Michael Pollack: LING 574 HW6

Due 11PM on May 9, 2024

Note: I was given permission by Shane on 4/15 to turn in this homework before the late submission deadline on 5/11 with no late penalty due to my sister's wedding.

1 Recurrent Neural Network Encoders [30 pts]

Q1: Understanding RNNs [10 pts]

- What is the main limitation of feed-forward neural networks that is overcome by recurrent networks, and how do recurrent networks achieve this? [6 pts]

  Answer:
  The main limitation of feed-forward neural networks is their inability to handle sequential data, or any data with temporal dependencies. The flow of information from input to output in a feed-forward model prevents the model's ability to recognize the history of the data

  RNNs overcome this limitation by creating loops in the network that allow them to process sequences of inputs while maintaining information about previous inputs that they have seen. This is primarily accomplished through the recurrent connection, which facilitates a process that ensures the output of the network is dependent not only on the current input, but also on the hidden state from the previous time step.

- The Vanilla RNN equation has the form $h_t = f(h_{t-1}, x_t)$. What extra 'ingredient' does the LSTM add to this general form? What problem is the LSTM designed to solve? [4 pts]

  Answer:
  The spicy new ingredient added by LSTMs to the RNN equation is the introduction of a memory cells, which are governed by an input gate, an output gate, a forget gate, and a candidate. This addition is meant to help the model address long-term dependencies, which are often overlooked in Vanilla RNNs due to the vanishing gradient problem.

Q2: LSTM Update One of the "central" equations in the LSTM computation is the following:

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

This equation performs an essential update of one part of the LSTM. Please answer: [10 pts]

- What is c_t?

  Answer:
  c_t represents the cell state at time step t in the LSTM architecture. This cell state stores information the is relevant to predicting the output at the current time step.

- What is the range of f_t and what is its purpose?

  Answer:
  f_t represents the forget gate output. It's range is $[0, 1]$ and it serves to regulate which information from the previous cell state should be retained or discarded. If the value is close to 0, tghe information will be mostly discarded. If the value is closer to 1, it will be mostly retained.

- What is the range of i_t and what is its purpose?

  Answer:
  i_t represents the input gate output. The range of i_t is similarly $[0, 1]$. This vector serves to control the amount of information added to the cell state c_t from the candidate cell state $\hat{c}_t$ (See description of $\hat{c}_t$ below). If the value of a an element within i_t is close to zero, then its corresponding element in $\hat{c}_t$ is mostly ignored and little information therefrom is added to the new cell state. If an element is closer to 1, however, the information is deemed to be more important, and is passed on to the current cell state.

- What is $\hat{c}_t$?

  Answer:
  $\hat{c}_t$ represents the candidate cell state at time step t in an LSTM network. It is based on the information from the current input and it represents the new candidate values for updating the cell state c_t.

- In your own words, describe how this equation implements the central "update" inside of an LSTM.

  Answer:
  The LSTM update works by combining the previous cell state c_{t-1} (moderated by the forget gate outputf_t) and the candidate cell $\hat{c}_t$ (moderated by the input gate output i_t). The forget gate and previous cell state combination serves as an implementation of memory. The forget gate itself uses weight matrices to combine the current input with the previous hidden state (in addition to a bias term), then puts that whole thing through a sigmoid function in order to determine what information in the previous state was important and what should be forgotten. This "forgetting" is applied by the cross multiplication of f_t with c_{t-1}. The input gate similarly uses a sigmoid function over the combination of current input and previous hidden state with respective weight matrices, which is applied by its combination with the candidate cell $\hat{c}_t$. The addition of these two cross multiplications gives us c_t, or our current cell state.

Q3: Counting parameters Let d_e be the dimension of word embeddings and d_h the hidden state size. Focusing on just the recurrent cell (and so ignoring the embedding and output layers): [10 pts]

- How many parameters are there in a Vanilla RNN cell? [4 pts]

  Answer:
  A vanilla RNN cell contains the input-to-hidden weights, which would be of size $d_e \times d_h$. Then there are the hidden-to-hidden weights, which are of size $d_h \times d_h$. Finally, there are the biases. Since there are typically one of these for each hidden unit, that would give us d_h biases. Together this gives us: $(d_e \times d_h) + (d_h \times d_h) + d_h$ parameters

- How many parameters are there in an LSTM cell? [6 pts]

Answer:
Similarly to a Vanilla RNN cell, we have input-to-hidden weights, hidden-to-hidden weights, and biases that contribute to our parameters. In LSTMs, however, we also have four gates: the input gate, the forget gate, the output gate, and the cell update gate. With this in consideration, we get the following:
$4((d_e \times d_h) + (d_h \times d_h) + d_h)$ parameters

Note: for this problem, you can assume that the RNN cell is at the 'bottom' of a possibly-deep RNN, so the inputs to the cell are word embeddings, not earlier layers' hidden states.

2 Implementing an RNN Sentiment Classifier [30 pts]

See hw6.tar.gz

3 Running the Classifier [15 pts]

`run.py` contains a basic training loop for SST classification, using the last hidden state of an RNN. It will record the training and dev loss at each epoch, and save the best model according to dev loss. At the end, it samples 10 random dev data points and prints the review, the gold label, and the model's prediction.

Q1: Four different runs By default, a Vanilla RNN will be used. You can use an LSTM by specifying `--lstm` as a command-line argument. Following this paper, we have added dropout to the non-recurrent connections (i.e. from the inputs and to the output) of the model.
Please run each of the following variations. For each run, include in your readme.pdf: the best dev loss, the epoch at which the best dev loss was achieved, and the best model's dev accuracy. [8 pts]

- Vanilla RNN, default parameters. (This is just `run.py` with no command-line arguments.)
  Answer:
  Best Dev loss → Epoch 1 dev loss: 1.568380355834961
  Best model dev accuracy: 0.28701180744777477

- Vanilla RNN, with L_2 regularization (via `--l2`) at 1e-4 and dropout (via `--dropout`) at 0.5.
  Answer:
  Best Dev loss → Epoch 7 dev loss: 1.5616225004196167
  Best model dev accuracy: 0.29064486830154407

- LSTM, default parameters.
  Answer:
  Best Dev loss → Epoch 5 dev loss: 1.4644883871078491
  Best model dev accuracy: 0.36966394187102636

- LSTM, with L_2 regularization (via `--l2`) at 1e-4 and dropout (via `--dropout`) at 0.5.
  Answer:
  Best Dev loss → Epoch 15 dev loss: 1.3695586919784546
  Best model dev accuracy: 0.40599455040871935

Q2: Inspecting outputs For the fourth run above, please include in your readme.pdf the 10 random dev examples, with gold labels and model predictions here. In 2-3 sentences, describe what you see and observe any trends in what the model gets right and what (and/or how) it gets things wrong. [7 pts]

Examples:
Review: it 's clear the filmmakers were n't sure where they wanted their story to go , and even more clear that they lack the skills to get us to this undetermined destination .
Gold label: 1
Model prediction: 1

Review: it 's sweet , harmless , dumb , occasionally funny and about as compelling as a fishing show .
Gold label: 2
Model prediction: 4

Review: for starters , the story is just too slim .
Gold label: 1
Model prediction: 1

Review: green might want to hang onto that ski mask , as robbery may be the only way to pay for his next project .
Gold label: 0
Model prediction: 1

Review: adults will wish the movie were less simplistic , obvious , clumsily plotted and shallowly characterized .
Gold label: 1
Model prediction: 1

Review: dragonfly has no atmosphere , no tension – nothing but costner , flailing away .
Gold label: 0
Model prediction: 1

Review: it 's a bit disappointing that it only manages to be decent instead of dead brilliant .
Gold label: 1
Model prediction: 1

Review: a quiet treasure – a film to be savored .
Gold label: 4
Model prediction: 3

Review: a coarse and stupid gross-out .
Gold label: 1
Model prediction: 1

Review: it 's not that kung pow is n't funny some of the time – it just is n't any funnier than bad martial arts movies are all by themselves , without all oedekerk 's impish augmentation .
Gold label: 1
Model prediction: 1

Answer:
The model seemed to mostly get close to the gold label, with the second review being the most inaccurate. This deviation appears to be due to the somewhat forgiving tone of the review, featuring the words "sweet", "harmless", and "compelling". The model seemed reluctant to give out any zeroes, giving a 1

to each review with a 0 label. This seems to be the tendency of the model, as it consistently correctly predicts all of the reviews with a 1 gold label.

Submission Instructions

In your submission, include the following:

- readme.(txt|pdf) that includes your answers to §1 and §3.
- `hw6.tar.gz` containing:
  - run_hw6.sh. This should contain the code for activating the conda environment and your run commands for §3 above. You can use run_hw2.sh from the previous assignment as a template.
  - data.py
  - model.py