HW2: Fine-tuning Language Models

CSCI 662: Fall 2024

out: Sep 30, 2024
due: Oct 18, 2024

Acknowledgement: This homework is inspired by He He's CSCI-GA 2590 homework 3, which was in turn inspired by NL-Augmenter - https://arxiv.org/abs/2112.02721.

Overview

This assignment has two parts. In the first part, which should be **very brief**, you'll review the equations for nonlinear models and do one short hand-calculated gradient update. In the second part, you will be fine-tuning a language model for a specific downstream task and understand the challenges involved in a practical setting. You will fine-tune a DistilBERT-base model for sentiment analysis using the IMDB dataset and then explore the challenges of out-of-distribution test sets.

Part 1: Model Equation Review (10%)

For this part of the homework, we're just looking for the requested answers. You don't have to write a research report. You may use *any* resources, including notes, the textbook, or anything you find online. You may discuss with classmates, instructors, and can ask questions about this hw on Slack. However, your writing should be your own (i.e., please don't directly copy off of each other; the point is to understand what you're doing).

Consider a feed-forward neural network classification model with input of batch size t and feature size f that predicts a class from label set of size m. There is a single hidden layer of size d. Hidden states and logits are formed with a bias term. The non-linear function used to form the hidden state is termed g but is concretely the ReLU function:

$$g(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$$

Let the weights for transforming from features to hidden dimension be the matrix H and let its bias vector be b_H. Let the weights for transforming from hidden dimension to classification dimension be U and its bias vector b_U.

Let X be a batch of feature vectors, represented as a matrix, and let Y be one-hot labels for each vector in X. Let K be the linear projection of X through H and b_H. Let S be the hidden state (g applied to K).

Let Z be the logits, i.e. the linear projection of S through U and b_U. Let O be the distribution over the label space of size m, i.e. softmax applied to Z. Let the loss ℓ be the cross-entropy of p and O, where

$$p(x, y) = \begin{cases} 1 & Y[x][y] = 1 \\ 0 & \text{otherwise} \end{cases}$$

for item (row) x in X. In each question, be sure to use transpose operator a^T where needed to turn $(i \times j)$ matrix a into a $(j \times i)$ matrix, and write terms (particularly matrices to be multiplied) in proper order. $a; b$ means concatenation along the columnar dimension. ab means matrix multiplication; if a has dimension $(i \times j)$ and b has dimension $(j \times k)$ then ab has dimension $(i \times k)$. $g(a)$ is elementwise function application. $a \odot b$ is Hadamard product (i.e., elementwise multiplication).

Problem 1

Draw a computation graph of the aforementioned feed-forward network that includes the following variables: $X, Y, H, b_H, K, S, U, b_U, Z, O, \ell$. A computation graph has one node for each variable and a directed arc $a \to b$ if the equation used to calculate b contains a. Label each node with the relevant variable. For variables, indicate the dimensions of the variable (noted in the description of this section) and the equation used to calculate the variable (unless there are no incoming arcs). For example, the equation to calculate K is $K = XH + b_H$ and the dimensions of X are $t \times f$. Thus, a node should be drawn for each of X, H, b_H, and K, arrows should lead from each of the first three to K, the equation noted for K should be the one from the prior sentence, and the dimensions noted for X should be the dimensions noted in the prior sentence. You can use g for ReLU and σ for softmax. You can use O_y to mean "$O[:][y]$ such that $Y[:][y] = 1$."[1]

Problem 2

Write the equation for each of the following partial derivatives as a product of two partial derivatives that are calculatable using the computation graph from Problem 1. One example has been provided. Let y be a subscript indicating the position along a range of m for which the relevant row in Y has a non-zero value.

$$\frac{\partial \ell}{\partial Z} = \frac{\partial \ell}{\partial O_y} \frac{\partial O_y}{\partial Z} \qquad \frac{\partial \ell}{\partial U} =$$

$$\frac{\partial \ell}{\partial K} = \qquad \frac{\partial \ell}{\partial b_U} =$$

$$\frac{\partial \ell}{\partial S} = \qquad \frac{\partial \ell}{\partial H} =$$

$$\frac{\partial \ell}{\partial b_H} =$$

Problem 3

Write the *actual* equation for each of the following partial derivatives in terms of functions mentioned above and variables in the computation graph; equations should not contain partial derivatives in their right hand sides. Show all your work for $\frac{\partial O_y}{\partial Z}$ and express the final result in terms of O; for the rest you can simply show the answer.

$$\frac{\partial \ell}{\partial Z} = \qquad \frac{\partial Z}{\partial b_U} = \qquad \frac{\partial S}{\partial K} =$$

$$\frac{\partial \ell}{\partial O_y} = \qquad \frac{\partial Z}{\partial U} = \qquad \frac{\partial K}{\partial H} =$$

$$\frac{\partial O_y}{\partial Z} = \qquad \frac{\partial Z}{\partial S} = \qquad \frac{\partial K}{\partial b_H} =$$

[1] The computation graph described here will look a little different from the one presented in class; if you use the one presented in class instead (that one includes functions as nodes, but is, in retrospect, not as principled) that is acceptable.

Problem 4

Given the following values for the feed-forward network described above:

$$X = \begin{bmatrix} 1 & -1 & 0 \\ -2 & 2 & 1 \\ -1 & 0.5 & 3 \\ 1 & 0.5 & 0.5 \end{bmatrix} \quad H = \begin{bmatrix} 0.1 & -0.1 & 0.05 & -0.05 & 0.2 \\ 0.05 & -0.0 & 0.1 & 0.1 & -0.2 \\ 0.2 & -0.05 & 0.05 & -0.1 & 0.2 \end{bmatrix} \quad b_H = \begin{bmatrix} 0.1 & -0.1 & -0.05 & 0.05 & 0.1 \end{bmatrix}$$

$$Y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 0.06 & -0.1 & -0.2 \\ 0.07 & 0.2 & -0.03 \\ 0.08 & -0.3 & -0.04 \\ 0.09 & 0.4 & -0.02 \\ 0.1 & -0.5 & -0.1 \end{bmatrix} \quad b_U = \begin{bmatrix} 0.2 & 0.1 & 0.05 \end{bmatrix}$$

calculate:

1. ℓ for each item in X
2. New values for H, b_H, U, b_U after backpropagating the gradient of ℓ (assume a learning rate of 1).
3. ℓ for each item in X with the new values.

Show your work. This can be done by providing (upload to vocareum) code, a notebook, or even pen-and-pencil evidence.

Fine-Tuning (90%)

One of the main assumptions made in supervised learning is *i.i.d.* (independent and identically distributed) test distribution. In other words, the test data is drawn from the same distribution as the training data. However, this assumption does not hold well in practice, especially in the real world. For example, the training data may have been collected with text that is well-formed and clean, but the resulting system from this data is deployed into the real world where these assumptions may not hold. This means that there are certain features that are specific to the dataset that may not work for other examples of the same task.

Therefore, the next main objective of this homework is to create transformations of the dataset to create out-of-distribution (OOD) data and evaluate your fine-tuned model on it. The aim is to construct transformations that are 'reasonable' (e.g. something that you can actually expect at test time) but not very trivial.

As with the previous homework, the main goal is to conduct a scientific study and write a report that clearly communicates what you have done, why you have done it, and what is interesting about it. Your report should be at least two pages long and not exceed four pages, excluding references.

Requirements

Fine-Tuning

1. First, you will fine-tune a DistilBERT-base language model on the sentiment analysis dataset based on movie reviews on IMDB.[2]
2. We have provided you with starter code for fine-tuning DistilBERT, but it contains some missing parts. Complete the missing parts in the `do_train` function in `main.py`, which is mostly the training loop. You are not required to modify any default hyperparameters but you are welcome to play with them.

[2] https://huggingface.co/datasets/imdb

3. Run `python main.py --train --eval`, which will fine-tune DistilBERT and evaluate on the test data. You should submit the generated output file for this part. An accuracy of more than 87% will earn you full points for code correctness.

Transformations

1. Next, you will design transformations to apply to the evaluation dataset which will serve as an out-of-distribution evaluation for your model. These transformations should be designed such that the new transformed example has the **same label** as the original example. In other words, a human would assign the same label to the original and transformed example. For example, a transformation that maintains the same label as "Titanic is the best movie I've ever seen!" is "Titanic's the best film I have ever seen."

2. Design a transformation and explain what it does. You should also explain why it is a reasonable transformation. Whether a transformation is reasonable is subjective, but use your best judgment. We will be lenient with what is reasonable apart from extreme cases such as using gibberish (e.g., 'The soup was hot' becomes 'The unnjk rxasqwer hot.').

3. Implement your proposed transformations. At a minimum, implement synonym replacement and synthetic typos. We have provided the rough guidelines for these implementations, so you can use those or use your own. We also provide a template for a toy transformation named `example_transform`, which you should follow to fill in the function `custom_transform()` in `utils.py`. To debug and see a few examples, run `python main.py --eval_transformed --debug_transformation`.

4. Your goal is for the transformations to be out of domain enough that the model performance *degrades*. You will see this when you run `python main.py --eval_transformed`. Any drop of up to four accuracy points will give you half credit for code correction in this part. A drop of more than 4 will get you full points.

Data Augmentation

1. One simple way to improve performance on the transformed set is to apply a similar transformation to the training data and train your model on the combination of the original data and the transformed training data. This is known as data augmentation. Augment the training data with 5000 randomly transformed examples to create the new augmented training dataset.

2. Fill in `create_augmented_data` in `main.py` to complete the code.

3. Run `python main.py --train_augmented --eval_transformed` to train a model on this augmented data and submit the generated output file.

4. Evaluate the trained model on the original test data using `python main.py --eval --model_dir out_augmented` and on the transformed test data using `python main.py --eval_transformed --model_dir out_augmented` and report the accuracies.

5. Does the model performance improve on the transformed test set after data augmentation? How does the model performance compare to the original test data after data augmentation? Explain the result with intuitive reasons.

6. Elaborate on one limitation of the data augmentation approach used here for improving performance on OOD test sets.

Coding Recommendations

- Get to a command line environment where you have GPU access. This can be a modern Mac with an M1 or M2 chip, or it can be HPCC discovery (we will provide access) or it can be any other infrastructure you have access to. If vocareum works you can perhaps do this work there.
- WARNING: CARC is a shared resource and sometimes it will take quite a while to get the resources you need. Plan ahead, and do not wait until the last minute to try to get this homework done!
- Here is a basic way to get a node with GPUs on discovery for one hour:
- Download the starter code from vocareum. Use scp to transfer the code over and use ssh to log in to `discovery.usc.edu`. Create a conda environment and install appropriate packages:

```
conda create -n cs662_hw2_24 python=3.12
conda activate cs662_hw2_24
```

```
salloc --time=1:00:00 --gres=gpu:v100:2 -p gpu
```

- You should read Slurm and CARC access manuals on carc.usc.edu for much more information on how to use them. In particular, learning about `sbatch` lets you submit asynchronous jobs that will run while you're not logged in. However, `salloc` is very helpful for getting started.
- For development purposes, start small and see that the training loop works as expected (i.e. loss drops, you can overfit to the training set, etc.). Training the model on the full dataset will take some time (more than 1 to 2 hours, depending on the GPU you have), so only run training on the full dataset once you are confident that your setup is correct.
- You are more than welcome to use any packages out there to help you train your model and augment the dataset. More time should be spent on trying out interesting transformation or modeling techniques to see if you can further boost performance on the original test set and your transformed test set. If you are having trouble getting the basic training going please reach out for help; this shouldn't be where you spend most of your time.

Your Report for part 1

Submit part 1 in a pdf of any format. It should be no more than two pages, and can probably be a single page long. It can be hand-written and scanned, typed in Word, or presented in whatever way is convenient

Your Report for part 2

Your report for this part should at a *minimum*:

- Describe the motivations for the transformations that you have applied and how you have implemented them.
- Analyze and discuss the performance difference between the original test set and the transformed test set.
- Describe technical details: any pre/post-processing steps, compute, training strategy, learning rate, etc.

Use the ACL style files: https://github.com/acl-org/acl-style-files

Your report should be a pdf of at least two pages long, including references, and not more than four pages long, not including references (i.e. you can have up to four pages of text if you need to). Just like a conference paper or journal article, it should contain an abstract, introduction, experimental results, and conclusion sections (as well as other sections as deemed necessary). Unlike a conference paper/journal article, a complete related works section is not obligatory (but you may include it if it is relevant to what you do).

Also submit your code, including modifications to the provided files and any additional code or configuration files, such that the staff can run your code. You do **not** need to (and should not) submit your model files.

Grading

Grading will be roughly broken down as follows:

- 5% – Did you get the equations right in the first part? You can check with the instructors ahead of time (during office hours; we're not guaranteed to be available last minute) so you should be able to get this perfect.
- 5% – Did you do the calculations right in the first part? You can check with the instructors ahead of time (during office hours; we're not guaranteed to be available last minute) so you should be able to get this perfect as well.
- 50% – Does your report for the second part read like a research paper? Did you clearly communicate your description of what you implemented, how you implemented it, what your experiments were, and what conclusions you drew from them? This includes appropriate use of graphics and tables where warranted that clearly explain your point. This also includes well-written explanations that tell a compelling story. Grammar and syntax are a small part of this (maybe 5% of the grade, so 10% of this section) but much more important is the narrative you tell. Also, a part of this is that you clearly acknowledged your sources and influences with appropriate bibliography and, where relevant, cited influencing prior work.
- 15% – Is your code correct? Did you implement what was asked for, and did you do it correctly?
- 15% – Is your code well-written, documented, and robust? Will it run from a different directory than the one you ran it in? Does it rely on hard-codes (other than those in the provided starter code)? Is it commented and structured such that we can read it and understand what you are doing?
- 10% – Did you go the extra mile (in the second part)? Did you push beyond what was asked for in the assignment, trying (well-justified) new models, features, or approaches? Did you use motivation (and document appropriately) from another researcher trying the same problem or from an unrelated but transferable paper?

'Extra Mile' ideas

This is not meant to be comprehensive and you do not have to do any of the things here (nor should you do all of them). But an 'extra mile' component is 10% of your grade (**not extra credit!**).

- Implement augmentations beyond synonym replacement and synthetic typos.
- Try other model architectures for this task. Explain why you get better or lower performance.
- Try training with additional data from other sentiment analysis datasets.
- Dig into the ACL archives and find ideas for other architectures, augmentations, or approaches; try them out, and analyze performance.

Rules

- This is an individual assignment. You may not work in teams or collaborate with other students. You must be the sole author of 100% of the code you turn in.
- Depending on the need and class interest, we may collaborate *in class* or *publicly on Slack* if you get stuck; this kind of collaboration is okay.
- You may not look for coded solutions on the web, or use code you find online or anywhere else. You can and are encouraged to read material beyond what you have been given in class (see above) but should not copy code.
- Generative language, code, and vision models (e.g. ChatGPT, Llama 2, Midjourney, Github Copilot, etc.; if you are unsure, ask and don't assume!!) can be used (to aid in report writing/coding, not to actually do the classification tasks) with the following caveats:
  - You must declare your use of the tools in your submitted artifact. If you don't declare the tool usage but you did use these tools, we will consider that plagiarism.
  - For code and image generation, you must indicate the prompt used and the output generated
  - For text generation you must provide either a link to the chat session you used to help write the content or an equivalent readout of the inputs you provided and outputs received from the system. You will lose credit if "the AI" is doing the work rather than you.
- Failure to follow the above rules is considered a violation of academic integrity and is grounds for failure of the assignment or, in serious cases failure of the course.
- We use plagiarism detection software to identify similarities between student assignments, and between student assignments and known solutions on the web. Any attempt to fool plagiarism detection, for example the modification of code to reduce its similarity to the source, will result in an automatic failing grade for the course.
- If you have questions about what is and isn't allowed, post them to Slack!