Final Projects

CSE 569 Fall 2018

I. Outline

4 topics have been selected for the final course project. The project implementation will involve reading a research paper and implemnting part of the research and doing your analysis according to the guidelines. Some guidelines for analysis are provided, but the projects are open ended and the extent of analysis is left to the students. The topics are:

1) **Binary Neural Networks**

   This project will be mentored by TA: Kevin Ding. For this project you will implement a binary neural network along the lines of [1]. The article outlines a procedure to train a neural network with binary weights and binary activations. You need to implement both the deterministic and the stochastic binarization mechanisms for comparison. The article also discusses binary variants for BatchNorm and ADAM (momentum) which are not required to be implemented. You can use a vanilla BatchNorm and ADAM (momentum) implementation. You will compare your implementation with a regular (non-binary) neural network of the same architecture. You will use the fashion-MNIST dataset [2] for evaluation. You need to compare the train times, performance, memory usage, speed for the binary and non-binary versions. Additional resources for the project are [3] and [4].

2) **Topic Modeling**

   This project will be mentored by TA: Yuzhen Ding. It is based on one of the techniques in topic modeling. For this project, you will study the Corr-LDA model paper [5], implement inference process using Gibbs sampling instead of variational inference which is presented in the paper, and test on the Corel 5K dataset with caption perplexity as the measurement [6]. Note the statistics in the Corel dataset is a little bit different from what was used in [5]. In [5], the visual feature is region based. In the Corel dataset, the visual features are image based SIFT features which are encoded into size 1000. Also, the size of text words vocabulary is 260 rather than 168. More details about the Corel dataset:

   # of docs in train 4500

   # of docs in test 499

Text vocabulary 260

Visual vocabulary 1000

Since the number of topics K needs to be predefined, when calculating caption perplexity, you can try various K, and draw a figure similar to Fig. 5 in [5]. Since the dataset is different, the caption perplexity would also be different.

3) **Gradient Descent Optimization**

This project will be mentored by the instructor: Hemanth Venkateswara. For this project you will evaluate gradient optimization techniques for neural networks. You will implement a 3-layer fully-connected neural network (do not use any deep learning libraries like TensorFlow, Keras, MatConvNet etc. - `numpy` can be used) to classify the fashion-MNIST dataset [2]. You will test your gradient descent optimization implementations upon this network. The techniques you will implement are (i) No Momentum (regular network without momentum), (ii) Polyak's classical momentum [7], (iii) Nesterov's Accelerated Gradient [8], (iv) RmsProp [9] and (v) ADAM [10]. You will compare the stability, speeds and accuracies of the different momentum techniques. In the report for this project, paste the code snippets for the gradient updates (only the gradient update code using momentum - not the entire project code) after the References section. For additional resources please refer to [11].

4) **Autoencoders**

This project will be mentored by the instructor: Hemanth Venkateswara. For this project you will implement and evaluate autoencoder models. You will use the fashion-MNIST data to train and test your models [2]. You will implement this project on your own without using neural network libraries like TensorFlow, Keras, MatConvNet etc. - `numpy` can be used. This project has two parts.

(i) De-noising autoencoder. In this part, you will train an autoencoder with one hidden layer to reconstruct de-noised images from noisy versions [12], [13]. You will vary the noise level in the input and evaluate the de-noising capabilities of your network.

(ii) Stacked autoencoder [14]. For this part, you will train an autoencoder layer-by-layer in an unsupervised manner. Train 3 such layers - the input is not considered as a layer. You will stack the 3 layers to create a stacked autoencoder that is an unsupervised feature extractor. Add a classifier on top of the final layer and fine-tune the stacked autoencoder with labeled samples. For the fine-tuning, you will consider (i) 1-labeled sample per class, (ii) 5-labeled samples per class. You will evaluate the classification accuracies on the test dataset using these models. For fine-tuning the autoencoder, you need to lower the learning rates of the stacked layers or freeze them and only train the weights for the classification layer. If you have large learning rates for the stacked layers,

then the feature extraction that the network has learned will be unlearned and the weights will change to overfit the limited labeled data yielding poor test accuracies.

5) **Your Own Project**

If you already have a project related to statistical learning that you would like to work on, your group will need to reach out to the instructor and get approval to submit it as a course project. Your group will have to send one email to the instructor with a 1-page summary of your proposed work along with references and your proposed deliverables for the project. Your group then have to setup a meeting with the instructor and get your project proposal approved before Nov 1, 2018.

Students will form groups of 3 or 4 to implement any one of these projects. If your own research aligns with your project topic, please feel free to include results from your research in addition to the suggested analysis.

II. Deliverables

The course final project has 30% weight in the final score. There are three deliverables (Midterm report - 5%, Poster presentation - 5%, Final report - 20%):

1) **Midterm report - Due Nov 8th midnight**

Report consisting of preliminary work. Minimum 2-pages excluding references in IEEE Transactions Journal format with 2 columns. The report is submitted as a pdf file "(your student id)_(your last name)_midterm_project_report.pdf".

For example `1234567890_Hawking_midterm_project_report.pdf`.

Every student must submit the preliminary report. This preliminary report is meant to ensure that you have started working on the project. For this report, make sure you have the necessary section headings for all the sections like, Introduction, Related Work, Method, Experiments, Conclusions, Division of Work (see below for explanation), Self-Peer Evaluation Table (can leave the table empty for preliminary report) and References. The submission will be evaluated mainly for adhering to the format guidelines.

2) **Poster presentation - Due Nov 27th and Nov 29th in class**

For this deliverable your team will make a poster of 6 - 9 slides (e.g., powerpoint slides). We will split the projects into 2 groups and conduct poster presentations in the class on $11/27/2018$ and $11/29/2018$. You will paste the slides on the wall in a poster format and remove them at the end of the class. The instructor will provide sticky-tape or you can use your own tape. Please don't

damage the walls. Each team will have 2 minutes to present their work and 2 minutes to answer questions which the instructor will ask to any of the group members.

3) **Final submission - Due Nov 29th midnight**

   This has two components - final report and code. The final report is 4 pages long, excluding references, in IEEE transactions Journal format with 2 columns. The project will be mainly evaluated using this report. Make sure you have a well organized article with all the sections listed in the guidelines. The report is in the form of an IEEE journal article with sections like: Introduction, Related Work, Method, Experiments, Conclusions, Division of Work (see below for explanation), Self-Peer Evaluation Table and References. The report is submitted as a pdf file "(your student id)_(your last name)_final_project_report.pdf".

   For example `1234567890_Hawking_final_project_report.pdf`.

   The code is submitted as a zip file "(your student id)_(your last name)_final_project.zip".

   For example `1234567890_Hawking_final_project.zip`.

Note: The submitted code must be your own implementation and self-contained. You may rely on open source code for reference but do not submit open source code as it will be tested for plagiarism. Even though you may work in groups, every student must submit their own code. Your code may be similar to your team members's code.

Every student must submit the report. Students working in a group may have similar reports. However, make sure there is a unique section (**Division of Work**) that outlines your contributions and the contributions of your team members. The (**Self-Peer evaluation** is a table where you give points to yourself and your team members (out of 20). It should contain your team member's name and their score out of 20 based on their contribution. For example:

Paul Dirac - 15	Richard Feynman 19	Michael Faraday 20	Myself 18

TABLE I: Self-Peer Evaluation Table

REFERENCES

[1] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[2] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[3] J. Bethge, H. Yang, C. Bartz, and C. Meinel, "Learning to train a binary neural network," *arXiv preprint arXiv:1809.10463*, 2018.

[4] R2RT, *Binary Stochastic Neurons in Tensorflow*, 2018 (accessed October 20, 2018). [Online]. Available: https://r2rt.com/binary-stochastic-neurons-in-tensorflow.html#fn1

[5] D. M. Blei and M. I. Jordan, "Modeling annotated data," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003, pp. 127–134.

[6] M. Guillaumin, *COREL 5K*, 2018 (accessed October 20, 2018). [Online]. Available: http://lear.inrialpes.fr/people/guillaumin/data.php

[7] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[8] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate o (1/kˆ 2)," in *Dokl. Akad. Nauk SSSR*, vol. 269, 1983, pp. 543–547.

[9] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] S. Ruder, *An overview of gradient descent optimization algorithms*, 2018 (accessed October 20, 2018). [Online]. Available: http://ruder.io/optimizing-gradient-descent/

[12] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[13] H. Larochelle, *Neural networks [6.6] : Autoencoder - denoising autoencoder*, 2018 (accessed October 20, 2018). [Online]. Available: https://www.youtube.com/watch?v=t2NQ_c5BFOc

[14] A. Ng, *Stacked Autoencoders*, 2018 (accessed October 20, 2018). [Online]. Available: http://deeplearning.stanford.edu/wiki/index.php/Stacked_Autoencoders