CS69000-DPL Spring 2020 Homework	— Homework 5—	Last update: April 8, 2020
https://www.cs.purdue.edu/homes/ribeirob/courses/Spring2020/index.html		Due: **6:00am**, Sunday, April 19th

Instructions and Policy: Each student should write up their own solutions independently, no copying of any form is allowed. You MUST indicate the names of the people you discussed a problem with; ideally you should discuss with no more than two other people.
YOU MUST INCLUDE YOUR NAME IN THE HOMEWORK
You need to submit your answer in PDF. LaTeX is typesetting is encouraged but not required. Please write clearly and concisely - clarity and brevity will be rewarded. Refer to known facts as necessary.

Q0 (0pts correct answer, -1,000pts incorrect answer: (0,-1,000) pts): A correct answer to the following questions is worth 0pts. An incorrect answer is worth -1,000pts, which carries over to other homeworks and exams, and can result in an F grade in the course.

(1) Student interaction with other students / individuals:

   (a) I have copied part of my homework from another student or another person (plagiarism).

   (b) Yes, I discussed the homework with another person but came up with my own answers. Their name(s) is (are) _____

   (c) No, I did not discuss the homework with anyone

(2) On using online resources:

   (a) I have copied one of my answers directly from a website (plagiarism).

   (b) I have used online resources to help me answer this question, but I came up with my own answers (you are allowed to use online resources as long as the answer is your own). Here is a list of the websites I have used in this homework:
   _____

   (c) I have not used any online resources except the ones provided in the course website.

Homework 5

Q1: Theoretical Questions (2pts)

1. (1pt) Detail how we can replace the equivariant representation of the Transformer architecture with an *equivariant representation* using Janossy pooling and an LSTM architecture as $\vec{f}$ (see $\vec{f}$ in Equation (1) in our set representation lecture). Note that Janossy pooling originally gives invariant representations. Specifically, replace the representation of the i-th word of the m-th self-attention head, $z_i^{(m)}$, of our Transformer lecture with the corresponding LSTM output using Janossy pooling.
   Hint: Make sure your representation is equivariant. You will probably want to use the h_t representation of the LSTM for your task (see Backpropagation-through-time lecture where we define the LSTM variables).

2. (1pt) (a) Explain one of the main shortcomings of RNNs and how Bi-directional RNNs help ameliorate this shortcoming. (b) Are Bi-directional LSTMs able to represent well dependencies between the first and the last elements of a sequence? Why or why not?

Q2: Word Embeddings using 2nd order Markov Chains (2pts)

In class, we have seen that a word2vec model will define word embeddings using the SKIPGRAM model with window K (assumed to be an odd number) as a 1st order Markov chain. The negative log-likelihood (NLL) is (not accounting for the corner cases at the end and beginning of a sentence):

$$\mathcal{L}_{\text{word2vec}} = -\sum_{t=k}^{n-k}\sum_{k=1}^{(K-1)/2} (\log p(x_{t-k}|x_t;\mathbf{U},\mathbf{V}) + \log p(x_{t+k}|x_t;\mathbf{U},\mathbf{V})),$$

and the probability function is

$$p(x|x_t;\mathbf{U},\mathbf{V}) = \frac{1}{Z(x_t;\mathbf{U},\mathbf{V})}\exp(\langle \mathbf{U}x, \mathbf{V}x_t\rangle),$$

with appropriately-defined one-hot encoding of x_i, $i = 1,\ldots,n$, and $Z(x_t;\mathbf{U},\mathbf{V})$ is a normalization function.

Task (2pts): Write down the NLL loss function and the *probability function* of a word embedding model that, again using a window of size K similar to the original SKIPGRAM approach, can embed words based on a 2nd order Markov chain.

Hint 1: We must change the original SKIPGRAM model since the dataset it creates is for a 1st order Markov chain.
Hint 2: There are multiple ways to define the model p. Note that you will need more parameters. Also pay attention that your 2nd order Markov chain that cannot be described by a 1st order Markov chain.

Q3: Domain Adaptation with Maximum Mean Discrepancy (MMD) (6pts)

In this homework, you are faced with the problem of unsupervised domain adaptation. You are provided a data set of labeled images (CIFAR-10), which we'll call the *source* domain. Your task is to predict the labels for a *target* domain, in which the images have their color shifted. See an example in Figure 1.

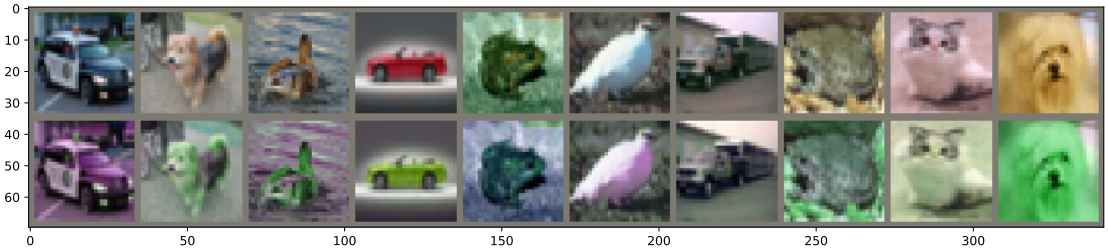

Figure 1: Example of images. *Top row*: Source domain, CIFAR-100. *Bottom row*: Target domain, shifted hue.

To aid you in this task, you also have access to *unlabeled* images from the target domain to use during training. In a more formal description:

- Training data: $\mathcal{D}^{(\text{tr})} = \{(x_i^{(\text{source})}, y_i^{(\text{source})})\}_{i=1}^{m} \cup \{x_i^{(\text{target})}\}_{i=1}^{n}$, where:
  - $(x_i^{(\text{source})}, y_i^{(\text{source})})$ are sampled iid from $p^{(\text{source})}(x)p(y|x)$
  - $x_i^{(\text{target})}$ are sampled iid from $p^{(\text{target})}(x)$
- Test data: $\mathcal{D}^{(\text{te})} = \{(x_i^{(\text{target})}, y_i^{(\text{target})})\}_{i=1}^{n^{(\text{te})}}$, where:
  - (x_i, y_i) are sampled iid from $p^{(\text{target})}(x)p(y|x)$.

For this homework, we will use the Maximum Mean Discrepancy (MMD) divergence to help us learn representations which are useful in both domains. Denote by $f(\Gamma(x; \boldsymbol{W}_\Gamma); \boldsymbol{W}_f)$ our neural network classifier, where $\Gamma(x; \boldsymbol{W}_\Gamma)$ is the part of the neural network which is tasked with learning the representations and $f(\cdot, \boldsymbol{W}_f)$ is the part of the neural network which is tasked on classifying an image given its representation.

We modify our loss function to incorporate the MMD as a regularization term:

$$\mathcal{L}(\mathcal{D}^{(\text{tr})}; \boldsymbol{W}_f, \boldsymbol{W}_\Gamma) = \mathcal{L}_{CE}(\mathcal{D}^{(\text{tr})}; \boldsymbol{W}_f, \boldsymbol{W}_\Gamma) + \lambda \cdot \text{MMD}^2(\mathcal{D}^{(\text{tr})}; \boldsymbol{W}_\Gamma),$$

where $\mathcal{L}_{CE}$ is the usual cross-entropy loss and λ is a hyperparameter.

We will be implementing MMD with the Gaussian kernel:

$$k(\boldsymbol{z}, \boldsymbol{z}') = \exp\left(-\frac{\|\boldsymbol{z} - \boldsymbol{z}'\|_2^2}{2\sigma^2}\right),$$

where σ is a hyperparameter of the kernel. We will apply the MMD to the representations computed by $\Gamma(\cdot; \boldsymbol{W}_\Gamma)$ for both domains, with the goal of learning a representation that is invariant to both domains. To compute the MMD term, we will use the unbiased estimator of Gretton et al. [2012] (Eq. 3, Lemma 6).

Action Items:

In your report, include the answer of the following questions:

1. (1 pt) Is this problem an example of *covariate shift* or *label shift*? Why? (describe through equations)
2. (1 pt) In your report, write the MMD equation below, replacing the question marks by the appropriate variables. Assume a batch of m examples from the source domain and a batch of n examples from the target domain. Your answer *must* use the variables variables $\boldsymbol{x}^{(\text{source})}, \boldsymbol{x}^{(\text{target})}, m, n$.

$$\begin{aligned}
\widehat{\text{MMD}}^2(\mathcal{D}^{(\text{tr})}; \boldsymbol{W}_\Gamma) &= \\
&= \frac{1}{?(?-1)} \sum_{i=1}^{?} \sum_{j \neq i}^{?} k(\Gamma(?_i; \boldsymbol{W}_\Gamma), \Gamma(?_j; \boldsymbol{W}_\Gamma)) \\
&\quad + \frac{1}{?(?-1)} \sum_{i=1}^{?} \sum_{j \neq i}^{?} k(\Gamma(?_i; \boldsymbol{W}_\Gamma), \Gamma(?_j; \boldsymbol{W}_\Gamma)) \\
&\quad - \frac{2}{??} \sum_{i=1}^{?} \sum_{j=1}^{?} k(\Gamma(?_i; \boldsymbol{W}_\Gamma), \Gamma(?_j; \boldsymbol{W}_\Gamma)).
\end{aligned}$$

3. (4 pts) Implement the MMD estimator in the file `mmd.py` and:
   (a) (2 pts) Run the code without using MMD, through the command `python hw5.py --reg_str 0`. Include in your report:
      i. Curve of training loss for each epoch.
      ii. Curve of validation accuracy for each epoch, for both source and target domain.
      iii. Final test accuracy of both source and target domain, for the best model.
   (b) (2 pts) Run the code with MMD, through the command `python hw5.py --reg_str 10 --kernel_sigma 20`. Include in your report:
      i. Curve of training loss for each epoch.
      ii. Curve of validation accuracy for each epoch, for both source and target domain.
      iii. Final test accuracy of both source and target domain, for the best model.

      Hint: If you have GPUs available, pass the option `--gpu 0` to use GPU 0. The code will run much faster on a GPU. The scholar cluster has GPUs, see the Piazza post on how to use it.

References

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012. URL `http://www.jmlr.org/papers/v13/gretton12a.html`.

Submission Instructions

Please read the instructions carefully. Failing to follow the instructions might lead to loss of points.

Naming convention: [your_purdue_login]_hw-5

All your submission files, including a ReadMe, and codes, should be included in one folder. The folder should be named with the above naming convention. For example, if my purdue account is "jsmith123", then for Homework 5 I should name my folder as "jsmith123_hw-5".

Remove any unnecessary files in your folder, such as training datasets (Data folder). Make sure your folder is structured as the tree shown in Overview section.

Submit: **TURNIN INSTRUCTIONS**

Please submit your homework files on **data.cs.purdue.edu** using the turnin command, e.g.:

`turnin -c cs690-dpl -p hw-5 jsmith123_hw-5`.

Please make sure you didn't use any library/source explicitly forbidden to use. If any such library/-source code is used, you will get 0 pt for the coding part of the assignment. If your code doesn't run on scholar.rcac.purdue.edu, then even if it compiles in another computer, your code will still be considered not-running and the respective part of the assignment will receive 0 pt.