HOMEWORK SET 5

CSCI 5980/8980 Think Deep Learning (Fall 2020)

Due 11:59 pm, Dec 22 2020

Instruction Typesetting your submission in LaTeX is now **optional**, and but you need to submit it as a single PDF file in Canvas. No late submission will be accepted. For each problem, your should acknowledge your collaborators if any. For problems containing multiple subproblems, there are often close logic connections between the subproblems. So always remember to build on previous ones, rather than work from scratch.

Notation We will use small letters (e.g., u) for scalars, small boldface letters (e.g., $\boldsymbol{a}$) for vectors, and capital boldface letters (e.g., $\boldsymbol{A}$) for matrices. For a matrix $\boldsymbol{A}$, $\boldsymbol{a}^i$ (supscripting) means its i-th row as a *row vector*, and $\boldsymbol{a}_j$ (subscripting) means the j-the column as a column vector, and a_{ij} means its (i, j)-th element. $\mathbb{R}$ is the set of real numbers. $\mathbb{R}^n$ is the space of n-dimensional real vectors, and similarly $\mathbb{R}^{m\times n}$ is the space of $m \times n$ real matrices. The dotted equal sign $\doteq$ means defining.

Problem 1 (transfer learning; 4/12) In computer vision and natural language processing, large-scale datasets are available and high-performing deep models that are already trained on these datasets, called **pretrained models**, are readily usable. For example, in Pytorch, a list of pretrained models on on renowned ImageNet dataset is available here https://pytorch.org/docs/stable/torchvision/models.html. Since these datasets are large-scale and believed to sufficiently represent the domain distributions, the features learned tend to be shareable across tasks. For example, in computer vision, when coming to a new image classification task, it is rare that people will train a model from scratch. Instead, a pretrained model will be taken and only fine tuning of the model on the new task will be performed.

The different possibilities of fine tuning have been explained in our class; this webpage provides an excellent summary https://cs231n.github.io/transfer-learning/. A Pytorch tutorial on implementing transfer learning for vision tasks can be found here https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html. In this problem, we will perform transfer learning for classification of pneumonia from chest x-rays.

(a) Read the instruction for this Kaggle competition https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia, download (you'll need a kaggle account) and setup the dataset. (1/12)

(b) Set up an appropriate transfer learning pipeline to perform the classification. Feel free to choose pretrained models you like (or can fit into your resource constraint—large models can be more powerful, but need powerful GPUs). You may want to play with different transfer learning strategy, and think about the following factors: (1) do you want to freeze all or only some of convolutional layers? (2) or do you want to make all trainable, but only iterate few steps? (3) or may be borrowing the model is sufficient and training can be done from scratch? (Hint on the training: the two classes are not balanced; it may be helpful to put different weights on the positive and negative samples—e.g., weighting the minority class slightly more than the dominant class—when constructing the training objective; lots of PyTorch functions already implement the weighting mechanism, e.g., the weight input in `torch.nn.CrossEntropyLoss` https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html)(2/12)

(c) A "90%" test: you'll get 1 point if your classification accuracy exceeds 90%. But make sure to show all your work in (b) even if you don't make it 90%. (1/12)

Problem 2 (recurrent neural networks; 3/12) In this problem, we will play with simple sentiment analysis based on text. The dataset can be found here https://www.kaggle.com/kazanova/sentiment140. This github site (https://github.com/bentrevett/pytorch-sentiment-analysis) includes detailed tutorials on performing sentiment analysis using basic and advanced RNN models in PyTorch on the classic IMDb dataset. Please go over the tutorials and feel free to adapt the codes there.

(a) Read the instruction from the Kaggle website and load the data from the sentiment140 dataset. We will use the text field to predict the target, i.e., polarity. The text field is not as clean as the IMDb dataset, e.g., the "@ xxxx" part is probably not useful for sentiment analysis. Perform data clean up when necessary. There is a single data file in the dataset. Please split it into 60% training, 20% validation, and 20% test. (1/12)

(b) Design and train a sentiment analysis model on the data. Again, feel free to start from the sentiment analysis tutorial mentioned above and adapt the models there (1.5/12).

(c) A "85%" test: you'll get 0.5 point if your classification accuracy exceeds 85%. (0.5/12)

Problem 3 (generative models; 5/12) Finally, we're here to generate new fashionable items! In other words, we will train generative models based on the famous Fashion-MNIST dataset https://github.com/zalandoresearch/fashion-mnist, which is available as a PyTorch standard dataset https://pytorch.org/docs/stable/torchvision/datasets.html#fashion-mnist. There possibly are implementations of the following algorithms on the Internet; you can occasionally consult these online resources when feeling uncertain, but your implementations must be your own work and copying from online sources is also considered plagiarism.

(a) Train a GAN and generate 10 new items after the training. For the GAN, you can use either the original form, or modified form based on other objective, e.g., W-GAN, or other modification. (2/12)

(b) Modify the above implementation into a conditional GAN, with the class labels as input augmentation to both the generator and discriminator. Repeat the training and generation, and show at least 1 new item from each class and visually compare with the results from (a). (1/12)

(c) Implement and train a variational autoencoder and similarly generate 10 random samples from it. As is standard in VAE, let's assume the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ and the conditional $p(\boldsymbol{x}|\boldsymbol{z})$ take multivariate Gaussian form with diagonal covariance structure. Sec. 3 and Appendix C of the original paper https://arxiv.org/abs/1312.6114 may help you to clear up doubts. Make sure to implement the reparametrization trick so that auto differentiation can be performed. (2/12)