PART 1.
LARGE SCALE DATA ANALYSIS USING MAPREDUCE

Instructor: Sangmi Lee Pallickara
Computer Science, Colorado State University
http://www.cs.colostate.edu/~cs435

Today’s topics

• FAQs
• Predictive Analysis
  • Linear Regression using MapReduce
• Midterm 1

FAQs

• PA2 dataset is available
• GTA will announce the instruction.
• Revised Deadline: March 25th (Friday) 5:00PM via Canvas
  • Firm deadline
• Help session for PA2 has been posted
  • Please check the assignment page
• PA3 is using AWS

Example topics from previous courses

• Similarity and Clustering on The Million Song Dataset
• Methods Used to Analyze Eclipse and Mozilla Bug Data
• Movie Recommendation using Collaborative Filtering
• Climate Visualization and Predictive Analysis
• Wikipedia Page Traffic Statistics Analysis
  Trending Topics and Page Count Prediction on Wikipedia Traffic Log Data
• Supporting Emergency Response During Natural Disasters with Twitter Data

Example topics from previous courses

• “Time to Answer” for Questions on stackoverflow.com using MapReduce
• Analysis of words for spell-checking in search queries using digitized books and articles
• Who is Building Wikipedia?

Components of the proposal documents

• 1,200-1,500 words
  • Do not exceed the limit

1. Title of your project
  • This should be concise and self-descriptive.

2. Problem formulation
  • The proposal should clearly identify the problem. It should include at least
    one or two carefully crafted paragraph that states and highlights the
    problem. The problem formulation should be able to answer following
    questions:
    • What is the problem you are solving?
    • This should also include the background for the problem.
    • Why is it interesting as a Big Data problem and who would use it if it were
      solved?
Components of the proposal documents

3. Your strategy to solve the problem
   - Describe your proposed approach to solve the problem. The description of the strategy should include,
     - The algorithms/techniques/models you plan to use in this project.
     - The framework you plan to use in this project.
     - Please note that you are also required to produce software as the final output of this project.

4. Functions targeted by your software
   - Your proposal should include a software design to provide a more specific view of your project. A simple description of major functions should be enough for this section. As your development proceeds, this may be updated.
     - What functions does your software provide to your users?
     - What will be the input and output of each function?

5. Plan for testing
   - Your software should be tested before you provide the final results and presentation.
     - What is your plan for testing your software?
     - What will be your test data?
     - What will be your testing scenario?
     - How will you collect your test data?
     - How will you deploy your software?

6. Evaluation method
   - The proposal should include an evaluation plan including metrics that you will use to identify if you have succeeded or not.
     - If you come up with a metric, also provide an intuitive feel for what this metric captures and why you think this is appropriate.

7. Bibliography
   - Included a bibliography. All references must be cited in the report.

8. Project timeline
   - You should provide a table with a weekly plan to complete the term project.
   - If you have teammate, the plan should also include information about the respective roles.

9. Submission
   - Via canvas, single copy per team

Predictive Analysis
Fitting Linear Regression Model to a Large Dataset

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math score</td>
<td>85</td>
<td>85</td>
<td>62</td>
<td>67</td>
<td>89</td>
<td>65</td>
<td>80</td>
<td>67</td>
</tr>
<tr>
<td>Physics score</td>
<td>89</td>
<td>92</td>
<td>70</td>
<td>95</td>
<td>95</td>
<td>80</td>
<td>75</td>
<td>80</td>
</tr>
</tbody>
</table>

- Is there any correlation between a student’s Math score and Physics score?
- If a student’s Math score is known, can we predict his/her Physics score?
Linear regression

\[ h_\theta(x) = \theta_0 + \theta_1 x_1 \]

where, the \( \theta \)s are the parameter vectors.

\( h_\theta(x) \), (Estimated Physics score) is called the regressand or endogeneous variable.

\( x_1 \), (Math score) is called regressor, or exogeneous variable.

Fitting the linear regression model

- How big is the error of the fitted model?
  - We would like to minimize this error.

- The model that fits the data best
  - The model with the minimum sum of errors on the training data.
  - e.g. The sum or mean of the squares of the errors.
  - Least squares regression.

Squared error

- Squared error
  - Strongly penalizes very large errors.
  - Drawback: Is very sensitive to the data.
  - Erroneous or outlying data points can severely skew the resultant linear function.

- We should choose the objective function to optimize.

Root Mean Squared Error

- Measures the differences between values predicted by model estimator and the values actually observed.

\[ \text{RMSD} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (h_\theta(x_i) - y_i)^2} \]

Linear Regression

\[ h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots \]

To simplify the notation,

\[ h(x) = \sum_{i=0}^{n} \theta_i x_i \]
Objective function (Cost function)

- For a given training set, how do we pick, or learn, the parameter \( \theta \)?
- Make \( h(x) \) close to \( y \)
- Make your prediction close to the real observation
- We define the objective (cost) function
  - Using Mean Squared Error and multiplying \( \frac{1}{2} \) for convenience
  \[
  J(\theta) = \frac{1}{2m} \sum_{i=0}^{m} (h_\theta(x^i) - y^i)^2
  \]

Minimization problem

- We have a function \( f(\theta_0, \theta_1) \)
- We want to find \( \min_{\theta_0, \theta_1} f(\theta_0, \theta_1) \)
- Goal: Find parameters to minimize the cost (output of the objective function)
- Outline of our approach:
  - Start with some \( \theta_0, \theta_1 \)
  - Keep changing \( \theta_0, \theta_1 \) to reduce \( J(\theta_0, \theta_1) \) until we end up at a minimum

Concept of Gradient descent algorithm (1/2)

Concept of Gradient descent algorithm (2/2)

Stochastic Gradient descent algorithm

Repeat until convergence {
  \[
  \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)
  \]
  (for \( j=0 \) and \( j=1 \))
}

- Simultaneous update
  - Your implementation should perform simultaneous update
  - See pp.53

Simultaneous update

<table>
<thead>
<tr>
<th>Correct: Simultaneous update</th>
<th>Incorrect:</th>
</tr>
</thead>
<tbody>
<tr>
<td>temp0 := ( \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) )</td>
<td>temp0 := ( \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) )</td>
</tr>
<tr>
<td>temp1 := ( \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) )</td>
<td>temp1 := ( \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) )</td>
</tr>
<tr>
<td>( \theta_0 := \text{temp0} )</td>
<td>( \theta_0 := \text{temp0} )</td>
</tr>
<tr>
<td>( \theta_1 := \text{temp1} )</td>
<td>( \theta_1 := \text{temp1} )</td>
</tr>
</tbody>
</table>
Decreasing $\theta_j$
- Positive slope
  $$\theta_j := \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_j)$$

Increasing $\theta_j$
- Negative slope
  $$\theta_j := \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_j)$$

Learning rates
- If $\alpha$ is too small, gradient descent can be slow
- If $\alpha$ is too large, gradient descent can overshoot the minimum
  - It may fail to converge
  - Or even diverge

Learning rates: starting at the optimal value?
- Current value of $\theta_j$ has converged already
  - No more iteration required

Fixed learning rate $\alpha$
- Gradient descent can converge to a local minimum, even with a fixed learning rate
  - As we approach a local minimum, gradient descent will automatically take smaller steps
    - No need to decrease $\alpha$ over time
  $$\theta_j := \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_j)$$

Using Gradient Descent Algorithm for Linear Regression Model
Gradient descent algorithm
Repeat until convergence
$$\begin{align*}
\theta_j &:= \theta_j - \alpha \frac{d}{d\theta_j} J(\theta_j) \\
\text{(for } j=0 \text{ and } j=1)\end{align*}$$

Linear Regression Model
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right)^2$$
Gradient descent for Linear Regression

Repeat until convergence:

\[
\begin{align*}
\theta_0 &\leftarrow \theta_0 - \alpha \frac{1}{m} \sum (h_\theta(x) - y) \\
\theta_1 &\leftarrow \theta_1 - \alpha \frac{1}{m} \sum (h_\theta(x) - y)x
\end{align*}
\]

Case 1, \( \theta_0 \) (\( j = 0 \)):

\[
\theta_0 = \theta_0 - \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)
\]

Case 2, \( \theta_1 \) (\( j = 1 \)):

\[
\theta_1 = \theta_1 - \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)
\]

Multiple local optimal points

Convex function

Fitting \( h_\theta(x) \)

Bowl-shaped function

h_\theta(x) and \( J(\theta_0, \theta_1) \)
### "Batch" Gradient Descent

- **Batch**
  - Each step of gradient descent uses all of the training example

\[
\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})
\]

\[
\theta_i = \theta_i - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_i(x^{(i)}) - y^{(i)}) x^{(i)}
\]

### Running with MapReduce

- **For the sample size 1,000 (m=1,000)**

- **Batch gradient descent**:

\[
\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})
\]

\[
\theta_i = \theta_i - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_i(x^{(i)}) - y^{(i)}) x^{(i)}
\]

- **Using 4 machines**

#### For \(\theta_0\)

- **Step 1. 4 input splits**

\[
\begin{align*}
\text{temp1} &= \sum_{i=1}^{250} (h_0(x^{(i)}) - y^{(i)}) \\
\text{temp2} &= \sum_{i=251}^{500} (h_0(x^{(i)}) - y^{(i)}) \\
\text{temp3} &= \sum_{i=501}^{750} (h_0(x^{(i)}) - y^{(i)}) \\
\text{temp4} &= \sum_{i=751}^{1000} (h_0(x^{(i)}) - y^{(i)})
\end{align*}
\]

- **Step 2. Calculate temp1 ~ 4**

- **Step 3. Calculate final results**

\[
\theta_0 = \theta_0 - \alpha \frac{1}{1,000} (\text{temp1} + \text{temp2} + \text{temp3} + \text{temp4})
\]

#### For \(\theta_1\)

- **Step 1. 4 input splits**

\[
\begin{align*}
\text{temp1} &= \sum_{i=1}^{250} (h_1(x^{(i)}) - y^{(i)}) x^{(i)} \\
\text{temp2} &= \sum_{i=251}^{500} (h_1(x^{(i)}) - y^{(i)}) x^{(i)} \\
\text{temp3} &= \sum_{i=501}^{750} (h_1(x^{(i)}) - y^{(i)}) x^{(i)} \\
\text{temp4} &= \sum_{i=751}^{1000} (h_1(x^{(i)}) - y^{(i)}) x^{(i)}
\end{align*}
\]

- **Step 2. Calculate temp1 ~ 4**

- **Step 3. Calculate final results**

\[
\theta_1 = \theta_1 - \alpha \frac{1}{1,000} (\text{temp1} + \text{temp2} + \text{temp3} + \text{temp4})
\]