Classificação do Fashion MNIST utilizando Regressão Logística, Softmax e Redes Neurais Artificiais

Augusto Krejci Bem-Haja
RA 227017
augustohaja@gmail.com

Lucas Vieira de Miranda
RA 211499
lucasvieirademiranda@gmail.com

I. Introdução

Este trabalho tem como objetivo implementar, testar e selecionar o melhor modelo entre os três algoritmos de classificação a regressão logística, a regressão softmax (regressão logística multinomial) e redes neurais artificias sobre o conjunto de dados chamado de *Fashion MNIST*.

O *Fashion MNIST* é um conjunto de dados de imagens em tons de cinza com dimensões de 28x28 divididos entre 10 classes onde cada uma representa uma peça de vestuário, além disso, possui 60000 exemplos para treinamento e 10000 exemplos para teste.

II. Implementação da Regressão Logística One-Vs-All

Ao implementar a regressão logística one-vs-all, buscou-se inspiração no *SGDClassifier* fornecido pela biblioteca *scikit-learn* que implementa esse algoritmo quando se configura o parâmetro *loss* com o valor "log", entretanto, na implementação fornecida pelo *scikit-learn* encontram-se diversos parâmetros ligados a regularização, e portanto, para simplificar a implementação esses parâmetros foram desprezados.

Em seguida implementou-se a classe *SGDClassifier* com a mesma interface apresentada pelo *scikit-learn*.

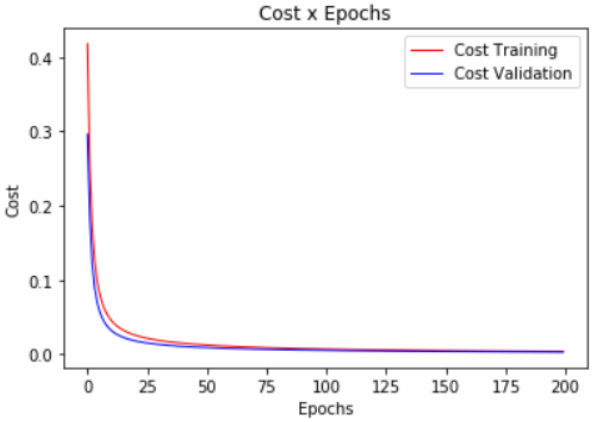
O método *partial_fit* é responsável pelo treinamento dos N (N classes) modelos gerados aplicando o algoritmo do gradiente descendente estocástico minimizando a função de custo *Cross Entropy Loss*, e recebe como parâmetros uma matriz de *features*, um vetor de *labels* e um vetor de *classes*.

O método *predict_prob* é responsável pelo cálculo das probabilidades de cada uma das classes através da aplicação da função logística de cada um dos modelos e as retorna na forma de um vetor.

O método *predict* é responsável pela previsão dos *labels* e tem como resultado a posição da maior probabilidade existente no vetor de probabilidades calculado pelo método *predict_prob*.

Para finalizar, utilizou-se um conjunto de dados sintéticos e o método *Train/Test Validation* utilizando 80% dos dados como conjunto de treinamento e 20% dos dados como conjunto de validação e a análise do gráfico de custo vs época para analisar se o algoritmo convergia corretamente, como pode ser visto na figura 1 [1].

Figure 1. Gráfico de Custo por Época para Regressão Logística One-Vs-All.



Fonte: produzida pelos autores

III. Implementação da Regressão Softmax

Ao implementar a regressão softmax, buscou-se inspiração nas interfaces fornecidas pelo *scikit-learn* e implementou-se a classe *SoftmaxRegression*.

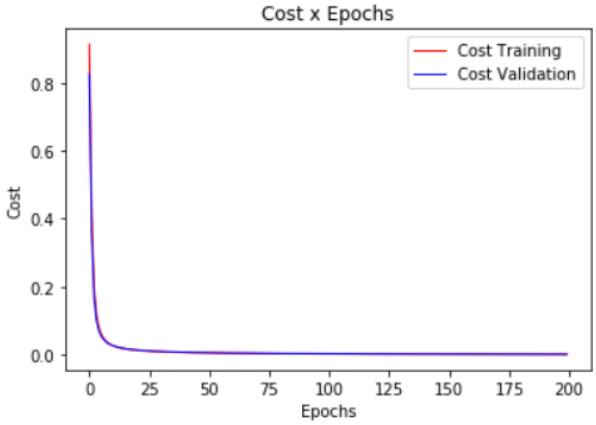
O método *partial_fit* é responsável pelo treinamento do modelo através do gradiente descendente estocástico minimizando a função de custo *Cross Entropy Loss* e os parâmetros de entrada são uma matriz de *features* e uma matriz de *labels* codificadas com *one-hot-encoding*.

O método *predict_prob* é responsável pelo cálculo das probabilidades de cada uma das classes através da aplicação da função softmax e as retorna na forma de um vetor.

O método *predict* é responsável pela previsão dos labels e tem como resultado a posição da maior probabilidade existente no vetor de probabilidades calculado através do método *predict_prob*.

Finalizando, utilizou-se um conjunto de dados sintéticos e o método *Train/Test Validation* utilizando 80% dos dados como conjunto de treinamento e 20% dos dados como conjunto de validação e a análise do gráfico de custo vs época para analisar se o algoritmo convergia corretamente, como pode ser visto na figura 2 [1].

Figure 2. Gráfico de Custo por Época para Regressão Softmax.



Fonte: produzida pelos autores

IV. Implementação da Rede Neural Artificial

Ao implementar a rede neural artificial, optou-se por fazê-lo da seguinte forma: Criou-se uma classe chamada *NeuralNetwork* contendo em seu construtor o número de nós nas camadas de entrada, primeira escondida, segunda escondida e saída, além disso, também é possível informar a taxa de aprendizado, as funções de ativação e derivadas das camadas escondidas, visando a facilidade na hora de testar diferentes funções de ativação. Na camada de saída utiliza-se a função de ativação softmax, uma vez que a rede neural será utilizada para classificação [2] [5]. A execução da inicialização dos pesos entre as camadas ocorre através da geração de números aleatórios entre -0.01 e 0.01 seguindo uma distribuição uniforme.

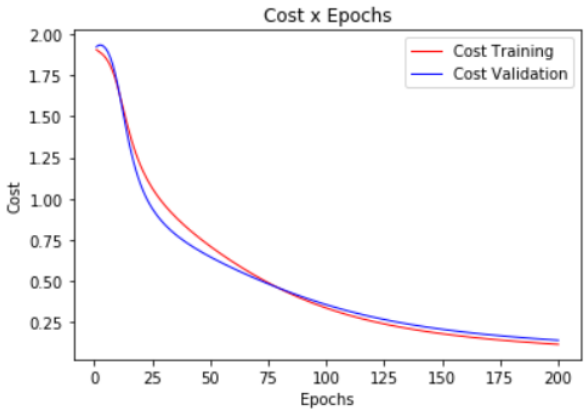
O método *partial_fit* é responsável pelo treinamento da rede neural através da execução do algoritmo *backpropagation*, e recebe como parâmetros uma matriz de *features* no qual cada linha representa um vetor de *features* e uma matriz de *labels* na qual cada linha representa um vetor de *labels* contendo os valores transformados por *one-hot-encoding*.

O método *predict_prob* é responsável pela execução do algoritmo forward propagation que tem como resultado o valor das probabilidades para cada saída da rede.

O método *predict* é responsável pela previsão dos *labels* e tem como resultado a posição da maior probabilidade existente no vetor de probabilidades calculado pelo método *predict_prob*.

Por fim, utilizou-se um conjunto de dados sintéticos e o método *Train/Test Validation* utilizando 80% dos dados como conjunto de treinamento e 20% dos dados como conjunto de validação e a análise do gráfico de custo vs época para analisar se o algoritmo convergia corretamente, como pode ser visto na figura 3 [1].

Figure 3. Gráfico de Custo por Época para Rede Neural.



Fonte: produzida pelos autores

V. Seleção do Modelo

A execução dos testes para a seleção do modelo consistiu em carregar os dados de treinamento do *Fashion MNIST* e na aplicação da técnica *K-Fold Cross Validation* com K = 5, utilizando 60 épocas em cada modelo e as *features* foram normalizadas utilizando a técnica *Min/Max Normalization* ajustada para oferecer valores entre 0.01 e 0.99, a ideia por trás disso é reduzir as multiplicações por 0 durante o treinamento da rede neural, ainda, escolheu-se o valor máximo de 255, pois independente da imagem, se ela for representada em tons de cinza os pixels só podem assumir valores entre 0 e 255.

No caso da regressão logística, após a execução do *k-fold*, obteve-se uma acurácia média de 0.8520166666666666.

No caso da regressão softmax, após a execução do *k-fold*, obteve-se uma acurácia média de 0.8553166666666666.

No caso da rede neural artificial executaram-se 6 modelos (testes), nos quais 2 utilizaram a função de ativação Logística (ou *Sigmoid*), 2 utilizaram a função de ativação ReLU (*Rectifier Linear Unit*) e 2 utilizaram a função de ativação TanH (Tangente Hiperbólica). Além disso, escolheu-se 15 neurônios na primeira camada escondida, com base na regressão logística *one-vs-all* que utiliza 10 classificadores logísticos obteve uma

acurácia de 0.85, e para segunda camada escondida escolheu-se 5 neurônios utilizando a lógica da metade das entradas.

Na primeira rede neural utilizou-se a função de ativação logística na camada escondida e a configuração de: 784 entradas, 15 neurônios na camada escondida e 10 neurônios na camada de saída; utilizando-se uma taxa de aprendizado de 0.001 e após a execução do *k-fold*, obteve-se uma acurácia média de 0.8654166666666668.

Na segunda rede neural utilizou-se a função logística nas camadas escondidas e a configuração de 784 entradas, 15 neurônios na primeira camada escondida, 5 neurônios na segunda camada escondida e 10 neurônios na camada de saída; utilizando-se uma taxa de aprendizado de 0.001 e após a execução do *k-fold*, obteve-se uma acurácia média de 0.8475333333333334.

Na terceira rede neural utilizou-se a função ReLU na camada escondida e a configuração de: 784 entradas, 15 neurônios na primeira camada escondida e 10 neurônios na camada de saída; utilizando-se uma taxa de aprendizado de 0.001 e após a execução do *k-fold*, obteve-se uma acurácia média de 0.5861333333333333.

Na quarta rede neural utilizou-se a função ReLU na camada escondida e a configuração de: 784 entradas, 15 neurônios na primeira camada escondida, 5 neurônios na segunda camada escondida e 10 neurônios na camada de saída; utilizando-se uma taxa de aprendizado de 0.001 e após a execução do *k-fold*, obteve-se uma acurácia média de 0.1.

Na quinta rede neural utilizou-se a função TanH na camada escondida e a configuração de: 784 entradas, 15 neurônios na primeira camada escondida e 10 neurônios na camada de saída; utilizando-se uma taxa de aprendizado de 0.001 e após a execução do *k-fold*, obteve-se uma acurácia média de 0.86195.

Na sexta rede neural utilizou-se a função TanH nas camadas escondidas e a configuração de: 784 entradas, 15 neurônios na primeira camada escondida, 5 neurônios na segunda camada escondida e 10 neurônios na camada de saída; utilizando-se uma taxa de aprendizado de 0.001 e após a execução do *k-fold*, obteve-se uma acurácia média de 0.8567166666666667.

Por fim, baseando-se na acurácia média de todos os modelos (Regressão Logística, Softmax e Redes Neurais), optou-se por escolher a primeira rede neural, ou seja, a rede que utiliza a função de ativação logística na camada escondida e possui a configuração de: 784 entradas, 15 neurônios na camada escondida e 10 neurônios na camada de saída; utilizando uma taxa de aprendizado de 0.001.

VI. Análise do Modelo

Após a seleção do modelo, instanciou-se um novo modelo com os mesmos parâmetros, carregou-se os dados de treino do *Fashion MNIST* e optou-se por utilizar a técnica *Train/Test Validation* com 80% dos dados como conjunto de treinamento e 20% dos dados destinados como conjunto de validação. Aplicou-se então a técnica de normalização *Min/Max Normalization* ajustada para oferecer valores entre 0.01 e 0.99, além disso, durante a normalização optou-se pelo valor máximo de 255, pois independente da imagem, se ela for representada em tons de cinza, seus pixels só poderão possuir valores entre 0 e 255. Em seguida, treinou-se o modelo utilizando técnica de regularização chamada de *Early Stopping* que consiste em escolher o modelo com o menor custo para o conjunto de validação e dessa forma evitar o *overfitting* [1].

Em seguida, carregou-se os dados de teste do *Fashion MNIST* e aplicou-se *Min/Max Normalization* de forma análoga a utilizada no conjunto de treinamento e calculou-se a acurácia obtendo-se o valor de 0.8715.

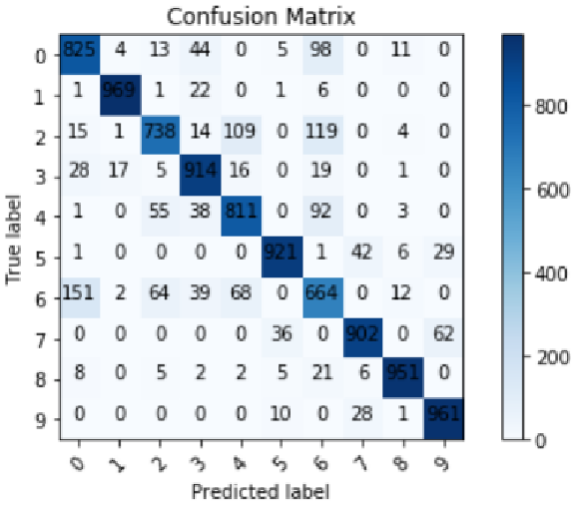
Continuando, plotou-se a matriz de confusão, que pode ser observada na figura 4, com o objetivo de extrair algumas métricas de avaliação, como por exemplo: *precision*, *recall* e *F1 Score* [2] [6]. Ao calcular o valor do *precision* médio entre as classes obteve-se o valor de 0.8715579011823589, indicando que entre os valores previstos como sendo de uma determinada classe 87% desses valores estão corretos e os outros 13% são falsos positivos, em seguida, calculou-se o valor do *recall* médio obteve-se o valor de 0.8714999999999999, indicando que entre os valores que realmente são de uma determinada classe 87% desses valores estão corretos e os outros 13% são falsos negativos, e por fim, calculando o *F1 Score* que é a média harmônica entre o *precision* e o *recall* obteve-se o valor de 0.8713223504069063, ou seja, a proximidade do valor do F1 Score com o *precision* e o *recall* se justifica pela semelhança entre os valores, e quanto mais próximo de 1 esse valor, melhor é o modelo, pois produzirá menos falsos positivos e falsos negativos [6].

VII. Conclusões

Durante o implementação da regressão logística *one-vs-all* e da regressão softmax observou-se que elas são muito semelhantes tanto na implementação quanto nos resultados obtidos, isto se deve ao fato da regressão softmax ser uma generalização da regressão logística para se trabalhar com problemas envolvendo mais de duas classes sem a necessidade de combinar classificadores binários [1].

Ao realizar a implementação da rede neural observou-se que a forma com que os pesos são inicializados ajuda na convergência do modelo, e optou-se por utilizar uma inicialização utilizando uma distribuição normal com valores entre -0.01 e 0.01, pois assim obteve-se melhores resultados.

Figure 4. Matriz de confusão plotada a partir do modelo selecionado. .



Fonte: produzida pelos autores

Já durante os testes realizados para a seleção do modelo dois pontos interessantes foram observados. O primeiro é que ao manter os mesmos parâmetros e incluir uma nova camada escondida obtém-se um pior desempenho nos casos das redes que utilizam a função logística e TanH, isso acontece devido ao problema conhecido como desaparecimento do gradiente (*Vanishing Gradient*), que consiste no fato de que conforme o gradiente é propagado da camada de saída para as camadas mais internas ele tende a se tornar menor, ou seja, as camadas mais próximas da camada de entrada tendem a aprender mais devagar do que as camadas mais próximas da camada de saída [5]. O segundo é que a utilização de ReLU junto a *Cross Entropy Loss* pode levar ao problema de explosão do gradiente (*Gradient Blow Up*), que consiste no cálculo de gradientes muito grandes, em alguns casos os valores do gradiente são tão grandes que geram valores NaNs (*Not a Number*) fazendo com que a rede pare de aprender. Uma forma de evitar a explosão do gradiente é reduzir a taxa de aprendizado, entretanto será necessário um maior número de épocas [3].

Apesar da utilização de poucas épocas (60), o modelo selecionado obteve um desempenho consideravelmente alto analisando-se pelo *F1 Score*, entretanto é possível que variando-se o número de épocas e a taxa de aprendizado obtenham-se valores melhores [6].

Além disso, no futuro, pode-se utilizar termos de regularização em todos os modelos para reduzir o *overfitting* e tentar obter desempenhos ainda melhores [1] [2] [5].

REFERÊNCIAS

[1] Géron Aurélien. "Hands-On Machine Learning with Scikit-Learn and TensorFlow". Disponível em <https://www.safaribooksonline.com/library/view/hands-on-machine-learning/9781491962282/ch04.html?orpq>. Acesso em 02/10/2018.

[2] ANDREW NG (United States). Stanford University. Machine Learning. [2018]. Disponível em: <https://www.coursera.org/learn/machine-learning>. Acesso em: 02/10/2018.

[3] Brownlee Jason (United States). "A Gentle Introduction to Exploding Gradients in Neural Network". Disponível em: <https://machinelearningmastery.com/exploding-gradients-in-neural-networks/>. Acesso em: 02/10/2018.

[4] Rasul Kashif and Xiao Han. Zalando Research. Disponível em: <https://research.zalando.com/welcome/mission/research-projects/fashion-mnist/>. Acesso em: 02/10/2018.

[5] Avila, Sandra. Artificial Neural Networks. 11 de setembro de 2018. Notas de Aula.

[6] Avila, Sandra. Testing and Error Metrics. 28 de agosto de 2018. Notas de Aula.