Estudos em Regressão Linear

Matheus Mortatti Diamantino
RA 156740
matheusmortatti@gmail.com

José Renato Vicente
RA 155984
joserenatovi@gmail.com

I. Introduction

Este projeto teve como intuito o estudo prático do método de regressão linear em Machine Learning. Foram utilizado os algoritmos de *Gradient Descent* conhecidos como *Stochastic, Batch, Mini Batch e Equação Normal*, de modo a compara-los em termos de complexidade e acurácia.

Foi feito um estudo de predição de preço de diamantes, utilizando uma base de dados com 54000 exemplos, em que são apresentados seus preços e nove features como tamanho, cor e número de quilates.

II. Activities

A. Regressão Linear

Regressão Linear é um método muito conhecido de Machine Learning, utilizado para predizer o valor de uma variável dependente baseado em valores de variáveis independentes. Essa regressão é chamada linear porque se considera que a relação da resposta às variáveis é uma função linear de alguns parâmetros. Desta forma, dado um vetor Theta de tamanho igual ao número de features, cujo valor queremos determinar, temos que:

$$PreçoAlvoEsperado = \sum_{i=1}^{m} \theta_i X_i = h_\theta(x) \tag{1}$$

Em que X é um vetor com os valores das features para um dado diamante, cujo preço queremos determinar.Para encontrar esse valor de Theta, utilizaremos alguns algoritmos e compararemos os resultados obtidos com cada um.

Cada algoritmo utilizado é baseado no método de *Descrida de Gradiente (ou Gradient Descent)*. Este é um método utilizado para achar o ponto mínimo de uma função, aproximando gradativamente seu valor até um ponto quando não é possível ser diminuido mais (i.e. a derivada da função neste ponto é zero). Este método consegue apenas achar mínimos locais e, com isso, não é garantido que o resultado obtido é o melhor para o dado problema.

Para medirmos a eficácia do algoritmo, utilizamos uma *Função de Custo* que nos diz o quão perto do resultado desejado estamos, dado um conjunto de dados. Esta função é definida por:

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

Como queremos minimizar a função de custo, queremos que cada passo da nossa descida de gradiente se aproxime mais do mínimo local. Para extrairmos a direção que temos que ir, utilizamos a derivada da função de custo:

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x^i \tag{3}$$

Logo, para aproximarmos os valores de θ de modo a nos aproximar do mínimo local, utilizamos a fórmula

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x^i \tag{4}$$

, onde $0 \leq j \leq m$, $x_0 = 1$ e α é o que chamamos de *Learning Rate* que define o quão agressivamente tentaremos nos aproximar do mínimo. Este método de descida de gradiente é chamada de *Batch Gradient Descent*. A seguir, veremos três outras variações deste algoritmo

a) Stochastic Gradient Descent: Neste método, utiliza-se apenas um exemplo de treino para cada passo da descida de gradiente. Deste modo, a equação 4 se transforma em

$$\theta_j := \theta_j - \alpha (h_\theta(x^i) - y^i) x^i \tag{5}$$

Utiliza-se este método quando procura-se rapidez de execução. Contudo, um ponto negativo deste método é que, como usamos como amostra apenas um exemplo de treino, um passo pode nos levar a um custo mais alto. Como o número de iterações é alta, porém, o método converge ao mínimo e mais eficientemente do que o *Batch* até um certo limite.

b) Mini Batch Gradient Descent: Para obtermos um resultado balanceado, utiliza-se uma mistura dos métodos *Batch* e *Stochastic*, em que define-se um tamanho para o lote de exemplos de treino que serão utilizados para atualizar cara θ.

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=k}^{k+b-1} (h_\theta(x^i) - y^i) x^i \tag{6}$$

, onde b é o tamanho do lote, $k = 0, b, 2b, ..., m-1$.

c) Equação Normal: Para a regressão linear, é possível derivar uma fórmula direta para o ponto de mínimo local que desejamos. Para isso, utilizamos manipulações matriciais na forma

$$\theta = (X^T X)^{-1} X^T y \tag{7}$$

, onde X é a matriz de features, y é a matriz dos dados que queremos prever e θ é a matriz dos coeficientes de $h(X)$.

III. Proposed Solutions

Normalização de Features

Para entender melhor como cada feature influencia o preço de um diamante, plotamos nove gráficos de preço x valor da feature:

Analisando esses gráficos, podemos perceber que algumas features parecem ter maior influencia sobre o preço do que outras. Também podemos perceber que algumas features parecem ter influencia mais próxima de uma função quadrática do que linear sobre o preço do diamante. Na seção de experimentos, mostraremos o que foi feito com relação à essas diferenças.

Já que alguns valores das features encontram-se na casa das unidades e outros das centenas, decidimos fazer a Normalização das features para garantir resultados o amis precisos posssźveis. Tal normalização foi feita da seguinte forma:

Regressão Linear:

Para a nossa primeira abordagem do problema, utilizamos uma regressão linear da forma: (FORMULA BASICA DA REGRESSAO LINEAR). Para descidir que Learning Rate utilizar e quantas iterações deveríamos ter, calculamos o erro ao longo da execução do algoritmo pela fórmula:

IV. Experiments and Discussion

Com o intuito de analisar o impacto de difentes algoritmos e inputs nas respostas finais, fizemos diversos experimentos. Foram testados os seguintes algoritmos: Gradient Descent nas versões Batch, Mini Batch e Stochastic Normal Equation Além disso, testamos os inputs: Learning Rates Parada por Numero de iterações e por Convergência

V. Conclusions and Future Work

The main conclusions of the work as well as some future directions for other people interested in continuing this work. [1]

References

[1] C. M. Bishop, “Pattern recognition and machine learning,” 2006.