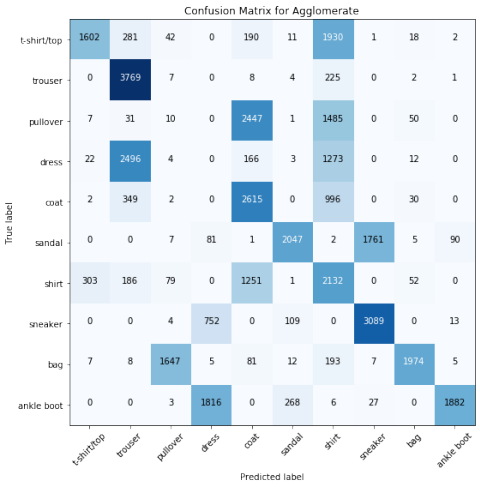

Figure 4. Matriz de Confusão do Agglomerative Clustering (Acc. 47,80%).

Nota-se que há muita intersecção espacial entre as classes, dificultando o uso de métodos não-supervisionados. Ambos os métodos de clustering ficaram próximos de 50% de acurácia, o que é muito inferior ao resultado obtido pelo método supervisionado aplicado.

A partir da transformação dos *clusters* em previsões, podemos utilizar outras métricas de avaliação descritas na seção III. Os resultados estão a seguir, com ARS: Adjusted Rand Score, AMS: Adjusted Mutual Score, VS: V-Measure Score, Acc: Acurácia e SS: Silhouette Score. O modelo 1 é o *K-Means* e o modelo 2 é o *Agglomerate Clustering*.

Mod	ARS	AMS	VS	Acc	SS
1	0,3498	0,4945	0,5067	0,4849	0,1374
2	0,3612	0,5544	0,5756	0,4780	0,1223

Vemos que os valores de avaliação são parecidos entre os métodos, e em geral positivos. As 2 primeiras métricas têm limites entre -1 e 1, então os valores obtidos são bons. Isso confirma que os *clusters* fazem sentido, e que o método está agrupando baseado em similaridade com sucesso. Ainda assim, como visto pela acurácia, o resultado está longe de ser similar aos métodos supervisionados.

A *Silhouette Score*, mesmo não-supervisionada, é o que mais se aproxima à acurácia, com o valor próximo a 0, com limites entre -1 e 1. Isso mostra que é uma boa métrica para se usar quando não há dados anotados.

V. Conclusões e Próximos Passos

A primeira conclusão desse projeto é que redução de dimensionalidade é um ótimo método para diminuir a quantidade de *features* de um problema, principalmente para redução do tempo de execução e simplificação, sem alterar a qualidade do resultado.

Desses métodos, alguns são melhores do que outros. Para o problema e o *dataset* em questão, PCA teve resultados muito parecidos que *autoencoders*, mas o método é mais rápido de ser executado, então foi o escolhido como melhor entre os testados. Percebe-se que houve uma grande redução no número de componentes mantendo 95% da variância do conjunto original.

Entre os *autoencoders*, redes densas tiveram resultados melhores ou iguais à convolucional, mas com bem menor tempo de treinamento. Assim, conclui-se que para o problema um AE convolucional não é necessário.

O modelo final do problema atingiu acurácia de 89% no conjunto de teste, resultado que pela facilidade notável do *dataset* (como visto em [2]) poderia ser melhor, mas não é um resultado ruim, para a simples exploração feita, e mostra que o modelo generalizou bem em relação ao conjunto de treinamento/validação, e *overfitting* foi evitado.

Em um próximo trabalho, poderiam ser testados outros tipos de redes mais profundas, como redes convolucionais clássicas ou construídas. Maior pré-processamento poderia ajudar também.

Por último, analisando as técnicas não-supervisionadas de *clustering*, percebe-se que ambas técnicas retornaram resultados similares, o que fica evidente ao se observar as matrizes de confusão, nas quais as mesmas duas classes ficaram com nenhum ou poucos acertos.

Ao todo, os métodos não-supervisionados retornaram soluções de qualidade muito inferior aos supervisionados, porém muitos dos *clusters* fazem um certo sentido, então as implementações estão funcionando para a finalidade que eles foram feitos. Isso demonstra a diferença que anotações em dados têm no resultado final, e o problema que se gera ao ter que agrupar dados sem o auxílio de anotações.

Próximos passos para tentar resolver o problema por *clustering* seriam testar tipos diferentes de métodos, como os baseados em densidade (DBSCAN, por exemplo), e outros tipos de pré-processamento dos dados para auxiliar no agrupamento.

References

[1] Scikit-Learn, "2.3.10: Clustering performance evaluation". Disponível em: https://scikit-learn.org/stable/modules/clustering.html#clustering-evaluation

[2] Zalando Research, "Fashion-MNIST". Disponível em: https://github.com/zalandoresearch/fashion-mnist