Entendendo Regressão Linear Big Data no Hadoop

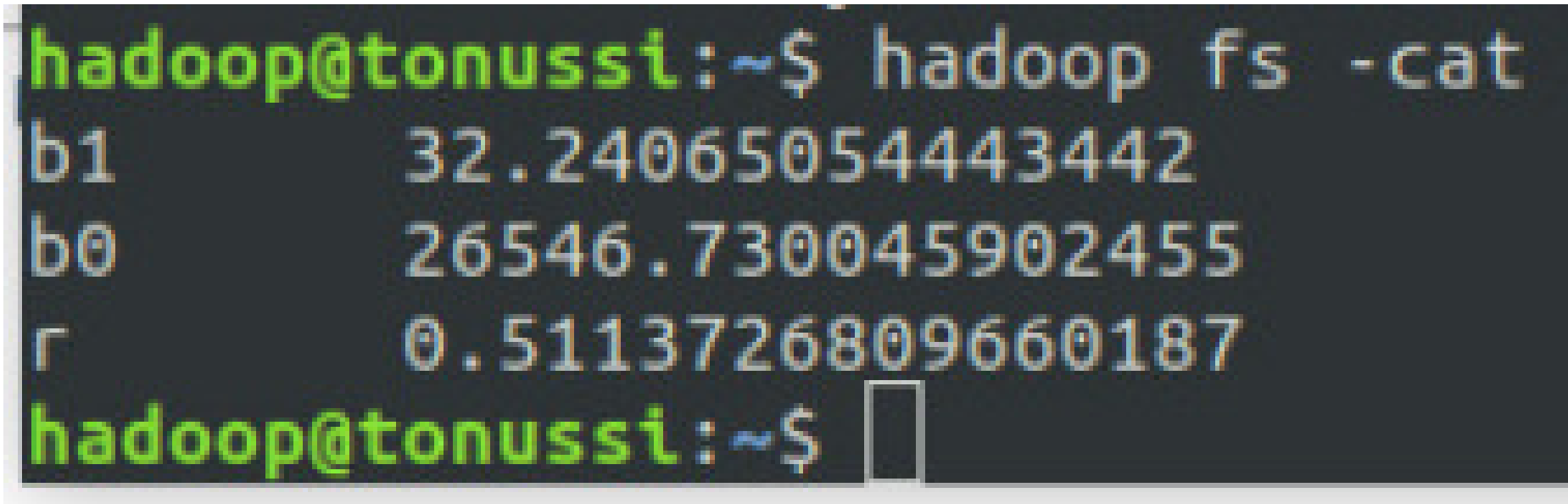
Lucas P. Tonussi

Universidade Federal de Santa Catarina
INE5645-05238A (20192) - Programação Paralela e Distribuída
Orientador: Prof. Dr. Odorico Machado Mendizabal

4 de Dezembro de 2019

O que é Regressão Linear Simples?

Regressão Linear é uma técnica estatística para entender qual a tendência dos dados quando criamos um relacionamento entre uma variável explicativa (X) e uma variável resposta (Y). A regressão linear nos ajuda a criar uma reta $Y_i = \beta_0 + \beta_1 X_i + \epsilon$. Com essa reta podemos fazer algumas predições. Teste de Correlação de Pearson (r) é um teste usado para saber o grau de correlação entre duas variáveis.



Como RLS pode nos ajuda em Big Data (Motivação)?

Quando temos um volume muito grande de dados, distribuído e sem estruturação, vamos primeiro querer dar sentido e organizar os dados e depois tentar entender se existe correlação entre eles. Uma boa maneira é usar Teste de Correlação de Pearson (r). E depois fazer a regressão linear para fazer algumas previsões de valores futuros. A Motivação por trás é: Imagine que quando temos Big Data Distribuída a gente não sabe como 2 colunas de dados estão se correlacionando, então por isso podemos usar o Hadoop para nos auxiliar com esse problema.

O que é Big Data?

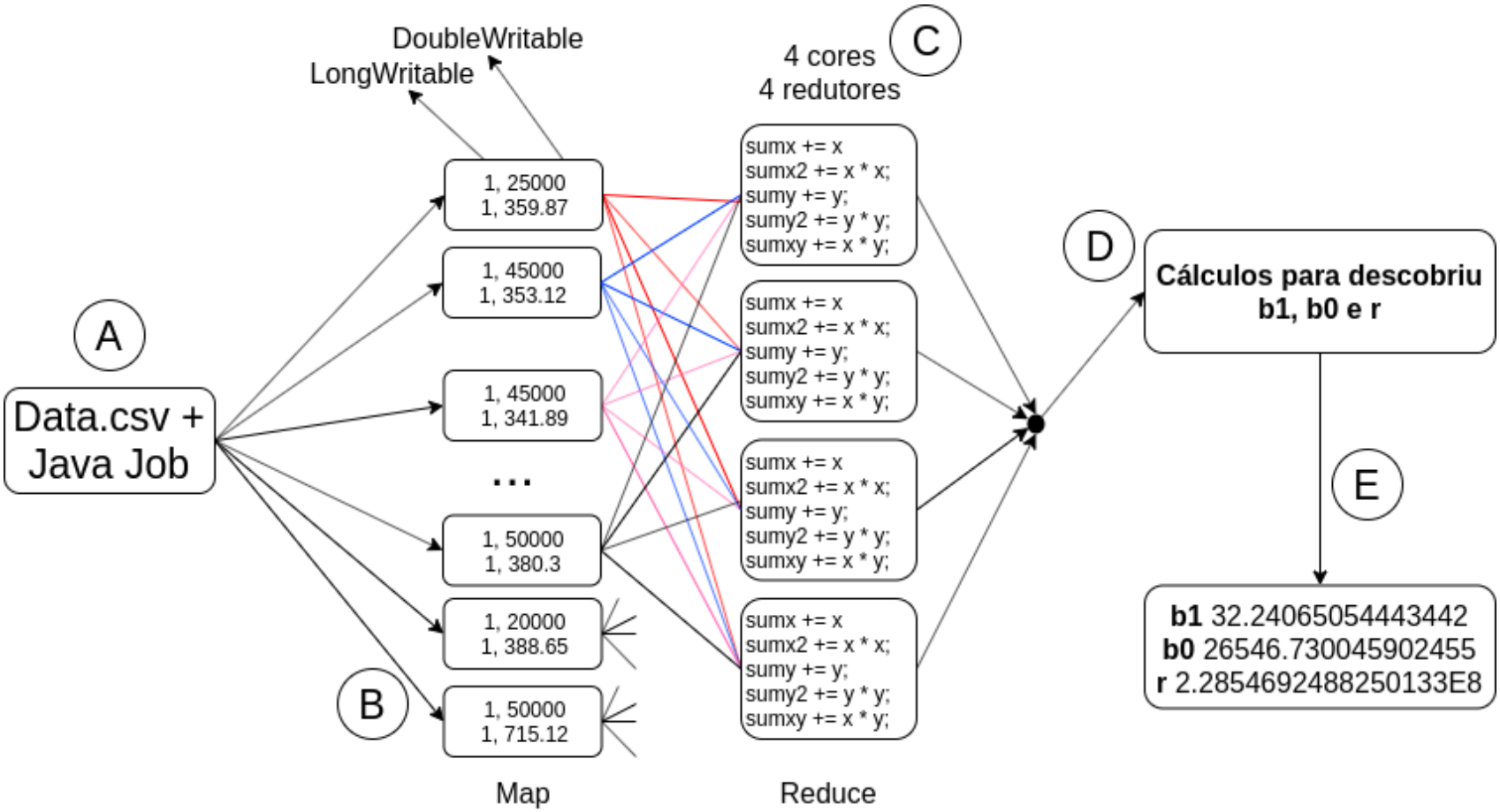
Big data é uma combinação de fatores e também um problema a ser resolvido. Big data se trata primeiro de um grande volume de dados, {com, semi, sem}-estruturação, dispersos na rede. E apresenta um problema grande pois ferramentas e algoritmos convencionais não tratam em tempo polinomial, o Big data por causa do tamanho. Então uma solução é distribuir o processamento usando Hadoop.

O que precisamos?

Os próximos slides estão preparados para apresentar o código usado para encontrar a regressão linear e o coeficiente de pearson. Ou seja, para acharmos a RLS precisamos:

1. Identificar no Big Data qual será nossa variável Y
2. Identificar no Big Data qual será nossa variável X
3. Encontrar o Intercepto (β_0)
4. Encontrar o Coeficiente Angular (β_1)
5. Construir a Equação da Reta.
6. Calcular o coeficiente de Pearson (r)

As fórmulas foram retiradas dos slides do professor Nakamura (INE5649-03238 (20192) - Técnicas Estatísticas de Predição).


DoubleWritable
LongWritable
4 cores
4 redutores
C
A
Data.csv +
Java Job
1, 25000
1, 359.87
1, 45000
1, 353.12
1, 45000
1, 341.89
...
1, 50000
1, 380.3
1, 20000
1, 388.65
1, 50000
1, 715.12
B
Map
sumx += x
sumx2 += x * x;
sumy += y;
sumy2 += y * y;
sumxy += x * y;
sumx += x
sumx2 += x * x;
sumy += y;
sumy2 += y * y;
sumxy += x * y;
sumx += x
sumx2 += x * x;
sumy += y;
sumy2 += y * y;
sumxy += x * y;
sumx += x
sumx2 += x * x;
sumy += y;
sumy2 += y * y;
sumxy += x * y;
Reduce
D
Cálculos para descobriu
b1, b0 e r
E
b1 32.24065054443442
b0 26546.730045902455
r 2.2854692488250133E8

O código Java (Job)

As linhas de código a seguir servem para configurar um Job simples, bastante genérico, para o Hadoop. E receber o caminho para os dados a serem usados. Perfazem o label A na figura [6]. E para executar o jar no Hadoop precisa ter Apache-{Hadoop, HDFS, YARN} instalado e executando normalmente, e então basta executar a linha de comando abaixo:

```
hadoop jar java/rlm-1.0-SNAPSHOT.jar
    br.ufsc.rlm.LinearRegressionDriver
    /analytics/geo.csv
    /analyticsout/geo56
```

```
JobClient jobClient
= new JobClient();

JobConf jobConfiguration =
new JobConf(LinearRegressionDriver.class);

jobConfiguration
.setJobName("LinearRegressionGeoData");

jobConfiguration
.setOutputKeyClass(LongWritable.class);

jobConfiguration
.setOutputValueClass(DoubleWritable.class);

jobConfiguration
.setMapperClass(LinearRegressionMapper.class);
```

```
jobConfiguration
.setReducerClass(LinearRegressionReducer.class);

jobConfiguration
.setInputFormat(TextInputFormat.class);

jobConfiguration
.setOutputFormat(TextOutputFormat.class);

FileInputFormat
.setInputPaths(jobConfiguration,
new Path(args[0]));

FileOutputFormat
.setOutputPath(jobConfiguration,
new Path(args[1]));
```

```
jobClient.setConf(jobConfiguration);
JobClient.runJob(jobConfiguration);
```

O código Java (Mapper)

O Mapper entrelaça os valores da coluna Y com os da coluna X, isso não é feito de maneira bruta pelo Apache-{HDFS} os mapeamentos são apontamentos estratégicos para que seja carregado na memória a medida que os recursos permitem, sem afogar o sistema. A chave 1 em todos os valores entrelaçados indica que os Redutores deverão ao final juntar esses mesmos dados pois eles pertencem à esse mesmo mapeamento. O mapeamento se trata da label B na figura [6] Exemplo:

Y	X	Entrelaçados
1	4	1: 1
2	5	1: 4
3	6	1: 2
		1: 5
		1: 3
		1: 6

```
LongWritable one = new LongWritable(1);

@Override
public void map(LongWritable key, Text value,
        OutputCollector<LongWritable,
                DoubleWritable> output,
        Reporter reporter) throws IOException {

    String line = value.toString();
    String[] features = line.split(",");

    output.collect(one, new DoubleWritable(
        new Double(features[1]))); // <- y

    output.collect(one, new DoubleWritable(
        new Double(features[2]))); // <- x
}
```

O código Java (Reducer)

A redução faz cálculos de somatórios usando os dados dos mapeamentos. Somatório de x, y, x^2, y^2, xy. E depois desses somatórios são feitos outros cálculos, imagino que o *Hadoop* cria *threads* de processamento paralelo para esses cálculos, incluindo os somatórios. Imagino também que o hadoop procura criar redutores na medida certa que o servidor aquenta e com um número de redutores em que o *server* irá se desempenhar melhor. A redução se trata das fases C, D, E. Na fase E os redutores escrever os resultados em um arquivo que fica disponível no sistema HDFS para análise posterior.

```
public class LinearRegressionReducer
  extends MapReduceBase
  implements Reducer<LongWritable,
  DoubleWritable, Text, DoubleWritable> {

    @Override
    public void reduce(LongWritable key,
            Iterator<DoubleWritable> values,
            OutputCollector<Text,
            DoubleWritable> output,
            Reporter reporter)
            throws IOException {

     int      n = 0;       double x = 0.0;
     double   y = 0.0;     double sumx = 0.0;
     double   sumy = 0.0;  double sumx2 = 0.0;
     double   sumy2 = 0.0; double sumxy = 0.0;
```

```
while (values.hasNext()) {
    x = values.next().get();
    sumx += x;
    sumx2 += x * x;

    y = values.next().get();
    sumy += y;
    sumy2 += y * y;
    sumxy += x * y;
    n++;
}
```

```
double x_bar = sumx / n;

double y_bar = sumy / n;

double b1_upper_part = sumxy
        - (n * x_bar * y_bar);

double b1_lower_part = sumx2
        - (n * x_bar * x_bar);

double b1 = b1_upper_part
          / b1_lower_part;
```

```
double b0 = y_bar
          - (b1 * x_bar);

double r = b1_upper_part
         / (Math.sqrt(sumx2
         - (n * x_bar * x_bar))
         * Math.sqrt((sumy2)
         - (n * y_bar * y_bar)));
```

```
output.collect(new Text("b1"),
new DoubleWritable(b1));

output.collect(new Text("b0"),
new DoubleWritable(b0));

output.collect(new Text("r"),
new DoubleWritable(r));
```

Testes com dados de geoprocessamento

Os dados que foram utilizados para testar a RLS no Hadoop foram dados provenientes do site kaggle.com/lptonussi/raquel. Accesso em 4 de Dezembro de 2019. Esses dados são da pesquisa de Rocha (2016).

Dados e Referências

ROCHA, R. R. Técnicas de geoprocessamento aplicadas à avaliação de imóveis. estudo de caso: Região central de ibirité. 2005. 44 f. monografia (especialização). Curso de Geoprocessamento, Cartografia, Instituto de Geociências, Universidade Federal de Minas Gerais, Belo Horizonte, 2005. Disponível em: link-pdf. Acesso em 4 de Dezembro de 2019.

Tomando as tabelas disponíveis em kaggle.com/lptonussi/raquel como referência, é importante mostrar como a autora fez para mapear esses dados:

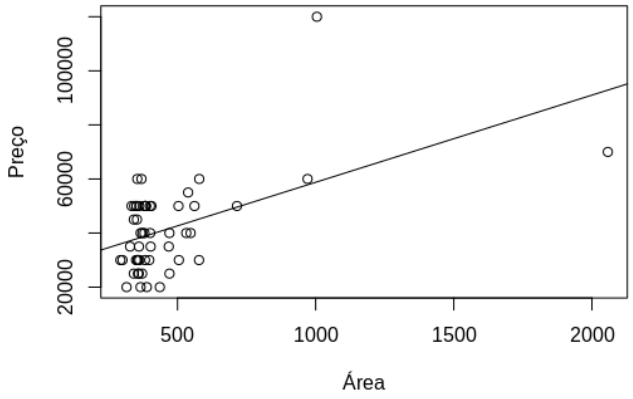
1. Preço do imóvel avaliado em reais (label PRECO).
2. Coeficiente de aproveitamento do imóvel sobre o lote (label COEFAP).
3. Área do lote onde o imóvel se localiza (label AREA).
4. Sentido predominante da topografia por lote (label ACLDECL).
5. Frente do lote (label FRENTE).

Resultados - Equações

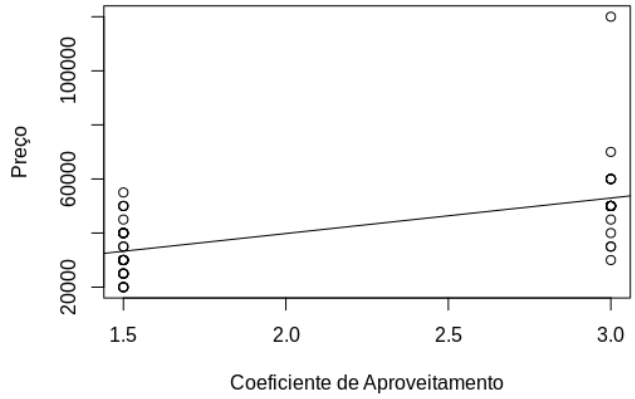
Os resultados para esse trabalho se tratam das equações que correlacionam Preço com cada uma das outras variáveis. Para termos algumas noções de como os dados estão se comportando.

1. $\hat{Y}_{preco} = 26546.7300 + 32.2406 X_{area}$
2. $\hat{Y}_{preco} = 49512.6897 - 28.9045 X_{dist}$
3. $\hat{Y}_{preco} = 13607.9545 + 13115.5303 X_{coefap}$
4. $\hat{Y}_{preco} = -33474.3287 + 80350.5224 X_{acldecl}$
5. $\hat{Y}_{preco} = 30629.0752 + 395.8859 X_{frente}$

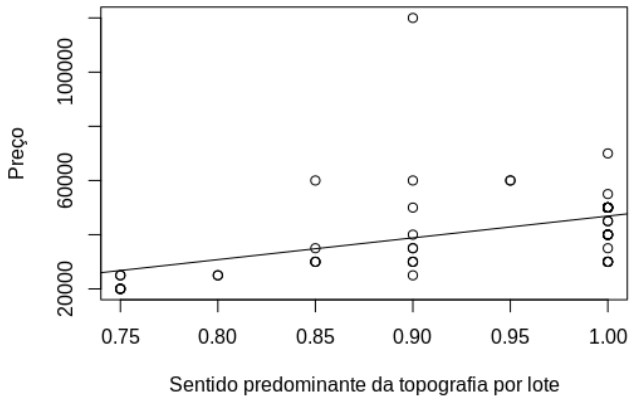
Com essas equações podemos calcular erros em relação aos preços originais. Podemos fazer algumas previsões para outros valores de X diferentes.
Entre 1090-1580 ms de tempo de CPU.



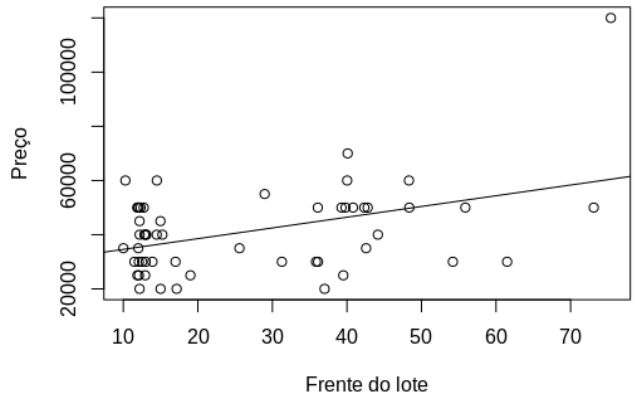
$r = 0.5113$



$r = 0.5917$



$r = 0.4296$



$r = 0.4218$

Figura: Diagramas de Dispersão com o r de Pearson calculado

Obrigado