Homework 3

Transfer Learning
Due October 30, 2019

1 Introduction

In this assignment you will learn about transfer learning. When a problem you want to solve does not have enough data, we use a different (larger) dataset to learn representations which can help us solve our task using the smaller task.

The abstract steps to transfer learning are as follows:

- Find a huge dataset with similar characteristics to the problem you are interested in.
- Choose a model powerful enough to extract meaningful representations from the huge dataset.
- Train this model on the huge dataset.
- Use this model to train on the smaller dataset.

2 Question 1

To grasp the high-level approach to transfer learning, let's first do a simple example using computer vision. The torchvision library has pretrained models (resnets, vggnets, etc) on the Imagenet dataset. Imagenet is a dataset with 1.3 million images covering over 1000 classes of objects. When you use one of these models, the weights of the model initialize with the weights saved from training on imagenet.
In this task we will:

- Choose a pretrained model (resnet18).
- Freeze the model so that the weights don't change.
- Fine-tune on a few labels of MNIST.

2.1 Part A

Your task here is to decide what model to use for fine-tuning. This can be a simple as a single layer MLP, or as powerful as another resnet with an MLP layer to map from the last output vector to class probabilities.

2.2 Part B

Here you will finetune your model.
There are two ways of finetuning.

Frozen feature extractor In the first type we pretrain with the FROZEN feature extractor and NEVER unfreeze it during finetuning.

Unfrozen feature extractor In the second, we finetune with a FROZEN feature extractor for a few epochs, then unfreeze the feature extractor and finish training.

Task In this section we'll do the first version of fine-tuning. Implement the FROZEN_fine_tune_mnist function which will update the weights of the fine_tune_model **only** - NOT the feature extractor.

2.3 Part C

Here you'll calculate the testing accuracy of your full model (feature extractor + fine tune model). Return a single scalar value $s \in [0, 1]$.

2.4 Part D

In this section you'll finetune your model using the **unfrozen** approach. To do this:

- keep the feature_extract frozen for a few epochs (10)
- Unfreeze the feature_extractor.
- Finish training.

Here you'll calculate the testing accuracy of your full model (feature extractor + fine tune model). Return a single scalar value $s \in [0, 1]$.

2.5 Grading

To see how well you did, run the grade_mnist_frozen function and the grade_mnist_unfrozen function. The latter should be higher than the former.

3 Question 2

Here we'll apply what we just learned to NLP. In this section we'll make our own feature extractor and pretrain it on Wikitext-2.
The WikiText language modeling dataset is a collection of over 100 million tokens extracted from the set of verified Good and Featured articles on Wikipedia. The dataset is available under the Creative Commons Attribution-ShareAlike License.

3.1 Part A

In this section you need to generate the training, validation and test split. Feel free to use code from your previous lectures.

3.2 Part B

Here we design our own feature extractor. In MNIST that was a resnet because we were dealing with images. Now we need to pick a model that can model sequences better. Design an RNN-based model here.

3.3 Part C

Here we design our own feature extractor. In MNIST that was a resnet because we were dealing with images. Now we need to pick a model that can model sequences better. Design an RNN-based model here.

3.4 Part D

Calculate the test perplexity on wikitext2. Feel free to recycle code from previous assignments from this class.

3.5 Grading

Run the grading function to see how you did.

4 Question 3

In this question you will use your feature$_extractor from question 2 to fine-tune on MNLI.$
(From the website): The Multi-Genre Natural Language Inference (MultiNLI) corpus is a crowd-sourced collection of 433k sentence pairs annotated with textual entailment information. The corpus is modeled on the SNLI corpus, but differs in that covers a range of genres of spoken and written text, and supports a distinctive cross-genre generalization evaluation. The corpus served as the basis for the shared task of the RepEval 2017 Workshop at EMNLP in Copenhagen.
MNLI has 3 genres (3 classes). The goal of this question is to maximize the test accuracy in MNLI.

4.1 Part A

In this section you need to generate the training, validation and test split. Feel free to use code from your previous lectures.

4.2 Part B

Here we again design a model for finetuning. Use the output of your feature-extractor as the input to this model. This should be a powerful neural classifier (up to you).

4.3 Part C

Use the feature extractor and your finetune model to finetune MNLI.

4.4 Part D

In this section please calculate the test accuracy on MNLI

4.5 Grading

Run the grading function to see how you did.

5 Question 4

A major direction in research came from a model called BERT, released last year. In this question you'll use BERT as your feature extractor instead of the model you designed yourself.

5.1 Part A

In this section you need to set up BERT for this task. Init a BERT instance using whatever hyperparameters you want.

5.2 Part B

Use BERT as your feature extractor to finetune MNLI. Use a new finetune model (reset weights).

5.3 Part C

In this section please calculate the test accuracy on MNLI.

5.4 Grading

Run the grading function to see how you did.

References