Regressão Logística e Redes Neurais

Tamara Martinelli de Campos
RA 157324
tamaramcampos02@gmail.com

João Vítor B. Silva
RA 155951
joaovbs96@gmail.com

1. Introdução

Nas atividades desenvolvidas neste projeto, propomos e estudamos modelos para resolução de um problema de classificação por meio do uso de regressão logística, com as abordagens: regressão logística one vs all, regressão logística multinomial com uso da função de ativação softmax, e redes neurais com uma e duas camadas escondidas, com funções de ativação diversas.

2. Descrição da Base de Dados

Para o desenvolvimento do projeto, foi utilizada a base de dados *Fashion-MNIST*[1], composta por 70.000 amostras de imagens de artigos de vestuário, de 28x28 pixels em escala de cinza, sendo cada um dos 784 pixels uma feature. O target é dado por *Label*, sendo um número de 0 à 10. Cada imagem é classificada como um tipo de roupa, indicada pela *label*, como Dress, representado pela *label* 3.

Antes da execução dos modelos, tanto para treino como teste, normalizamos a base de dados de forma diferente entre os modelos. Em regressão logística one vs all e em multinomial, utilizamos a normalização min-max do sklearn. Já nos modelos de rede neurais, normalizamos por meio de divisão dos valores das features pelo valor máximo apresentado, no caso 255.0, pois as imagens são monocromáticas. Em ambos os casos as features ficaram escaladas de 0 a 1. Também utilizamos a redução de dimensionalidade por PCA, através da implementação da biblioteca SciKit-Learn, mantendo 98% de variância.

Separamos a base de dados em 3 conjuntos: treino, com 48.000 amostras, validação, com 12.000 amostras (80%, 20%), e teste, com 10.000 amostras.

3. Modelos Propostos e Resultados Obtidos

Quatro modelos distintos são propostos, dois por uso de regressões logísticas e dois por meio de redes neurais. Para avaliação dos modelos obtidos, executamos os modelos para diversos valores de learning rate, sempre após a execução de 1000 iterações.

Utilizamos, também, a métrica do F1-Score da biblioteca SciKit-Learn, com ponderamento 'micro', que calcula métricas globalmente contando o total de verdadeiros positivos, falsos negativos e falsos positivos, o que é desejado em problemas de classificação como o estudado.

3.1 Regressão Logística One vs All

Para o modelo de regressão Logística One vs All, os elementos do target são mapeados para um formato binário, com a classe de interesse sendo igual à 1 em cada execução e as outras classes sendo igual a 0, obtendo 10 vetores de thetas distintos. Aplicamos a função sigmoid nos thetas obtidos e selecionamos como valor predito, para cada observação, a classe de maior probabilidade.

Durante a implementação, diversos problemas foram encontrados. Frequentemente obtivemos uma matriz de resultado contendo apenas zeros, uns ou 'nan's. Identificamos que para os resultados zeros ou uns, a sigmoid estava nos extremos e o modelo não estava aprendendo corretamente. Resolvemos esse problema

[1] https://github.com/zalandoresearch/fashion-mnist

normalizando os dados do database. Para 'nan's, vimos que em alguns casos, no cálculo do erro, o valor da nossa hipótese era 0, e o cálculo do log de 0 é indefinido. Resolvemos esse problema ao adicionar um limiar pequeno aos termos de log.

Tabela 1 - F1-Score calculado para modelo de regressão logística One vs all para diversos valores de learning rate

Learning Rate	F1-Score	
	Validação	Teste
0.1	77,13%	76,85%
0.01	74,89%	74,85%
0.001	43,61%	44.77%

Analisando os resultados obtidos, vemos que o modelo One vs All proposto é mais preciso com o uso de learning rate igual a 0.1 para 1000 iterações.

3.2 Regressão Logística Multinomial

O modelo de Regressão Logística Multinomial é semelhante ao One vs All. A principal diferença é o uso de one hot encode no target, possibilitando treinamento simultâneo das 10 classes, através de uma função softmax, o que diminui o tempo de execução.

Houveram alguns problemas com a convergência do modelo durante sua implementação. Após várias pesquisas e testes, utilizamos a função de softmax com regularização por divisão da matriz de entrada pelo seu maior valor (stable softmax). Por ser uma função exponencial, números altos resultam em valores 'nan'. A função de custo também foi alterada para a 'cross entropy', presente no slide da aula, pois utilizando a função de custo utilizada no modelo anterior, o modelo multinomial não convergia.

Tabela 2 - F1-Score calculado para modelo de Regressão Logística Multinomial para diversos valores de learning rate

Learning Rate	F1-Score	
	Validação	Teste
0.1	77,46%	77,36%
0.01	74,98%	74,89%
0.001	75,03%	74,94%

Analisando os resultados obtidos, vemos que o modelo Multinomial proposto é, assim como o One vs All, mais preciso com o uso de learning rate igual a 0.1 para 1000 iterações

3.3 Rede Neural com Uma Camada Escondida

Com relação ao modelo de Rede Neural com Uma Camada Escondida, inicialmente aplicamos a função sigmoid para ativação de todas as camadas em uma abordagem estilo One vs All. O tamanho da camada escondida foi escolhido arbitrariamente como 64 e os pesos inicializados com valores aleatórios no intervalos [0, 1).

Um dos problemas encontrados na implementação do modelo foi a obtenção de resultados apenas 1, ou 0, para todas as amostras, o que fazia com que a acurácia do modelo fosse de apenas 10%, ou seja, o número de classes 0 ou 1 que havia no dataset. Concluímos que isso ocorre nos casos em que a exponencial do sigmoid era muito pequena, tendendo a zero, gerando a divisão de 1 por 1 e impedindo que a rede aprendesse. Já para valores somente 0, concluímos que estava ocorrendo o problema de *Vanishing Gradient*, onde os valores dos pesos tendiam a 0 e esse valor se propagava pela rede.

Para a resolução desses problemas, passamos a utilizar ReLU na camada escondida e Softmax na camada de saída, o que se mostrou bem sucedido. Vale notar que, para que o modelo convergisse, foram utilizados valores muito pequenos de learning rate de 10^{-6} a 10^{-8}

Tabela 3 - F1-Score calculado para modelo de Redes Neurais com Uma Camada Escondida para diversos valores de learning rate

Learning Rate	F1-Score	
	Validação	Teste
0.000001	78.21%	78.57%
0.0000001	60.36%	60.30%
0.00000001	20.68%	21.51%

Analisando os resultados obtidos, vemos que o modelo com uma camada escondida proposto é mais preciso com o uso de learning rate igual à 10^{-6} para 1000 iterações.

3.4 Rede Neural com Duas Camadas Escondidas

A implementação do modelo de rede neural com duas camadas escondidas é bastante similar ao modelo de uma camada. Mantivemos o número de 64 neurônios para cada uma das camadas escondidas.

Durante a execução dos modelos, percebemos que a convergência é mais lenta do que nos abordagens anteriores, e quanto menor o learning rate maior o número de iterações necessário para convergir. Mantivemos o limite de 1000 iterações, para analisar os modelos em condições igualitárias. O treinamento do modelo com um número de iterações maior iria requerer, também, um tempo elevado de execução. Uma das possíveis razões para tal é o nível de complexidade da rede, maior do que dos modelos anteriores.

Para validar o modelo de duas camadas, utilizamos quatro funções de ativação diferentes: TanH, Sigmoid, ReLU e Leaky ReLU. No uso de ReLU e Leaky ReLU nas camadas escondidas, tivemos problemas com a precisão dos resultados, pois os números eram muito grandes e explodiam, necessitando que o retorno das funções de ativação fossem escalados no intervalo [0, 1). Para a função softmax, aplicada na camada de saída, não obtivemos problemas.

Para as funções TanH e Sigmoid, não tivemos problemas em termos de execução, mas podemos ver que a acurácia de ambos em relação a ReLU e a Leaky ReLU é menor. Tanto as funções de ativação tanh quanto sigmoid sofrem com o problema de vanishing gradient, e ficam presas em intervalos [-1,1] e [0,1], respectivamente, o que pode fazer com que a rede fique mais propensa a ficar "presa" durante o treinamento.

Tabela 4 - F1-Score calculado para modelo de Redes Neurais com Duas Camadas Escondidas para diversos valores de learning rate e cada função de ativação utilizada

Learning Rate	F1-Score			
	Tanh		Sigmoid	
	Validação	Teste	Validação	Teste
0.0001	13,49%	13,45%	77,49%	76,84%
0.00001	27,76%	26,63%	59,07%	59,20%
0.000001	23,71%	24,24%	18,70%	19,09%

Learning Rate	F1-Score			
	ReLU		Leaky ReLU	
	Validação	Teste	Validação	Teste
0.0001	86,32%	85,76%	86,60%	86.46%
0.00001	78,36%	78,29%	68,20%	67,66%
0.000001	47,92%	47,51%	56,00%	55,24%

Analisando os resultados obtidos, vemos que o modelo com duas camadas escondidas é mais preciso com o uso de learning rate igual à 10^{-4} para 1000 iterações e utilizando a função de ativação Leaky ReLU nas camadas escondidas e softmax na camada de saída.

4. Conclusão

Analisando os testes efetuados, podemos concluir que o melhor modelo dentre todos os propostos é o de Redes Neurais com Duas Camadas Escondidas com o uso de função de ativação Leaky

ReLU nas camadas escondidas e Softmax na camada de saída e Learning Rate igual a 10^{-4} para 1000 iterações. Dentre todos os modelos e combinações testadas, foi o que obteve maior F1-Score, tanto para o conjunto de validação quanto para o de teste. Na Figura 1, vemos o gráfico de erro por número de iterações para o modelo obtido e, na Figura 2, a matriz de confusão e seu HeatMap.

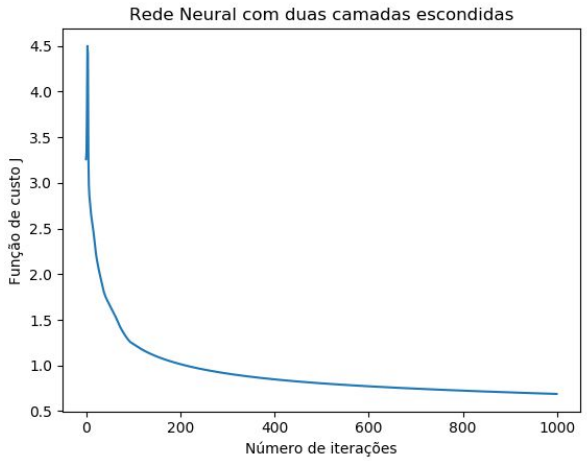

Figura 1 - Gráfico de Erro por número de iterações para o modelo de Redes Neurais com Duas Camadas Escondidas, executado com 1000 iterações, learning rate 0.0001 e função Leaky ReLU nas camadas escondidas e softmax na camada de saída para conjunto de teste.

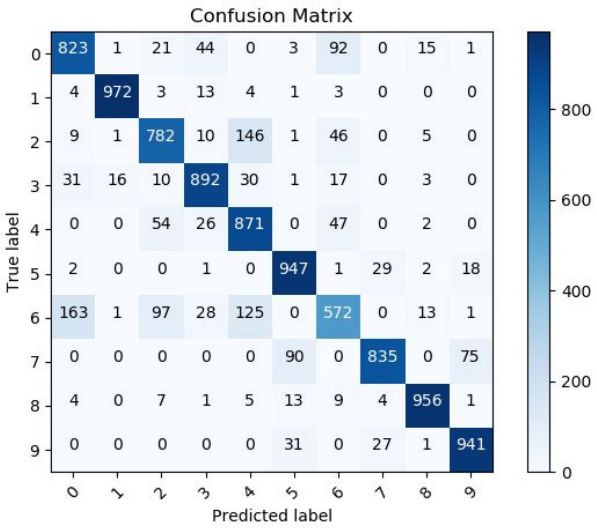

Figura 2 - HeatMap da matriz de confusão do modelo de Redes Neurais com Duas Camadas Escondidas, executado com 1000 iterações, learning rate 0.0001 função Leaky ReLU nas camadas escondidas e softmax na camada de saída para conjunto de teste.

Analisando os resultados obtidos, uma das razões que pode ter contribuído para que o modelo de Duas Camadas fosse o mais preciso, é o fato de que sua arquitetura é mais complexa e robusta do que a de Uma Camada. De maneira geral, no entanto, dado um número suficiente de iterações, acreditamos que qualquer um dos modelos apresentados poderia alcançar níveis semelhantes de precisão na classificação. Note que, como os dois modelos de Redes Neurais são inicializados de maneira aleatória, seus resultados podem variar de maneira mínima. Ainda assim, durante nossos testes o modelo escolhido se mostrou consistentemente melhor que os outros implementados.

Possíveis melhorias dos resultados dos outros modelos implementados poderiam ser obtidas ao se efetuar o treino por um número maior de iterações, em especial para o uso da função TanH no modelo de Duas Camadas, que apresentou níveis de F1-Score baixos, apesar de estar, efetivamente, aprendendo. Via de regra, aumentar o número de iterações em um fator de 10 mantendo o learning rate estável mostrou resultados satisfatórios. Ao mesmo tempo, essa abordagem pode resultar em overfitting se utilizada em demasia.

Vemos também que para todos os modelos o maior learning rate obteve melhor resultado. Acreditamos que isso aconteça pois com learning rate menor, o modelo dá passos menores e demora mais para convergir. Logo, para 1000 iterações, é esperado que o maior learning rate apresente melhor resultado. Acreditamos que com um número maior de iterações, learning rates menores possam ser mais satisfatórios.

5. Execução e Implementação

Cada modelo proposto foi implementado em um arquivo fonte separado, por motivos de organização. Os comandos de execução dos mesmos encontram-se no arquivo `readme.txt`, incluído junto deste relatório.

Todas as implementações utilizaram a linguagem de programação Python e bibliotecas como NumPy e SciPy, sempre recorrendo a operações vetoriais para manipulação dos dados, a fim de reduzir o tempo de processamento.