Mining Massive Data (SS2024)

Programming Assignment 2

Large-scale SVM Training

Availability date: 08.04.2024
Submission due date: 06.05.2024
Peer-review due date: 13.05.2024

Number of tasks: 1
Maximum achievable points: 100 points (60+30+10 ... Submission / Review / Quality of review)

General Remarks

- This is one of 3 programming assignments. For each assignment you can earn up to 100 points. Details on grading are provided below.
- The deadline for this assignment is noted on the header. No deadline extensions will be granted. Upload your solution by the deadline on the course's Moodle page [3]. If you fail to upload your solution by the deadline you will not be awarded any points.
- If you have questions or encounter problems, do not hesitate to contact us.
- Solve this programming assignment in Python. Don't use any non-standard libraries in your code, e.g., only use `numpy`, `scipy`, `scikit-learn`, unless explicitly mentioned.
- 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. **If your code is not executable, we reserve the right to not award any points.**
- Solve the tasks on your own (considered group-wise). If we observe plagiarism or group work, we reserve the right to not award any points.
- 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.
- 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 tutor via mail (Megi Teneqexhi). 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, Timo Klein, Sebastian Tschiatschek).

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 (in terms of passes through the data) when using standard vanilla SGD and Adagrad.

Data. In Table 1, we provide an overview about the 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 reveal some effects on runtime and quality when you compare the two approaches for parallel learning and serial learning. The third dataset, `IMDB` [2] is intended to give you an intuition on what can be expected when working with real world datasets. The dataset is about movie ratings (positive or negative) based on a written review which we already converted for you into a bag-of-words representation (TFIDF). 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`. IMDB data is provided as `.npz` – you can load it using `data = numpy.load("imdb.npz", allow_pickle =True)`. 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 be accessed using `data["test"]` and `data["test_labels"]`. Please note that the features are stored in a sparse format[1].

Table 1: Datasets for this assignment.

name	number of samples	dimensionality
`toydata.csv`	200	2
`toydata_large.csv`	20,0000	8
`IMDB`	50,000 (total)	10,000

For your implementation, you can use the multi-class hinge loss (if you want):

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

where $\mathbf{w}^0, \ldots, \mathbf{w}^K$ are weights of the SVM for K different classes, $\mathbf{x}$ are the features of a sample and y is the corresponding label.

[1] `https://docs.scipy.org/doc/scipy/reference/sparse.html`

Performance estimation. For all datasets except IMDB, use 5-fold cross validation for computing performances, and use classification accuracy as measure of prediction performance. For IMDB, there is explicit training and test data, and you can split off a validation set of size 5,000 from the training data.

Assignment

Implement a SVM (in the primal; hinge-loss + regularizer) and use SGD/Adagrad 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 Gaussian kernel. Both models will be trained using a standard variant of SGD or Adagrad.

Linear SVM Model (34%). As a baseline model implement a linear SVM and train it in a non-parallel fashion (i.e., use standard mini-batch 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.

Optimization using Adagrad (33%). Implement Adagrad [1] using the diagonal approximation for optimization. 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 adagrad regarding the performance in comparison to standard SGD for the different datasets?

☐ Repeat your experiments using random dropout of input features with probability $p = 0.5$ for the IMDB data. Does this change affect performance and if so, how?

☐ Briefly discuss your implementation of Adagrad. How did you adjust it to perform stochastic optimization instead of online learning. Provide pseudo-code.

Random Fourier Features (33%). Use RFFs to approximate a Gaussian kernel in a SVM [4]. 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 ($\geq$ 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/Adagrad for your selected hyperparamters (this can be for a single fold, and should show the training error over the number of SGD/Adagrad 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 IMDB, report the runtime and performance (in plots or a single plot) when training on 1000, 2000, and 3000 training samples, respectively. Report the same when using sklearn's `svm.SVC` class. What do you observe?

Report

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

References

[1] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[2] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Ed. by Dekang Lin, Yuji Matsumoto, and Rada Mihalcea. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL: https://aclanthology.org/P11-1015.

[3] *Mining Massive Data Moodle Page*. URL: https://moodle.univie.ac.at/course/view.php?id=391343.

[4] Ali Rahimi and Benjamin Recht. "Random features for large-scale kernel machines". In: *Advances in neural information processing systems* 20 (2007).