Relatório Trabalho 1 - Regressão Linear

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

Giovanni Bertão
RA 173325
g173325@dac.unicamp.br

I. Introdução

Atualmente pode-se fazer muito com dados. *Data science* está em alta e lidera um dos setores profissionais mais promissores para o futuro. Sua principal ferramenta são os dados, contudo, em si só não representam o cerne cuja a profissão se sustenta, mas sim as conclusões e modelos que se podem obter através deles. De forma simplificada as atividades de um *data scientist* consistem em: coleta de dados, filtragem dos dados, elaboração de modelos, testes experimentais dos modelos e avaliação dos resultados. Tendo em mente essa receita básica, um *data scientist* consegue, de um *dataset*, formular um modelo capaz de prever comportamentos futuros ou de elementos que não compunham o *dataset* original. Tais modelos estão relacionados a diversas áreas do conhecimento: saúde, física, estatística, arte, botânica, computação, matemática, economia, direito, dentre outros. Existem diversas técnicas para se trabalhar com dados e elaborar modelos, cada qual tem sua especificidade e são dependentes do tipo do *dataset* analisado. Uma das técnicas mais conhecidas é a técnica de regressão linear, que busca, dado um *dataset* e uma função de hipótese inicial, obter os parâmetros da função, a fim de permitir que a mesma consiga descrever o *dataset* em valores contínuos. Além disso, para obter os parâmetros é aplicado o algoritmo de descida do gradiente, que atualiza os valores dos parâmetros a cada iteração, esperando que os valores dos parâmetros convirjam e se estabilizem.

Nesse contexto, este trabalho trata-se da aplicação de regressão linear no conjunto de dados *"Metro Interstate Traffic Volume Data Set"* [1] para criar um modelo que possibilitasse prever o volume de trafego de um metro inter-estadual.

II. Atividades

Primeiramente começamos com tentativas de implementação simples, criamos dados sintéticos, utilizamos o famoso exemplo de tamanho e preço de casas [2], com apenas 10 linhas e valores fáceis para realizar a regressão. Assim, foi possível aprender como utilizar as bibliotecas, métodos necessários para importar os dados, como manipular e visualizar os dados e o mais importante, verificar facilmente se a nossa implementação estava correta. Tudo isso olhando implementações de regressão linear já prontas na internet e criando a nossa própria do zero, adequando ao nosso conjunto de dados minúsculo. Após isto, basicamente já estávamos mais familiarizados com as ferramentas e já tínhamos uma ideia de como começar o trabalho com um volume maior de dados.

As aulas [3] foram muito importantes para sabermos o que fazer com os dados mais complexos, como tratar features fora de escala e como transformar features categóricas. Diferentes métodos de descida de gradiente, equações mais complexas, como avaliar o erro nos diferentes conjuntos e se há *underfitting* ou *overfitting* foram coisas essenciais para trabalharmos com um conjunto de dados mais complexo, portanto o comparecimento às aulas ajudou a sair do básico para um entendimento mais abrangente das técnicas de regressão linear e consequentemente uma implementação com melhores resultados.

III. Dataset

O *dataset* consiste em 48204 linhas e 9 colunas, sendo uma delas a coluna *traffic_volume*, que é o valor que queremos predizer. Os dados do *dataset* consistem em dados cedidos pelo Departamento de Transporte de Minesota [4] e dados climáticos do *OpenWeatherMap* [5]. A partir deste dataset original, embaralhamos as linhas e criamos outros três novos *datasets*, que representam nosso conjunto de treino, validação e teste, com porcentagens de 80, 10 e 10, em relação ao total de linhas original, respectivamente.

IV. Solução Proposta

Para gerar um modelo com base no *dataset* a nossa solução consiste em aplicar a técnica de regressão linear usando o algoritmo de descida do gradiente. Para *features* categóricas decidimos realizar a técnica de *One Hot Encoding*, por conta disso, acabamos com um total de 57 *features*, entretanto esse número caiu para 54 quando notamos que haviam algumas *features* que não se correlacionavam tanto com o nosso *target*. Também decidimos implementar normalização para evitar problemas com escala de valores, ou seja, salvas as partes onde explicitamos a conversão para os valores originais, todos os resultados obtidos são em relação à variáveis normalizadas. Sendo assim, esta seção detalha como as técnicas são empregadas para compor a solução.

A. Hipótese

A primeira etapa para realizar a regressão linear é definir uma hipótese a ser testada e avaliada como modelo para o *dataset* utilizado. A função de hipótese é definida como o produto escalar de dois vetores: o vetor x que contém os valores das *features* e o vetor θ que contém o valor dos parâmetros associados a cada *feature*. Dessa forma, nossa hipótese é expressa pela Equação 1. Considerando os dados

pós-processamento identificamos 54 *features* logo a dimensão do nosso vetor θ é de 54x1.w

$$h_\theta(x) = x \cdot \theta \tag{1}$$

B. Função de Custo

Para aplicar a descida do gradiente é necessário definir uma função de custo ou função *Loss* que será responsável por atualizar os valores de θ a cada iteração esperando que a função de custo converja para o mínimo global do modelo. Para tanto, a função de custo $J(\theta)$ utilizada neste trabalho consiste na diferença quadrática do valor previsto$h_\theta(x_i)$ para o valor referência y_i como mostra a Equação 2.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 \tag{2}$$

C. Função Gradiente

Utilizamos os valores do gradiente da função de custo J para permitir que o valor de θ seja atualizado a cada iteração. O gradiente nada mais é que um vetor de cada derivada parcial de J em relação a cada um dos parâmetros de θ. A Equação 3 apresenta o gradiente utilizado e que pode ser sintetizada na relação da Equação 4.

$$\nabla J(\theta) = \left(\frac{\partial J(\theta)}{\partial \theta_1}, ..., \frac{\partial J(\theta)}{\partial \theta_{54}}\right) \tag{3}$$

$$\nabla J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i) \cdot x_i \tag{4}$$

D. Taxa de aprendizado

Para atualizar os valores de θ não podemos simplesmente utilizar diretamente o valor resultante do gradiente. Pois, isso pode nos afastar do melhor θ que reduz a equação de custo para o menor valor possível. Uma interpretação geométrica seria de que temos a função de custo como uma região em um plano e queremos partir de um ponto e encontrar o mínimo global dessa região. Para tanto, devemos caminhar na região em passos não muito largos para permitir que exploremos a região de forma suficiente, mas também não queremos caminhar em passos tão pequenos, pois queremos garantir chegar no mínimo global em tempo hábil. Diante disso, antes de atualizar o o valor de θ o gradiente é multiplicado por um taxa α conhecida como taxa de aprendizado.

E. Descida do Gradiente

Apresentadas as equações anteriores, O Algoritmo 1 apresenta o processo de descida do gradiente. No Algoritmo 1 generalizamos a condição de parada. Tal condição pode ser adotada como: "Enquanto não convergiu", "Repita X épocas", "Enquanto o erro médio for maior do que Y", etc. Em nosso experimento utilizamos como condição de parada o número máximo de épocas, que inicialmente foi de 10.000.

Algoritmo 1 Descida do Gradiente

while condição **do**
  $\theta = \theta - \alpha\nabla J(\theta)$
end while

V. Experimentação e Discussão

Durante a execução do experimento diversas propriedades de execução foram alteradas. Esta seção explicita as alterações realizadas em cada experimento e discute os dados resultantes.

A. Configuração Inicial

A primeira execução levou em consideração os seguintes parâmetros:

- θ: Vetor 54x1 gerado de forma aleatória;
- Condição de parada: Repetição de 10.000 épocas;
- Taxa de aprendizado: 0.001;

A Figura 1 mostra a relação da função de custo para cada época iterada. O processo todo levou aproximadamente 17 minutos. Da Figura 1 é possível observar que os dados de treino e de validação estão convergindo, porém de forma lenta. Entretanto, o erro dos dois conjuntos estão bem próximos e, por fim, o menor e maior erro absoluto entre o valor predito e o valor esperado dos dados de validação foram de respectivamente 0 e 549.21.

Analisando esses resultados, nota-se que ambos os gráficos estão convergindo de maneira similar logo é possível deduzir que o número de épocas é suficiente. Contudo, a lentidão de convergência e a magnitude do maior erro ser consideravelmente grande implica que a taxa de aprendizado está muito pequena.

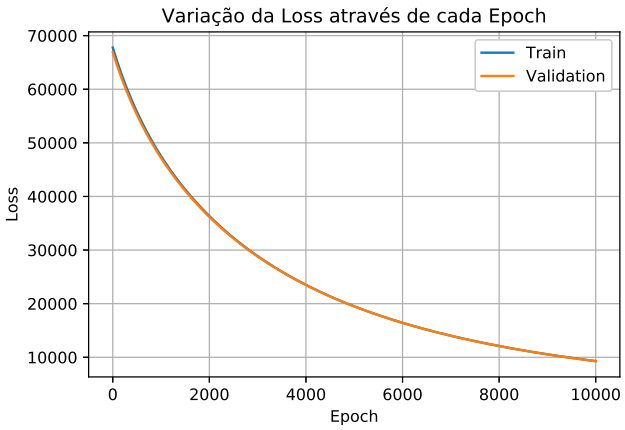

Figura 1. Relação entre a *Loss* e cada época iterada para a execução inicial

B. Segunda Configuração

Os parâmetros da segunda execução foram:

- θ: Vetor 54x1 gerado de forma aleatória;
- Condição de parada: Repetição de 1.000 épocas;
- Taxa de aprendizado: 0.001;

A Figura 2 mostra a relação da função de custo para cada época iterada. O processo todo levou aproximadamente 106 segundos. Da Figura 2 é possível ver melhor a diferença entre o conjunto de treino e validação, além de que o maior e o menor erro absoluto para os dados de validação foram de 861.71 e 0.08, respectivamente.

Desses resultados, sabemos que a velocidade de convergência é igual ao da primeira configuração, além disso, o tempo de execução caiu muito, contudo, o maior erro aumentou. Esses resultados, são impactos claros da redução do número de épocas sem aumentar a taxa de aprendizado, pois convergindo à mesma velocidade e executando por menos tempo, não é possível chegar a um erro máximo menor.

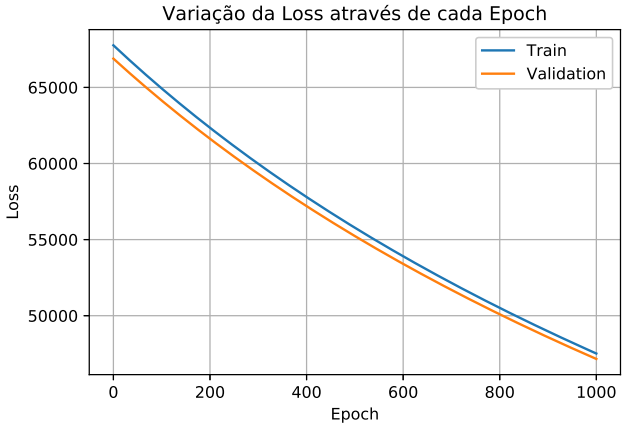

Figura 2. Relação entre a *Loss* e cada época iterada para a segunda configuração

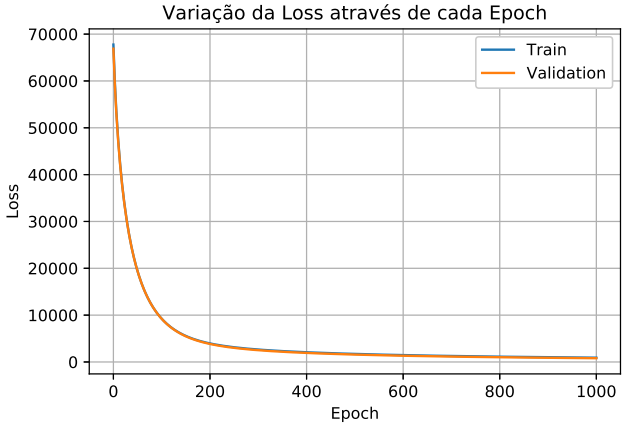

Figura 3. Relação entre a *Loss* e cada época iterada para a terceira configuração

C. Terceira Configuração

Os parâmetros dessa execução foram:

- θ: Vetor 54x1 gerado de forma aleatória;
- Condição de parada: Repetição de 1000 épocas;
- Taxa de aprendizado: 0.1;

A Figura 3 mostra a relação da função de custo para cada época iterada. O processo todo levou aproximadamente 105 segundos. Da Figura 3 é possível observar que os dados de treino e de validação convergiram. Além disso, o maior e o menor erro absoluto para os dados de validação foram de 441.27 e 0, respectivamente.

Dada a redução da taxa de aprendizado, sabíamos que talvez o erro poderia começar a aumentar em certo ponto. Contudo, os gráficos mostram que ambos os dados de treino e validação convergiram e obtivemos uma redução na taxa de erro absoluto. Portanto, comparado às duas configurações anteriores, obtivemos melhores resultados em melhor tempo.

D. Usando Sklearn

A biblioteca do `sklearn` [6] é uma maneira simples de coletar, analisar dados e utilizar algoritmos de aprendizado de maquina. Uma de suas funcionalidades é a função `SGDRegressor` [7] que gera um modelo linear baseado no algoritmo de descida do gradiente estocástico. Um de nossos experimentos consistiu em executar essa função e comparar os resultados com os obtidos pelo nosso melhor modelo. Para executar o `SGDRegressor` utilizamos como parametros da função os seguintes valores:

- `learning_rate = "constant"`: Torna a taxa de aprendizado constante entre todas as épocas.
- `eta0 = 0.1`: Determina o valor da taxa de aprendizado.
- `max_iter = 1000`: Determina o número máximo de épocas.

Após a execução da função obtivemos que o menor e maior erro absoluto no conjunto de validação foram respectivamente de 0 e 0.0011. Nota-se que o maior erro desse método é muito menor do que o menor erro encontrado pela nossa implementação. A engenharia das *features* e a escolha do θ que o `SGDRegressor` realiza são melhores do que as nossas, consequentemente levam o algoritmo a um erro muito menor e provavelmente um tempo menor de execução.

E. Aplicando técnica de Mini-Batch

Visto a enorme diferença do nosso melhor modelo com o resultado gerado pelo `sklearn` decidimos implementar a técnica de Mini-Batch usando a seguinte configuração:

- Número de épocas: 1000
- Tamanho do *Batch*: 50
- Taxa de aprendizado: 0.1

A Figura 4 mostra a relação da função de custo para cada época iterada. O processo de aprendizado levou 29 segundos. Da Figura 4 nota-se que a velocidade com que o algoritmo converge é muito maior. Além disso, o maior e o menor erro encontrado no conjunto de validação foi de 4.01 e 0, respectivamente.

Diante desses resultados, o *Mini-Batch* obteve melhor tempo e resultado em comparação com a técnica *Vanilla*. Contudo, os resultados ainda são inferiores quando comparados aos resultados do `sklearn`.

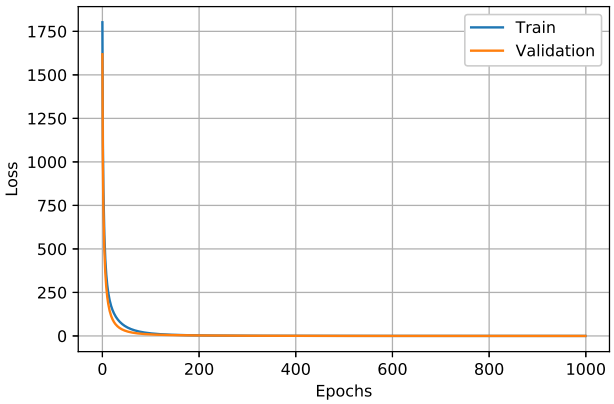

Figura 4. Relação entre a *Loss* e cada época iterada para o algoritmo de *Mini-Batch*

Como criamos nossa própria normalização e o *Mini-Batch* foi o melhor modelo não analítico implementado do zero, resolvemos verificar o erro absoluto médio entre os valores preditos e reais nos conjuntos de treino e validação ao longo das épocas. Para isso, a cada época, desfizemos a normalização e capturamos o erro médio entre os dois conjuntos, obtendo um erro médio na casa de dezenas para as ultimas 10 épocas e um resultado satisfatório, que pode ser conferido na Figura 5.

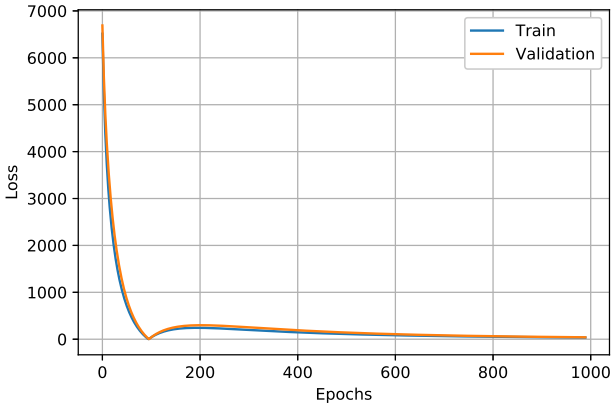

Figura 5. Relação entre erro absoluto e cada época iterada à partir da décima época no algoritmo de *Mini-Batch*

VI. Usando a Função Normal

Existe a forma analítica de resolver o problema de regressão. Para tal feito, é necessário utilizar a função normal. De maneira simplificada, a função normal é um conjunto de operações matriciais que devolvem a resposta analítica de θ. A função normal esta descrita na Equação 5 em que X é o conjunto de *features* e y a resposta esperada.

$$\theta = \left(X^T X\right)^{-1} \left(X^T y\right) \tag{5}$$

Aplicamos a função normal em nosso conjunto de treino e obtivemos o θ analítico em menos de 1 segundo. Em seguida, usamos o θ obtido no conjunto de validação e identificamos o menor e o maior erro absoluto como sendo 0 e 0.0229 respectivamente. Desses resultados é possível comparar a eficiência e superioridade do método da função normal em relação aos métodos criados do zero.

VII. Desempenho no Conjunto de Teste

Finalizados as etapas de treino e validação, o ultimo passo consiste em analisar os parâmetros encontrados no conjunto de teste. A Tabela I sumariza o resultado de cada método.

Tabela I
Sumarização dos resultados obtidos no conjunto de testes separados por: Método utilizado, menor erro absoluto e maior erro absoluto

Método	MIN |Error|	MAX |Error|
Vanilla	0	483.10
Mini-Batch	0	1.4657
Função Normal	0	0.0184
Sklearn	0	0.0013

VIII. Conclusão e Trabalhos Futuros

Apresentados os resultados e os métodos utilizados para a obtenção dos mesmos. Concluímos que a melhor solução foi obtida através da aplicação do método Sklearn. Um resultado plausível, tendo em vista que é um um método bem famoso e muito tempo e esforço foram gastos aperfeiçoando-o. Além do resultado final, foi possível observar parâmetros gulosos (por exemplo: colocar 10.000 épocas ou um valor muito pequeno de taxa de aprendizado), não necessariamente geram a melhor solução em questão de recursos e resultados, como observado nos experimentos do método *Vanilla* e de *Mini-Batch*.

Desta forma, é necessário experimentar diferentes valores para cada um dos parâmetros e analisar os resultados obtidos. Por fim, nesse trabalho, utilizamos a técnica de regressão linear *Vanilla*, *Mini-Batch*, *Função Normal* e a função `SGDRegressor` da biblioteca `sklearn`. Considerando a natureza do problema, pode-se atacar o mesmo com diferentes estratégias. Nossa solução se concentrou na utilização das *features* do *dataset* que não incluiam temperatura, data e hora, pois percebemos que utilizando todas as *features*, os resultados normalizados variavam pouco, mas quando restaurávamos a escala dos dados o resultado estava muito volátil.

Referências

[1] UCI, "Metro interstate traffic volume data set." https://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume.
[2] V. Valkov, "Predicting house prices with linear regression | machine learning from scratch (part ii)." https://towardsdatascience.com/predicting-house-prices-with-linear-regression-machine-learning-from-scratch-part-ii-47a0238aeac1.
[3] S. Avila, "Linear regression." http://www.ic.unicamp.br/~sandra/pdf/class/2019-2/mc886/2019-08-14-MC886-Linear-Regression.pdf.
[4] M. D. of Transportation, "Mn department of transportation." https://www.dot.state.mn.us/.
[5] O. W. Map, "Open weather map." https://openweathermap.org/.
[6] S.-K. Learn, "Sci-kit learn." https://scikit-learn.org/.
[7] S.-K. Learn, "sklearn.linear_model.sgdregressor." https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html.