Birju Patel

October 4, 2023

CS2731

Homework 2

Part 1:

$\theta_{t+1} = \theta_t - \eta * \nabla L(f(x; \theta), y)$

$\nabla L(f(x; \theta), y) = ((\sigma(w * x) - y) * x_1, (\sigma(w * x) - y) * x_2, \dots , \sigma(w * x) - y)$

Initial Values – $w_1 = 0$, $w_2 = 0$, $b = 0$, $\eta = 0.2$, $\theta_0 = (0, 0, 0)$

Step 1 – $x^{(1)} = (2, 1)$, $y^{(1)} = 1$

$\sigma((0, 0) * (2, 1) + 0) - 1 = -0.5$

$\eta * \nabla = (-0.5 * 0.2 * 2, -0.5 * 0.2 * 1, -0.5 * 0.2) = (-0.2, -0.125, -0.125)$

$\theta_1 = (0, 0, 0) - (-0.2, -0.125, -0.125) = (0.2, 0.125, 0.125)$

Step 2 – $x^{(2)} = (1, 3)$, $y^{(2)} = 0$

$\sigma((0.2, 0.125) * (1, 3) + 0.125) - 0 = 0.668$

$\eta * \nabla = (0.668 * 0.2 * 1, 0.668 * 0.2 * 3, 0.668 * 0.2) = (0.131, 0.401, 0.134)$

$\theta_2 = (0.2, 0.125, 0.125) - (0.131, 0.401, 0.134) = (0.069, -0.276, -0.009)$

Step 3 – $x^{(3)} = (0, 4)$, $y^{(3)} = 0$

$\sigma((0.069, -0.276) * (0, 4) - 0.009) - 0 = 0.247$

$\eta * \nabla = (0.247 * 0.2 * 0, 0.247 * 0.2 * 4, 0.247 * 0.2) = (0, 0.1976, 0.0494)$

$\theta_3 = (0.069, -0.276, -0.009) - (0, 0.1976, 0.0494) = (0.069, -0.4736, -0.0584)$

Final Values – $w_1 = 0.069$, $w_2 = -0.4736$, $b = -0.0584$

The sign of the gradient is determined by the value of y. When y is 1, we add to the weights, and when y is 0, we subtract from the weights.

Part 2:

Linear Regression, Unigram	0	1	Accuracy
Precision	0.688797	0.715415	0.702429
Recall	0.697479	0.707031	0.702429
f1-score	0.693111	0.711198	0.702429

Linear Regression, Unigram	0	1
0	166	72
1	75	181

Linear Regression, Custom	0	1	Accuracy
Precision	0.745902	0.748000	0.746964
Recall	0.742857	0.751004	0.746964
f1-score	0.744376	0.749499	0.746964

Linear Regression, Custom	0	1
0	182	63
1	62	187

Sample False Negatives (Linear Regression, Custom) –

- Don't worry about it. Take it easy, keep the ranting to a minimum and the productive editing to a maximum :) And you're still not convinced you should get an account?

- Is it a directed, acyclic graph? or are cycles allowed?

- Can you please be more specific?? What do u mean by "but it's not a problem as far as the tutorial goes" ??

- Are you looking to round to those specific 12 colors in Google image search, or was that just an example of color rounding? If the latter, how many colors would you ideally like to end up with?

- What type of device will be doing the dropping? ASA, IPTables, WFP, ect?

Sample False Positives (Linear Regression, Custom) -

- Even worse: "The authorized use of this data is limited to informational and educational purposes only, and NOT for operational or commercial purposes." This clearly disqualifies the text for Wikipedia purposes, I'm afraid we'll have to delete the article. Do you know of any similar contributions you've made?

- Thanks! Can you please explain me in a way that a mere CS major can understand, how can I get this filter, and what should this filter, erm, filter?

- You tags are misleading - Algorithm, data structures and in your question you are asking for an answer to the puzzle. what do you want?

- "It's my own program " - then, presumably you have the source code. Debug that?

- Okay, I have built a radar system before, I need one major thing, that I am not sure if you intended to answer. Do you plan to do this at a single frequency?

Top Features in Impolite Comments (Linear Regression, Custom) -

- feature: "whi", coefficient: -5.588388805398528

- feature: "?", coefficient: -2.9595353747448754

Top Features in Polite Comments (Linear Regression, Custom) -

- feature: "thank", coefficient: 5.685109897318477

- feature: "could", coefficient: 3.162892778698414

Neural Network	0	1	Accuracy
Precision	0.616949	0.557789	0.593117
Recall	0.674074	0.495536	0.593117
f1-score	0.644248	0.524823	0.593117

Neural Network	0	1
0	182	88
1	113	111

For the custom feature-based model, I chose to do custom preprocessing, use tf-idf vectorization, and add the length of the comment as a feature. In a comment, the presence of a negative word can totally change the meaning of a sentence. For instance, the phrase “this code is good” and “this code is not good” have completely different meanings. To express this, I used the technique mentioned in the book, where I appended the prefix “NOT_” to every word following a negative word until punctuation was reached. For instance, the previous phrase becomes “this code is not NOT_good.” To accomplish this, I created a CustomVectorizer class that extended the TfidfVectorizer. I modified the tokenizer to stem and negate sentences once they had been tokenized by the vectorizer of the superclass. These modifications did improve the performance in comparison with the basic bag of words classifier. My custom classifier outperformed it by between 2% and 5% in accuracy, precision, and recall.

I was surprised that “?” was a feature often present in impolite comments. Often when I am trying to be polite, I ask a lot of questions so the other person knows I am engaged with what they are saying. I also do not understand why “whi”, which looks like a stemmed version of “which”, is strongly associated with impoliteness. However, it makes sense that “thank” and “could” are associated with politeness.

The basic and modified linear regression classifiers have a similar number of false negatives and false positives. In contrast, the neural network classifier has an unusually high

number of false positives. For the custom linear classifier, it seems that false negatives and false positives are rich in features that are most closely associated with their opposite. For instance, the second false positive in the list contains the word “thanks”, which is typically associated with politeness, while the all the false negatives listed seem to have a large number of “?”, which are associated with impoliteness by the model. I am not sure that a linear model would be able to account for these types of errors. It might be the case that a small number of “?” associates with politeness, and a large amount associates with impoliteness. If we used a non-linear model, we could potentially account for this variation.

To train the neural network, I first had to determine a word embedding scheme. I chose to use the genism “glove-twitter-25” dataset because it was easy to integrate directly into by script and the vector size of 25 was small enough that training of the model was fast. I initially tried word2vec, but their vectors contained 300 features, which caused the model to take too long to train.

I represented the document using a single vector. I took each word in the document and found the GloVe vector that corresponded with it. I then appended this vector to the document vector. When I reached an unrecognized word, I appended a zero vector of length 25. Also, I padded the vectors with zeros so that they were all a standard length. Initially, I took the standard length to be the length of the longest comment in the corpus, but this created too many nodes in the model. Instead, I took the mean and standard deviation of the length of the comments and set the max vector length to the value two standard deviations above the mean. This captured most word data while keeping the neural network relatively small. I used a multi-layer perceptron as the model. I chose this because it is the model we discussed in class. I set the hidden layer node counts by trial and error, modifying, re-training, and then checking the result. Eventually, I decided on a two-layer network with 200 nodes in each layer. This allows training to complete in a reasonable time, while also not sacrificing too much accuracy.