Mining Massive Data (SS2022)

Programming Assignment 2

Training SVMs with Stochastic Gradient Descent

Availability date: 15.4.2022
Submission due date: 16.5.2022
Peer-review due date: 23.5.2022

Number of tasks: 1
Maximum achievable points: 100

General Remarks

Please observe the following remarks:

- This is one of three programming assignments in *Mining Massive Data*. For each assignment you can earn up to 100 points. Details on grading are provided below.
- The deadline for this assignment is noted in the header. No deadline extensions will be granted.
- If you have questions or encounter problems, do not hesitate to contact us. See the contact details below.
- You have to solve this programming assignment in Python. Do not use any existing LSH library. You should implement LSH on your own.
- Pack your code, your report, and execution instructions into a single .zip file with the following name: `group(group number).zip` file (e.g., `group01.zip`) and upload it on Moodle.
- Only **one** team-member submits the zip file in the Moodle system.

Questions

Please ask questions primarily on the respective Moodle forum. If you don't get a response in reasonable time (48 hours) from us or your colleagues, get in touch with our tutors via mail (Lorenz Kummer, Kevin Sidak). If you don't get a response in reasonable time (48 hours) from the tutors, get in touch with the rest of us (Simon Rittel, Sebastian Tschiatschek).

Submission

Pack your code, your report, and execution instructions into a single .zip file with the following name: `group(group number).zip` file (e.g., `group01.zip`) and upload it on Moodle. Make sure that your code is executable, e.g., if you created a jupyter notebook, make sure that the notebook can be executed without error when you reset the kernel and execute the notebook from the beginning. Don't use any non-standard libraries in your code, e.g., only use numpy, scipy, etc.

Grading

You can achieve up to 100 points on this assignment. These 100 points are composed as follows:

- We grade your submission and award you up to 60 points.
- Peers grade your submission and award you up to 30 points.
- You get up to 10 points for peer-reviewing one of your colleagues' submissions.

The peer-review process will be managed on Moodle and you have 1 week for reviewing your colleagues' submissions. Guidelines for your peer-review and how you should grade your peers' works are available on Moodle. The points for peer-reviews are given to individual students and not to their respective programming groups.

🟩 Support Vector Machines using SGD

Goal. The goal of this assignment is to study different approaches for scaling-up supervised learning. In particular you will study Support Vector Machines (SVMs). You will (a) train SVMs using different variants of Stochastic Gradient Descent (SGD) and (b) train approximate kernelized SVMs using Random Fourier Features (RFFs). Regarding different variants of SGD algorithms you will study the impact on accuracy and runtime when using a parallel learning algorithm instead of a serial learning algorithm.

Data. In Table 1, we provide an overview about 3 different datasets which you will use to test your implementation. The smallest one, `toydata.csv`, is intended to develop the code, test your implementation, and visualize your initial results. The second one, `toydata_large.csv`, is intended to see some effects in runtime and quality when you compare the two approaches for parallel learning and serial learning. The third dataset, `MNIST` is intended to give you an intuition on what can be expected when working with real world datasets. All datasets are available on Moodle. `toydata.csv` and `toydata_large.csv` are provided as `.csv` files and you can load them for example using `pandas`. MNIST data is provided as `.npz` – you can load it using `data = numpy.load("mnist.npz")`. The features and labels of the training data can then be accessed using `data["train"]` and `data["train_labels"]`. Similarly, the features and labels of the test data can then be accessed using `data["test"]` and `data["test_labels"]`.

name	number of samples	dimensionality
`toydata.csv`	200	2
`toydata_large.csv`	200000	8
`MNIST`	60000 (train)	784

Table 1: Datasets for this assignment.

`MNIST` has 10 classes, hence you cannot directly use the hinge loss discussed in class. Instead, use the multi-class hinge loss

$$\ell(\mathbf{w}^0, \dots, \mathbf{w}^9; \mathbf{x}, y) = \max\{0, 1 + \max_{j \in \{0,\dots,9\}, j \neq y} \mathbf{w}^{j,T}\mathbf{x} - \mathbf{w}^{y,T}\mathbf{x}\}, \tag{1}$$

where $\mathbf{w}^0, \dots, \mathbf{w}^9$ are weights of the SVM for each of the 10 possible digits, $\mathbf{x}$ are the features of a sample and y is the corresponding label.

Performance estimation. For all datasets except MNIST, use 5-fold cross validation for computing performances, and use classification accuracy as measure of prediction performance. For MNIST, there is explicit training and test data, and you can split off a validation set of size 10000 from the training data.

Assignment

Implement a SVM (in the primal; hinge-loss + regularizer) and use SGD for optimization. You will consider the following two models/features: (1) A linear SVM using the original features of the data; and (b) a SVM using RFFs for approximating a radial basis function kernel. Both models will be trained using a *serial* and a *parallel* variant of SGD.

Linear SVM Model (34%). As a baseline model implement a linear SVM and train it in a non-parallel fashion (i.e., use standard SGD). Train a SVM for each of the datasets, select a suitable learning rate and regularization parameter.

☐ Report how you selected the learning rate and regularization parameter, and report the selected hyperparameters. Did you use different parameters for the different datasets?

☐ Include a plot illustrating convergence of SGD for your selected hyperparamters (this can be for a single fold, and should show the training error over the number of SGD epochs).

☐ Report the achieved classification accuracy and runtime for each of the datasets.

☐ Briefly discuss your implementation of the SVM and SGD.

Random Fourier Features (33%). Use RFFs to approximate a Laplacian kernel in a SVM [1]. Train SVMs with RFFs on each of the datasets, selecting a suitable learning rate and regularization parameter. Test at least 3 different numbers of RFFs (≥ 100 features).

☐ Report how you selected the learning rate and regularization parameter, and report the selected hyperparameters. Did you use different parameters for the different datasets?

☐ Include a plot illustrating convergence of SGD for your selected hyperparamters (this can be for a single fold, and should show the training error over the number of SGD epochs).

☐ Report the achieved classification accuracy and runtime for each of the datasets. Does the number of RFFs affect the classification accuracy? If so, how?

☐ Briefly discuss your implementation of RFFs.

☐ For MNIST, report the runtime and performance (in plots or a single plot) when training on 1000, 2000, and 3000 training samples, respectivel. Report the same when using sklearn's `svm.SVC` class. What do you observe?

Parallel Implementation (33%). There are parallel variants of SGD as discussed in the lecture. For this assignment, you have to implement the algorithm `SimuParallelSGD` in [2] (Algorithm 3). The algorithm has three parameters:

- *examples* (i.e., the data),
- the *learning-rate* (which needs to be determined experimentally),
- and the *number of machines*.

For solving this task you do not have to actually implement a parallel algorithm (you can perform all computation on your computer sequentially) but you are asked to report runtimes as if the algorithm was implemented in parallel (assuming that no additional time is needed for communication). Nevertheless, you can obtain **10 bonus points** if you actually implement a parallel variant of the algorithm, e.g., using Python threads[1] or any other threading library (in this case the number of machines corresponds to the number of threads you are using). Train a linear SVMs using the original features (i.e., do not use the RFFs).

☐ Report how you selected the learning rate and regularization parameter, and report the selected hyperparameters. Did you use different parameters for the different datasets?

☐ Report the achieved classification accuracy and runtime for each of the datasets. What is the impact of the number of machines on the runtime? How does the achieved accuracy compare to that obtained using standard SGD? Also, does the accuracy depend on the number of machines?

☐ Briefly discuss your parallel implementation of SGD. If you used threading, please provide information on the used threading library as well.

Report

Write a report about your work on this assignment, your findings and results. Make sure to report all the information indicated above.

[1] `https://www.tutorialspoint.com/python3/python_multithreading.htm`

References

[1] Rahimi, A., and Recht, B. Random features for large-scale kernel machines. *Advances in neural information processing systems 20* (2007).

[2] Zinkevich, M., Weimer, M., Smola, A. J., and Li, L. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada* (2010), J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds., Curran Associates, Inc., pp. 2595–2603.