Trabalho Final - Introdução ao Aprendizado de Máquina Classificadores de Pôsteres

Arthur Pinheiro Nascimento[1]

[1]Instituto de Computação – Universidade Federal do Rio de Janeiro (UFRJ)
Pedro Calmon, 550 – Edifício Jorge Machado Moreira – Térreo
Cidade Universitária – Rio de Janeiro – RJ – CEP: 21941-630

[2].

arthurpn@ic.ufrj.br

Abstract. *This paper was made with the intention to answer one question: is it possible to infer the genre of a movie solely by it's poster? Two classifying methods were used, one neural network and one random forest. In it's totality, this project intends to explain what was done to clean the data, the challenges faced and the conclusion found by the author.*

Resumo. *Este artigo foi feito com a intenção de responder uma pergunta: é possível inferir o gênero de um filme apenas pelo seu pôster? Foram utilizados 2 classificadores, uma rede neural e uma floresta aleatória . No total o trabalho se propõem a explicar como foi feito o tratamento dos dados, os desafios encontrados e a conclusão encontrada pelo autor*

1. Introdução

Pessoalmente, pretendo seguir por uma linha de tratamento de imagens e visão computacional, então procurei especificamente por um dataset que pudesse me fazer ganhar mais experiência na área. Foi quando encontrei o dataset utilizado aqui, na própria descrição dele no kaggle estava explicitado a pergunta a ser respondida pelo artigo: é possível inferir o gênero de um filme apenas pelo seu pôster?

A ideia inicial é que filmes do mesmo gênero tivessem aspectos similares, por exemplo, filmes de romance teriam um casal ou filmes de crime teriam uma arma e um tom mais escuro. Claro, ainda é muito cedo para tirar conclusões, mas a expectativa era que a resposta para a pergunta seria sim. Então vamos modelar o trabalho em volta de um teste de hipótese, sobre a acurácia dos nossos modelos.

$$\begin{cases} H_o : acc \geq 70 \\ H_1 : acc < 70 \end{cases}$$

Se pelo menos um dos modelos chegar à 70% de acurácia, aceitamos a hipotése nula e consideramos que é possível inferir o gênero de um filme pelo seu pôster. Caso o contrário, consideramos que não há evidências o suficiente para a aceitar.

Sobre o dataset em si, ele é composto por uma pasta com imagens chamado "SampleMoviePosters"e um arquivo chamado "MovieGenre.csv". O arquivo csv está estruturado em 6 colunas, com um exemplo a seguir:

	imdbId	Imdb Link	Title	IMDB Score	Genre	Poster
0	114709	http://www.imdb.com/title/tt114709	Toy Story (1995)	8.3	Animation|Adventure|Comedy	https://images-na.ssl-images-amazon.com/images...
1	113497	http://www.imdb.com/title/tt113497	Jumanji (1995)	6.9	Action|Adventure|Family	https://images-na.ssl-images-amazon.com/images...
2	113228	http://www.imdb.com/title/tt113228	Grumpier Old Men (1995)	6.6	Comedy|Romance	https://images-na.ssl-images-amazon.com/images...
3	114885	http://www.imdb.com/title/tt114885	Waiting to Exhale (1995)	5.7	Comedy|Drama|Romance	https://images-na.ssl-images-amazon.com/images...
4	113041	http://www.imdb.com/title/tt113041	Father of the Bride Part II (1995)	5.9	Comedy|Family|Romance	https://images-na.ssl-images-amazon.com/images...

Figura 1. MovieGenre.csv

A partir das colunas observadas, de longe, a mais importante para o trabalho é a coluna ”Genre”, mas as colunas ”imdbId”e ”Poster”também serão importantes mais a frente, uma para identificar as imagens e outra para baixar novas.

Já a pasta com as imagens, contém pouco menos de 1000 imagens em formato .jpg de diversos filmes, cada pôster está nomeado com o seu respectivo ”imdbID”, por exemplo, o pôster do filme de Toy Story teria o nome de ”114709.jpg”.

Para responder a pergunta proposta anteriormente, foram utilizados 2 métodos de classificação, um deles foi a rede neural. O motivo da escolha dela foi simples, era esperado que ao utilizar camadas convolucionais, a rede aprenderia padrões que aparecem recorrentemente nos pôsteres de cada gênero e seria um bom classificador. Outro motivo foi que nas pesquisas feitas para a escolha dos modelos, a rede neural convolucional sempre aparecia como recomendação.

O segundo método utilizado será a floresta aleatória. A escolha deste modelo se deu apenas pois o professor, em aula, comentou que este é um dos melhores algoritmos de classificação que existem. Dessa forma, buscando a melhor acurácia na classificação, este foi o segundo escolhido.

Para concluir essa seção queria dar algumas ressalvas. O meu computador, do autor do trabalho, é velho, não possui placa de vídeo e apenas 4GB de RAM, por isso o trabalho foi feito inteiramente pelo colab que disponibilza GPU e TPU gratuitamente. No entanto, isso também significa que o andamento do projeto foi limitado ao poder de processamento do ambiente de execução. Por conta disso, não foi utilizado K-Fold para separar conjunto de treino, pois ele derruba o colab quando tento passar os dados de treino para a rede neural. Dito isso, podemos seguir para o tratamento de dados.

2. Tratamento de dados

Para me guiar, inicialmente tentei buscar inspirações em notebooks disponíveis no próprio kaggle para dar um direcionamento no projeto. Entretando, os esforços foram em vão, apenas 3 notebooks tentaram classificar os filmes por gênero e nenhum tratou os dados de forma que considerei adequada, então, segui por conta própria.

Para começar, quis checar a proporção de gêneros das imagens disponíveis. Para isso explodi o dataset filtrado apenas com as imagens já baixadas e descobri que ele estava muito enviesado para Drama (cerca de 27%), Comédia (17%) e Romance (14%). Além, é claro, de haver poucas imagens disponibilizadas. A partir daqui decidi que iria utilizar a biblioteca **requests** para baixar novas imagens em busca de deixar o conjunto de dados

mais balanceado e diverso.

Mas antes disso, fiz uma análise para descobrir quais são os gêneros contemplados no arquivo csv. No total, haviam 28 gêneros, acredito que classificar filmes em 28 gêneros poderia confundir os modelos, pois os filmes podem ser classificados em mais de um gênero, como mostrado na Figura 1. Sob esse prisma, decidi reduzir para que a classificação seja feita sobre apenas 6 gêneros, houveram 2 critérios de escolha para esta tarefa. O primeiro era que um gênero qualquer houvesse pelo menos 2000 filmes classificados nele e o segundo foi gosto pessoal dentro dos que passaram da primeira seleção. Com isso os gêneros selecionados foram:

- Drama
- Comedy
- Romance
- Action
- Crime
- Documentary

Com as classes definidas, fiz uma função que baixasse 2000 imagens de cada um dos gêneros selecionados, totalizando 12.000 imagens. Agora falta apenas separar os dados em treino e teste. Como citado anteriormente, ao tentar utilizar K-fold o ambiente de execução é derrubado, então utilizei apenas uma separação normal entre treino/teste, com 70% destinado ao treino, 15% para o teste e 15% para validação. Um adendo, também usei uma função de preprocessamento do keras que aplica algumas transformações nas imagens para a rede neural, será explicado mais a frente.

Para o segundo classificador, as imagens foram pre-processadas, utilizando a função ”hog”da biblitoeca **skimage.features**. Esta encontra features na sua imagem e retorna outra imagem apenas com as as características encontradas. Estas podem consistir de diversas coisas, como, rostos, mãos, roupa, etc., contornos no geral. Depois disso, elas foram separadas nos conjuntos de treino(80%) e teste(20%) e direcionadas para o treino da floresta aleatória. Não foi separado dados para validação nesse caso, pois a biblioteca não permite passar esse tipo de conjunto para o classificador, então a análise só poderá ser feita após o término do treino.

3. Modelos utilizados

Nesta seção serão discutidos como cada classificador foi construído.

3.1. Rede Neural

Para a implementação da rede neural, a biblioteca tensorflow e a API keras foram utilizados.

3.1.1. Arquitetura da Rede

Como estamos lidando com imagens, desde o início já era claro que iríamos construir uma rede neural convolucional. Este também foi o modelo mais desafiador e que eu mais tive que fazer ajustes para chegar na acurácia final, também é o que mais demora para treinar.

Antes de seguir o guia no site do tensorflow, estava tentando fazer as coisas sozinho com ajuda apenas de alguns poucos sites e o ChatGPT. Por algum motivo a acurácia durante o treino estava horrível, o melhor resultado que cheguei nesse modelo foi uma taxa de acerto de 0,02. Com a ajuda do guia consegui montar a rede final com uma acurácia muito melhor.

O resultado disso significou alterar a camada de entrada da rede para normalizar as imagens. Estas costumam ser guardadas seguindo o padrão RGB, isso significa que existem 3 canais de cor e cada pixel é um vetor com números no intervalo [0, 255]. A explicação dada pelo tensorflow, é que o valor alto que um pixel pode assumir (eg. 249) impacta no treinamento da rede neural e o mais recomendado para uma boa prática é adicionar uma camada de entrada que normalize todos os pixels para estarem dentro do intervalo de valor [0, 1].

A segunda ação tomada foi adicionar mais uma camada convolucional, totalizando 3 camadas convolucionais. Com todas essas novas práticas segui para o treinamento do modelo. Agora a acurácia de validação começou a subir mas depois de um tempo convergia para valores próximos de 0.18. Com este ponto de partida, testei mais algumas arquiteturas mexendo poucas coisas até chegar na rede final, a diferença foi adicionar mais uma camada escondida (densa/totalmente conectada) e testar novamente mexendo na quantidade de neurônios em cada uma delas.

A última recomendação do site foi utilizar uma função do keras de preprocessamento de imagens que aumenta artificialmente seu dataset. Ela aplica diversas transformações na hora de carregar as imagens o que deveria ajudar a rede a reconhecer melhor diferentes orientações e distorções na imagem. Ela também aplica resize (apesar de todas já serem do mesmo tamanho, como constatado neste notebook) e define o tipo de classe escolhida (nesse caso foi 'categorigal')

Por fim, cada camada aplica a função "relu", exceto na saída que foi utilizado "softmax". Esta é uma função utilizada especialmente em classificadores de classe única que consiste em aplicar a função exponencial no vetor de saída e tirar a média dos valores no vetor, dando como resultado a probabilidade de classificação em cada uma das classes. Vamos passar por um exemplo para explicar melhor, esta função é definida como:

$$\sigma(\mathbf{z_i}) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{1}$$

onde:
- $\sigma(\mathbf{z_i})$ é a i-ésima saída do vetor de probabilidades.
- z_i é a i-ésima entrada do vetor de valores reais $\mathbf{z}$.

Considere o vetor $\mathbf{z} = [2, 1, 0.1]$. Vamos calcular as probabilidades usando a função softmax. Primeiro, calculamos o exponencial de cada componente do vetor:

$$e^{z_1} = e^2 \approx 7.389$$
$$e^{z_2} = e^1 \approx 2.718$$
$$e^{z_3} = e^{0.1} \approx 1.105$$

Em seguida, somamos todos esses valores exponenciais:

$$\sum_{j=1}^{3} e^{z_j} \approx 7.389 + 2.718 + 1.105 = 11.212 \tag{2}$$

Finalmente, calculamos as probabilidades:

$$\sigma(\mathbf{z})_1 = \frac{e^{z_1}}{\sum_{j=1}^{3} e^{z_j}} \approx \frac{7.389}{11.212} \approx 0.659$$
$$\sigma(\mathbf{z})_2 = \frac{e^{z_2}}{\sum_{j=1}^{3} e^{z_j}} \approx \frac{2.718}{11.212} \approx 0.242$$
$$\sigma(\mathbf{z})_3 = \frac{e^{z_3}}{\sum_{j=1}^{3} e^{z_j}} \approx \frac{1.105}{11.212} \approx 0.099$$

Portanto, as probabilidades resultantes do vetor $\mathbf{z} = [2, 1, 0.1]$ usando a função softmax são aproximadamente $[0.659, 0.242, 0.099]$.

Esse foi um texto muoto expositivo, então irei resumir brevemente a arquitetura para que possamos olhar melhor para a rede final (treinada em 20 épocas pois é o que ocolab aguenta).

1. Rescaling(1/255) - Camada de entrada - 97200 neurônios
2. Convolução e maxPooling 1 - ativação: ReLu
3. Convolução e maxPooling 2 - ativação: ReLu
4. Convolução e maxPooling 3 - ativação: ReLu
5. Flatten()
6. Densa - ativação: ReLu
7. Dropout(0.4)
8. Densa - ativação: ReLu
9. Densa(6) - ativação: softmax (Camada de saída)

3.1.2. Função de perda e Otimizador

A função de perda escolhida foi a categorical cross entropy, pois é a mais apropriada para classificação multi-classe (o objetivo do trabalho). Deve-se atentar à apenas uma coisa, essa função precisa que os labels (nesse caso os gêneros) estejam one-hot encoded em um vetor, felizmente isso já é tratado na hora da separação do dataset em para treino/validação/teste.

Já o otimizador Adam (Adaptive Moment Estimation) combina as vantagens dos métodos AdaGrad e RMSProp, ajustando as taxas de aprendizado individualmente para cada parâmetro com base em médias móveis dos gradientes (momento da primeira ordem) e dos quadrados dos gradientes (momento da segunda ordem). Isso permite que esse otimizador seja bom para problemas com gradientes com muito ruído ou esparsos, proporcionando uma convergência mais rápida e estável. Mas acima de tudo isso, todo lugar que eu pesquisei para fazer classificação de imagens, esse otimizador era utilizado.

3.2. Floresta Aleatória

Este classificador foi muito mais fácil de se trabalhar, apenas um site me ajudou a montar todo ele, ademais, o tempo de treinamento não chega a ser 1 décimo do tempo da rede neural. O único problema seria que a classificação tem uma acurácia (muito pouco) menor. Ainda assim, considero este modelo mais vantajoso de se usar.

Antes de se treinar o modelo ele, é recomendado tirar as ”hog features”das imagens. Isso significa fazer um preprocessamento na imagem tal que certas características na imagem sejam encontradas, como formatos ou a estrutura de objetos, o que nos ajudará na hora de fazer a classificação das imagens.

Depois disso, bastou passar as imagens pré-processadas para o classificador e em menos de 15 minutos ele me retorna uma acurácia de 0.24, além de diversas outras métricas de teste. Não só isso, mas internamente ele aplica um bootstrap nos dados.

Sobre a estrutura do ensembler, a floresta foi treinada para utilizar 1000 árvores no total, cada uma com altura máxima de 5. O critério para classificação também foi mudado para entropia ao invés do default(gini), assim representando o que foi visto em sala de aula. No entanto, parece que definir a altura máxima da árvore foi mais prejudicial que vantajoso, então reitrei essa restrição e deixei o default, as outras ficaram como descritas.

4. Problemas encontrados

Da forma que o texto foi escrito, explicitei que fiz uma pesquisa para construir meus classificadores, o que não ficou claro é que levei 2 semanas para atingir resultados que considerei aceitáveis para a conclusão do trabalho. Parte das dificuldades encontradas se deu pelo uso do colab que às vezes desconetava sozinho e se fazia necessário começar o treino do 0. Além de ter que descobrir na experiência que quando algo é baixado no ambiente do colab e este é reiniciado, todos os arquivos baixados são perdidos. Então precisei passar todas as 12.000 imagens para o meu drive montá-lo toda vez que voltava para o trabalho.

Outra coisa que ficou explícito no texto foi que não tive problemas com a floresta aleatória, o que é verdade, mas toda essa facilidade foi compensado pela rede neural, o motivo de ter demorado tanto tempo para terminar este trabalho. Como dito anteriormente, no começo tentei fazer sozinho, mesmo já sabendo que provavelmente não iria atingir um resultado satisfatório, mas eu queria tentar. Depois disso tentei seguir um tutorial que encontrei no youtube, mas ele fazia classificação binária e utilizava funções (tanto no tratamento de dados quanto na rede) que eu não entendia, ele também não explicava. Os comentários do vídeo também estavam dizendo que o classificador dele sofreu overfitting, então não me extendi muito mais aqui.

Depois disso, com mais pesquisas, percebi que poderia melhorar a arquitetura da rede neural e também percebi que tinha capacidade para baixar mais imagens e assim o fiz. Aqui, não sabia mais como melhorar o que já estava feito mas a acurácia ainda estava muito ruim (nos 2 classificadores). Foi aí que perguntei ao professor o que poderia melhorar o resultado que tinha atingido e ele me deu o insight de que as minhas imagens de treino poderiam estar classificads em mais de um classe. Fiz uma revisão dos cartazes baixados e de fato, um cartaz por vezes pertencia à mais de uma classe no treino/validação, consertei isso e cheguei no resultado final do trabalho que será dscutido

no capítulo abaixo.

5. Resultados e Conclusão

A seguir seguem as métricas e/ou matrizes de confusão encontradas para cada modelo, os resultados serão dsiscutidos posteriormente.

Acurácia: 0.14777777777777779

	precision	recall	f1-score	support
Documentary	0.14	0.17	0.16	300
Crime	0.11	0.11	0.11	300
Action	0.15	0.10	0.12	300
Romance	0.15	0.17	0.16	300
Comedy	0.19	0.13	0.15	300
Drama	0.16	0.21	0.18	300
accuracy			0.15	1800
macro avg	0.15	0.15	0.15	1800
weighted avg	0.15	0.15	0.15	1800

Figura 2. Métricas: Rede Neural

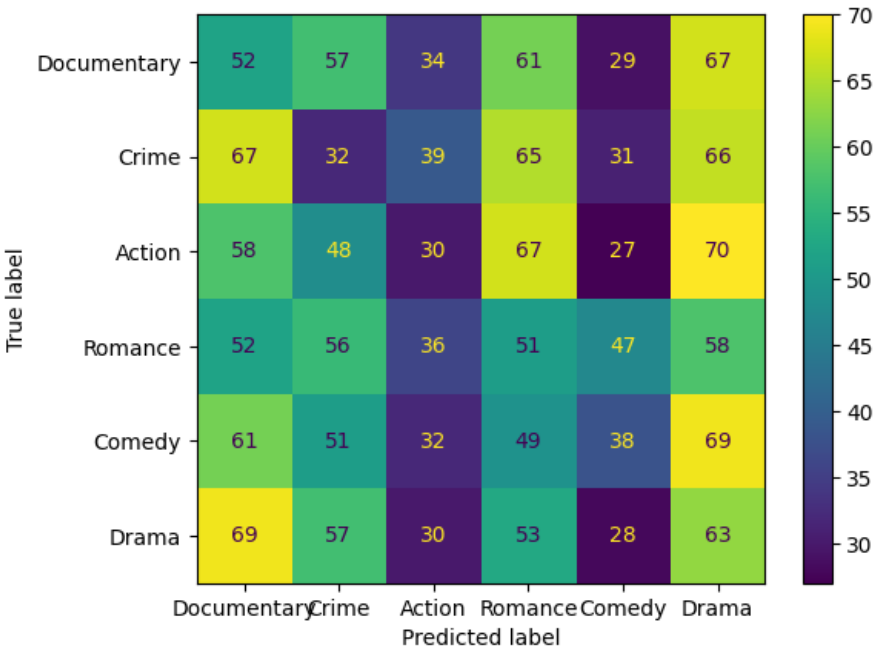

Figura 3. Matriz de Confusão: Rede Neural

Acurácia: 0.24222222222222223

	precision	recall	f1-score
Documentary	0.28	0.31	0.30
Crime	0.26	0.20	0.23
Action	0.26	0.50	0.35
Romance	0.17	0.18	0.17
Comedy	0.21	0.15	0.18
Drama	0.22	0.10	0.14
accuracy			0.24
macro avg	0.24	0.24	0.23
weighted avg	0.24	0.24	0.23

Figura 4. Métricas: Floresta Aleatória

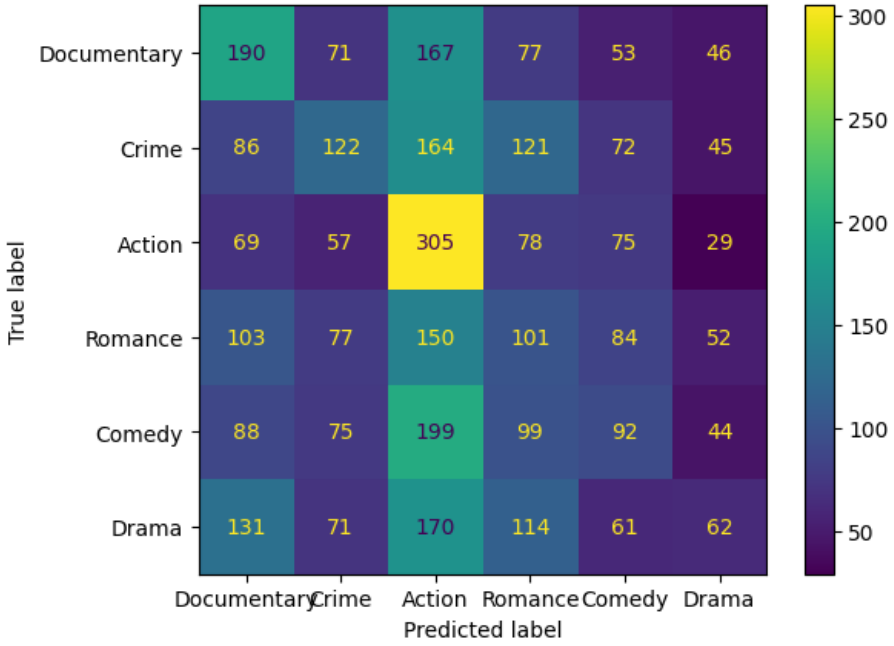

Figura 5. Matriz de Confusão: Floresta Aleatória

Analisando a matriz de confusão da floresta aleatória é bem perceptível que ela está enviesado para chutar os filmes como sendo de Ação, ao mesmo tempo que tem dificuldade de os classificar como Drama. As métricas calculadas mostram bem isso, em que Ação possui um recall de 0.5 (o maior) e Drama de 0.10 (o menor). A acurácia do modelo foi de 24,22%.

Para a rede neural, sua acurácia foi de 14,72% no conjunto de teste, o que me é estranho visto que sua acurácia no treino e validação foram cerca de 30%. Sua matriz de confusão parece demonstrar o mesmo problema da floresta aleatória mas em uma escala menor, há um viés na previsão para Drama. Experimentando com redes diferentes, o resultado apresentado foi o melhor encontrado.

Não é possível ter certeza do porque este foi o resultado encontrado, mas há alguns fatores que podem ter influenciado para chegar á este resultado. O primeiro é que um cartaz não precisa seguir nenhum padrão de acordo com seu gênero, além disso, um pôster

também pode ser pensado como uma forma de arte e ter formatos abstratos ou passar a ideia que o filme quer passar e não necessariamente seu gênero. O segundo motivo possível para isso seria a idade dos filmes. No dataset escolhido haviam filmes feitos em todas as épocas, analisei ele por cima e o filme mais antigo que encontrei foi feito no século XIX **muito** tempo atrás. Com filmes tão antigos, é difícil afirmar que por tanto tempo eles seguiram os mesmos padrões até os dias de hoje, mas nem é necessário voltar tanto assim no tempo, basta analisar um pôster de um filme na década de 50 para um de hoje em dia e a diferença é clara.

Para finalmente concluir o objetivo do trabalho, observando as acurácias encontradas, ambas se encontram abaixo de 70%. Portanto, consideramos não haver evidências o suficiente para aceitar a hipótese nula definida no início do artigo.

6. Referências

1. Como construir um modelo de floresta aleatória: https://www.geeksforgeeks.org/random-forest-for-image-classification-using-ope
2. Como criar uma rede neural convolucional: https://www.tensorflow.org/tutorials/images/classification?hl=pt-br
3. Diferença entre funções de perda: https://medium.com/@shireenchand/choosing-between-cross-entropy-and-sparse-cross-entr
4. Parâmetros para a floresta aleatória: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html