Minibatch P dc dp

Upwork project

Introduction

Instead of evaluating the entire dataset at once, we randomly sample a subset (called a mini-batch) **Xb** of size **b** from the dataset and run a PDC-DP-Means iteration on it, without updating the centers. We parallelize the processing 14 of **Xb** across the available cores. In each batch Xb we cache both the index and the distance of the most distances observation **xj** , and if that observation is at a distance of at least √λ from its nearest cluster, we instantiate a new cluster, centered at **xj** . Unlike in PDC-DP-Means, however, here we do not recalculate the cluster centers in each iteration, rather instead we take a step towards the observations assigned to the cluster, using the following (gradient-based) formula,

$$\boldsymbol{\mu}_k \leftarrow \left(1 - \frac{1}{n_k}\right) \boldsymbol{\mu}_k + \frac{1}{n_k} \boldsymbol{x}_j,$$

where **μk** is the current cluster center, **xj** is the new observation assigned to cluster **k**, and **nk** is the total number of observations assigned to cluster **k**, including **xj**

This algorithm also supports an online setting, where the main iteration is executed not on some sample from the dataset, but on the current available data. When new data arrives, we process it in the main iteration. Thus, there is no need to store the previously-seen data at any point in time.

Code Explanation

This template will guide you through adding content for each section. Use as is, or easily customize with your own images, fonts, and colors. This modern and versatile format includes space for all

the details you'd like to include. Simply replace this content with your own to start creating a more polished looking document.
Use the design as is, or easily customize with your own images, fonts, and colors. Quickly summarize complex information with bullet points, compelling data, or even interesting facts. This helps engage your readers without overloading them with information. Include quotes from notables or highlight positive reviews or testimonials. These testimonials should resonate emotionally with your reader and reinforce your values.

Do_in_parallel function

Input:

I. **Batch** – a subset points of the whole set selected at random, the batch is then split into p parts where p in the number of cpu processors available
II. **Num_items_in_part_p** – contains the length of part pi
III. **S_p** – contains the squared l2 norm of points in the part pi
IV. **I_max** – contains the point with max distance with the part pi
V. **d_max** – contains thedistance of point with max distance with the part pi

We take all these inputs and create p processes and run each process in parallel. Where each process handles a part of the batch and performs a series of computations.
After taking the inputs, the function updates the variable named "splitted labels" , this labels contains which point in part p belongs to which cluster. And then compares if the distance of each point is greater than d_max, if yes , then store the distance in d_max and update the respective variables.

Move_to_Xi function

This function works like a gradient descent optimization technique.

Input:

I. **Batch –** same explanation as above
II. **K** – total number of clusters formed till now

After taking the input, it creates a copy of the centroids from the previous iteration. Then for each point, it calculates which cluster it belongs to and performs gradient descent on each point and then recalculates the centroids and moves the cluster slowly towards the correct position.

Fit function

Input – X (the set of points observed till now)

Create a minibatch from X and perform parallel processing of dcdp means algorithm on the minibatch and use gradient descent optimization technique to recalculate new centers. Run till convergence