A3 homework submission
Team: the gurecki
Deep Learning 2015, Spring

David Halpern
Department of Psychology
New York University
david.halpern@nyu.edu

Anselm Rothe
Department of Psychology
New York University
ar3918@nyu.edu

Alex Rich
Department of Psychology
New York University
asr443@nyu.edu

1 Architecture

In order to leverage the advantages of both character-based and word-based models, we trained two models in parallel using completely different representations of the underlying data, and then combined their predictions to form our final model.

1.1 Model 1

We extended the baseline model to use Glove 100 [1] (we couldn't solve memory errors that we got for larger Glove files). We added a hidden layer with 500 units.

```
(1): nn.Reshape(100)
(2): nn.ReLU
(3): nn.Linear(100 -> 500)
(4): nn.Dropout
(5): nn.ReLU
(6): nn.Linear(500 -> 5)
(7): nn.LogSoftMax
```

1.2 Model 2

For model 2, we used the Crepe [2] model of character-based text recognition. This model begins with the raw character representation of the text and transforms it through several layers of temporal convolution and temporal max pooling.

```
  (1): nn.TemporalConvolution
  (2): nn.Threshold
  (3): nn.TemporalMaxPooling
  (4): nn.TemporalConvolution
  (5): nn.Threshold
  (6): nn.TemporalMaxPooling
  (7): nn.TemporalConvolution
  (8): nn.Threshold
  (9): nn.TemporalConvolution
```

```
(10): nn.Threshold
(11): nn.TemporalConvolution
(12): nn.Threshold
(13): nn.TemporalConvolution
(14): nn.Threshold
(15): nn.TemporalMaxPooling
(16): nn.Reshape(8704)
(17): nn.Linear(8704 -> 1024)
(18): nn.Threshold
(19): nn.Dropout
(20): nn.Linear(1024 -> 1024)
(21): nn.Threshold
(22): nn.Dropout
(23): nn.Linear(1024 -> 5)
(24): nn.LogSoftMax
```

2 Learning Techniques

Both models used dropout of 0.5. We did not augment the data.

3 Training Procedure

For both models we used the typical classNLLcriterion as the loss function. We used a validation set of 13K documents (10%) per class of the original 130K labeled training examples per class. The optimization procedure was run over the remaining 117K (90%) training documents per class.

3.1 Model 1

Model 1 was trained in 10 epochs using stochastic gradient descent with a batch size of 32, a learning rate of .1, a learning rate decay of 1×10^{-7}, a momentum of .9, and no weight decay.

3.2 Model 2

Model 2 was trained in 7 epochs using stochastic gradient descent with a batch size of 128, fixed data width of 1014, a momentum of 0.9, a learning rate which started at 0.01 and was halved periodically, and a weight decay of 1×10^{-5}.

4 Individual model performance

Performance for the two models is shown in Figure 1. Model 2 reached a performance of 65% accuracy in the training set, and 60% accuracy on the validation set. Model 1 reached a performance of only 44% accuracy on the training set, with, surprisingly, nearly identical performance on the validation set. (For this reason, and because Model 1 was inexplicably 8gb in size, we believe there may have been some remaining bugs in this model.)

5 Model averaging

To yield a final prediction, we combined the predictions of the two models. Despite the better performance of Model 2, we believe the combinations of the two kinds of representations may produce more robust predictions and reduce generalization error. To combine the predictions, we first ran a review through both models to produce the log probability of each rating. Then we simply added the log probabilities, and predicted the resulting maximum. This method penalizes ratings that are considered extremely unlikely by either model, yielding results that both models consider plausible.

Unfortunately, we did not have time to systematically test our averaged model.

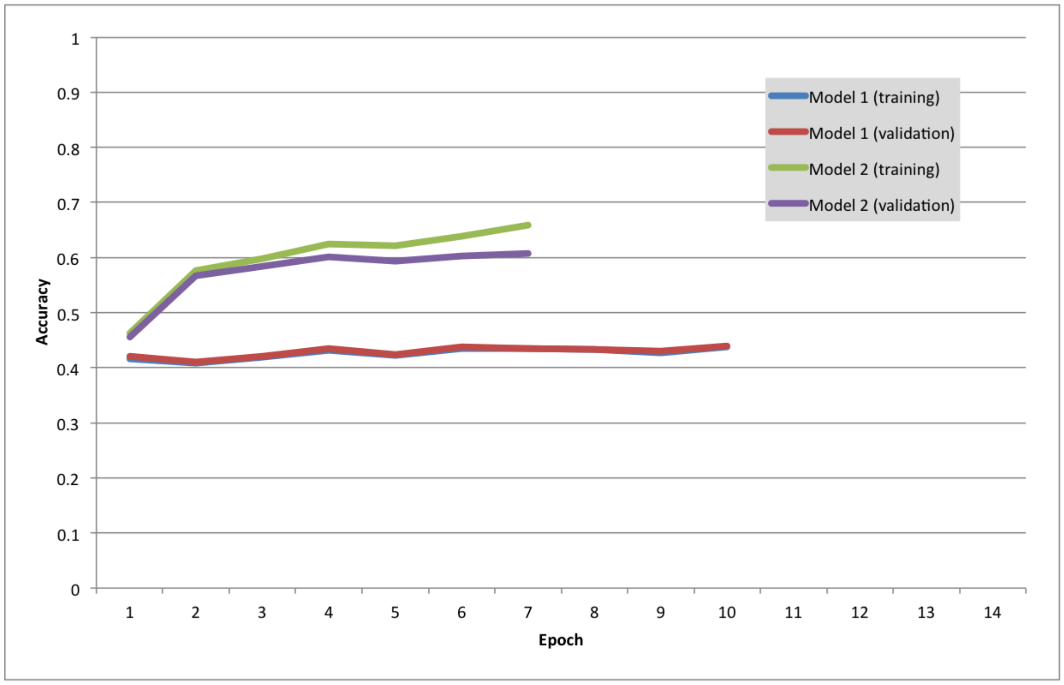

Figure 1: Individual model performance for Models 1 and 2

6 Experiments

These experiments did not help to boost the performance and thus where not included in our final submission.

6.1 Continuous output variable

We explored the idea that a single continuous output variable might be better than the five seperate classes because the star ratings are rank-ordered. For example, if the model predicts high likelihoods for class 2 and 4, then class 3 might actually be its best guess.

```
(6): nn.Linear(500 -> 4)
(7): nn.Sigmoid
(8): nn.Sum
(9): nn.Add

criterion = nn.MSECriterion()
```

Layer 8 is taking the sum of 4 sigmoids which results in a number between 0 and 4 to which layer 9 adds a bias to bring the range close to 1–5. This structure gave better results than a simple unconstrained continuous output

```
(6): nn.Linear(500 -> 1)
```

but did not reach the performance of the 5-classes output that we eventually used for Model 1.

6.2 Phoneme understanding from scratch

We attempted to make a model that was similar to the Crepe [2] model but rather than using characters as the features, we created a vector that used phonemic translations of the English words as features. The idea was that humans typically learn languages through sound so it is possible that the statistical dependencies of sounds would be more informative for learning than the characters themselves. If the phonemes are a nonlinear transformation of the characters that are not in the space of representations that can be learned with the standard point non-linearities and pooling operations that are often used with convolutional nets, a model using phonemes could perhaps learn better than one using just characters. In order to implement this model, we used a standard pronouncing dictionary used for NLP called CMUDict. The details of this dictionary can be found on its website: http://www.speech.cs.cmu.edu/cgi-bin/cmudict. The dictionary uses 39 standard phonemes, with 15 of these being vowels. Each vowel sound had three possible stress markers resulting in 84 possible phonemic characters. We then added a preprocessing step to translate all words in the yelp reviews that were included in the dictionary into their phonetic representations. For words that were not in the dictionary, we used the standard alphabet of lower-case english letters and punctuation markings, adding an additional 69 characters. Reviews are then represented in the same way as in the original Crepe [2] model but with a sparse vector of length 159 for each phoneme/character. While early performance was promising, we could not figure out how to deal with exploding memory/garbage collection issues in converting the reviews to phonemes and thus we could not really assess this model's performance.

References

[1] Pennington, Jeffrey, Socher, Richard, & Manning, Christopher D. (2014). Glove: Global vectors for word representation. Proceedings of the Empirical Methods in Natural Language Processing.

[2] Xiang Zhang and (2015). Text Understanding from Scratch. CoRR, abs/1502.01710, .