VE472 — Methods and tools for big data

Homework 6

Manuel — UM-JI (Summer 2021)

Reminders

- Write in a neat and legible handwriting or use LaTeX
- Clearly explain the reasoning process
- Write in a complete style (subject, verb, and object)
- Be critical on your results

In this homework we want to investigate Gradient Descent, a data analytics method which plays a very important role in the field of machine learning.

Ex. 1 — *Discovering gradient descent*

Since Reapor has started his new diet is pays great attention to everything he eats. To make fun of him Frank has collected information on various cereal brands that Reapor has been eating over the past few weeks. He obtained the following table.

Protein (g)	70	120	50	60	1	100	90	130	105	108
Sugar (g)	6	8	5	1	2	10	14	14	9	10

1. Draw a graph showing Protein and Sugar on the x and y axes, respectively. Explain what it means to "find the best least squares line fit". Show it on your graph.
2. Basic gradient descent.
   a) Standardize the all the values from the table.
   b) Assume the best least square line has equation $y_p = a + bx$, where y_p is the predicted y for a protein weight x and a and b are in $[0, 1]$. Then compute y_p using the standardized x and some randomly selected a and b. Determine the Sum of Squared Errors (SSE).

Now that frank has a starting point he would like to improve on the quality of his line with respect to SSE. Ideally SSE should be as close to zero as possible. The feeling he has is that his SSE is a bit high, then he can make it decrease by "adjusting" his line. However "after a while" the error will grow again.

To explain the idea to Krystor, Frank asks him to imagine that b is fixed and he can only adjust a. His first line will be, lets say, "below the points", however as he "moves it up" by drawing new parallel lines it will get closer and closer from the points, hence decreasing the SSE. However at some stage the line will be "too high" and the SSE will start to increase again.

It is now clear to both Krystor and Frank that the goal is to minimize SSE, i.e. the function

$$\sum_{i=1}^{10}(y_i - y_{i,p})^2 = \sum_{i=1}^{10}(y_i - a - bx_i)^2.$$

3. Gradient descent algorithm.
   a) Explain the relationship between *gradient descent* and what Frank and Krystor understood.
   Hint: consider $\partial\text{SSE}/\partial a$ and $\partial\text{SSE}/\partial b$, where $\text{SSE} = \sum_{i=1}^{10}(y_i - a - bx_i)^2$.
   b) Describe a simple iterative algorithm performing gradient descent and apply it to the cereal data collected by Frank.

4. Briefly explain and run (i) stochastic gradient descent and (ii) batch gradient descent on the cereal dataset.

Gradient descent

From a more general point of view gradient descent is an optimization problem

$$\min_w F(w) = \sum_{i=1}^{m} F_i(w, x_i, w_i),$$

where $x_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$ is a "label" and w is what we want to optimize over. In essence gradient descent starts with a random initial w and has the goal of iteratively improving on it by computing $w_{k+1} = w_k + \alpha \nabla F(w_k)$ where α is small.[a] The choice of the objective function F corresponds to the loss function of interest for the problem.

[a] In fact from a theoretical point of view, this works well as soon as F is differentiable, and ∇F is L-Lipschitz continuous, i.e. for any $L > 0$, $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$, for all x, y in the domain of F.

PRAM model

Recalling our PRAM model briefly described in chapter 1 (1.9|1.28), we introduce the notions of

- *Work:* the number of processors multiplied by the amount of time used to complete the computation; this can be viewed as the cost of the algorithm is time needs to be bought on a super computer;
- *Depth:* dependencies on operations in a parallel algorithm is represented using a Directed Acyclic Graph (DAG) where each computation is a node and the dependencies between operations are edges[a]. By construction only operations at a same level in the tree can be performed concurrently, i.e. in parallel. However for a node the computation cannot start until all its children have completed, meaning that the deeper the tree the longer the computation.

[a] A DAG is clearly a connected tree since no cycle should appear in it and all operations are useful to obtain the final result. The root corresponds to the output and the children the steps leading to the it.

Ex. 2 — *Gradient descent on big data*

In this exercise we want to better evaluate the difficulty of gradient descent from a computational point of view and see how it could be implemented in Spark.

Based on the boxes *Gradient descent* and *PRAM model* we want to investigate how gradient descent methods scale with respect to m (data parallelism) and n (model parallelism). We restrict our attention to least square gradient descent or in other words

$$F(w) = \sum_{i=1}^{m} F_i(w, x_i, w_i) = \sum_{i=1}^{m} \|x_i^\top w - y_i\|_2^2,$$

1. Gradient descent.

   a) Determine the work and depth for $F(w)$.

b) Calculate $\nabla_w F$ and determine its work and depth.

Hint: assume that to achieve error $\frac{1}{\varepsilon}$, $\mathcal{O}\left(\log \frac{1}{\varepsilon}\right)$ iterations are necessary.

c) Conclude on the suitability of gradient descent for parallelization and big data.

2. Stochastic gradient descent.

a) Determine work and depth for stochastic gradient descent.

Hint: since not all the data is used at each iteration convergence happens at the slower rate where error ε is achieved only after $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ iterations.

b) Being in a multiprocessor setup, what is the problem if w is in a piece f the memory shared by all cores?

c) Would the use of locks solve the problem? It is a viable solution?

d) Assuming a large m and n and that each F_i only acts upon a small number of components of w, are locks necessary? Explain the feeling behind your answer.

3. Gradient descent in Spark.

a) What would be the best way to store the data in the cluster? For instance should it be stored by row, columns, elements, or blocks?

b) Assuming each node computes a ∇F_i, Describe how to sum them up across the cluster.

c) Discuss the communication cost.

4. Stochastic gradient descent in Spark.

a) Knowing that Spark makes heavy use of barriers, how suitable is the idea developed in 2.d?[1]

b) Based on what was discussed in the lectures at the beginning of the semester, what would be potential software alternatives to perform stochastic gradient descent?

Ex. 3 — *PCA and gradient descent*

Gradient descent being a relatively expensive algorithm to run it becomes impractical when dealing with big data. A simple strategy is to extract the principal components of the big dataset and then work on the resulting approximation. However there exists another very interesting connection between PCA and gradient descent.

As explained in the lectures the purpose of PCA is to find the axes that will maximize the variance, or said otherwise our goal is to find which the directions for which the expectation of $XX^\top$ is maximized. From a high level perspective explain how gradient descent could be applied to obtain the principal components.

Ex. 4 — *Course evaluation*

Please fill in the course evaluation survey and get a bonus on your homework grade.

[1] The strategy consisting in skipping the locks is know as going *Hogwild!*.

Gradient descent and the simplex method

In ve477 a lab discusses Linear programming which states as follows.

> Given a mathematical model where an objective is represented as a linear function and some constraints are expressed as equalities or inequalities, maximize the objective while respecting the constraints.

Knowing that any maximization problem can be rephrased into a minimization one, and vice versa, is there any relationship between the simplex algorithm[a] and gradient descent?

In fact the simplex method is following a similar pattern as "gradient descent", although there is no direct notion of gradient involved in this algorithm. It should rather be viewed as a local search, where a pivot operation improves on the current basic solution. Obviously different choices of pivot can lead to different different "speeds". What allows the simplex method to converge to the maximum (or minimum) is that linear programming is convex, meaning that a local minimum is also a global minimum. As a result any choice of pivot leads to an optimal solution.
Although finding the "best pivot" to improve the objective function is possible, it is expensive in practice. But this theoretical point explains the parallel between gradient descent and simplex method, the "best pivot" being seen as the best "direction", i.e. the steepest direction to improve the objective.

[a] A very famous method to solve linear programming problems.