Project1_Report

February 5, 2024

1 Report for CS-165A Coding Project 1: Classifier Agent

1.0.1 Name: Junhwan Lee

1.0.2 PERM #: 5657408

1.0.3 The project was worked on individually.

1.1 Part 1: Gradient Calculations

$$\begin{aligned}
\frac{\partial l}{\partial w} l(w,x,y) &= -y \log\left(\frac{\exp(w^T x)}{1+\exp(w^T x)}\right) - (1-y)\log\left(1 - \frac{\exp(w^T x)}{1+\exp(w^T x)}\right) \\
\frac{\partial l}{\partial w} &= \frac{\partial}{\partial l}(-y\log(p) - (1-y)\log(1-p)) \\
&= -y\frac{1}{p}\frac{\partial p}{\partial w} + (1-y)\frac{1}{1-p}\frac{\partial(1-p)}{\partial w}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial p}{\partial w} &= \frac{\exp(w^T x)(1+\exp(w^T x)) - \exp(w^T x)\exp(w^T x)}{(1+\exp(w^T x))^2}\, x \\
&= p(1-p)x
\end{aligned}$$

$$\begin{aligned}
\frac{\partial(1-p)}{\partial w} &= -\frac{\exp(w^T x)}{(1+\exp(w^T x))^2}\, x \\
&= -p(1-p)x
\end{aligned}$$

Susbtitute,

$$\begin{aligned}
\frac{\partial l}{\partial w} &= -y\frac{1}{p}p(1-p)x + (1-y)\frac{1}{1-p}(-p)(1-p)x \\
&= -y(1-p)x + (1-y)px \\
&= (p-y)x
\end{aligned}$$

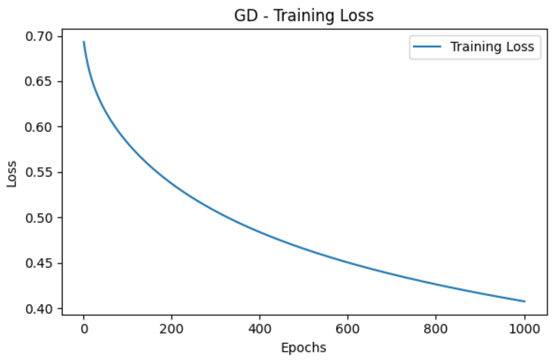
1.2 Part 2: Gradient Descent vs Stochastic Gradient Descent

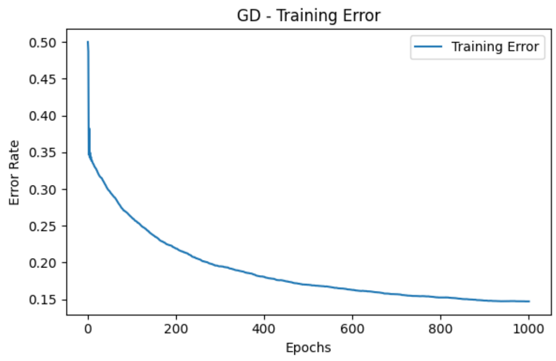
Based on the model in this example, gradient descent took about 17.21 seconds to finish 1000 iterations while stochastic gradient descent took roughly 62.59 seconds for 9 epochs, 10000 iterations. We can observe that stochastic gradient descent is much faster than gradient descent based on the number of iterations.

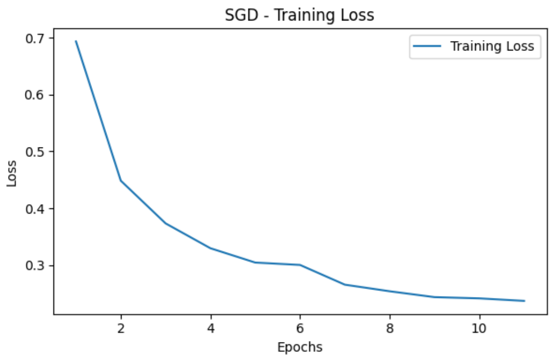
Also, SGD displays much better performance in terms of error. The average test errors are as following:

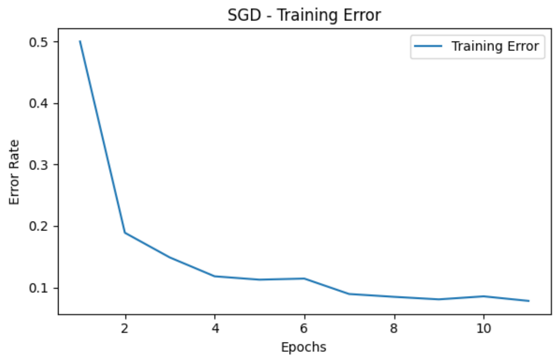
GD: test err = 0.18033333333333335
SGD: test err = 0.16766666666666666

Here is the following plots:









We can observe that SGD displays lower rate in both loss and error

1.3 Part 3: Apply the model to your own text

[ ]:
```
from classifier_solved import load_data,tokenize, compute_word_idf
from classifier_solved import custom_feature_extractor, classifier_agent
import numpy as np

# First load the classifier

with open('data/vocab.txt') as file:
    reading = file.readlines()
    vocab_list = [item.strip() for item in reading]


# By default this is doing the bag of words, change this into your custom␣
↪feature extractor
```

```
# so it works with your "best_model.npy"
feat_map = custom_feature_extractor(vocab_list, tokenize)

d = len(vocab_list)
params = np.array([0.0 for i in range(d)])
custom_classifier = classifier_agent(feat_map, params)
custom_classifier.load_params_from_file('best_model.npy')
```

```
[ ]: # Try it out!

my_sentence = "This movie is amazing! Truly a masterpiece."

my_sentence2 = "The book is really, really good. The movie is just dreadful."

ypred = custom_classifier.predict(my_sentence,RAW_TEXT=True)

ypred2 = custom_classifier.predict(my_sentence2,RAW_TEXT=True)

print(ypred,ypred2)
```

```
[ True] [False]
```

1.3.1 We can also try predicting for each word in the input so as to get a sense of how the classifier arrived at the prediction

```
[ ]: import pandas as pd

# function for set text color of positive
# values in Dataframes
def color_predictions(val):
    eps = 0.02
    if isinstance(val,float):
        if val > eps:
            color = 'blue'
        elif val < -eps:
            color = 'red'
        else:
            color = 'black'
    else:
        color='black'
    return 'color: %s' % color

my_sentence_list = tokenize(my_sentence2)
ypred_per_word = custom_classifier.
 ↪predict(my_sentence_list,RAW_TEXT=True,RETURN_SCORE=True)

df = pd.DataFrame([my_sentence_list,ypred_per_word])
```

```
df.style.applymap(color_predictions)
```

```
[ ]: <pandas.io.formats.style.Styler at 0x10f936430>
```

1.3.2 Answer the questions:

1. Are the above results making intuitive sense and why?
2. What are some limitation of a linear classifier with BoW features?
3. what are some ideas you can come up with to overcome these limitations (i.e., what are your ideas of constructing informative features)?

1.3.3 Answers

1. Some of the words do not have any strong connotations such as "book" and "is".
   However, we can see that those words have strong weights towards negative review which is not necessarliy true.

2. BoW features are not flexible. It lacks semantic understanding.
   BoW features extracts words independently without looking at the whole sentence like transformer models.

3. Instead of representing individual words, we can include sequences of adjacent words in the feature set to extract.
   Use word-embeddings such as Word2Vec to capture semantic relationships between words.

1.4 Part 4: Document what you did for custom feature extractors

Not applicable

1.5 Part 5: Anything else you'd like to write about. Your instructor / TA will read them.

You may answer for instance:

- What have you learned from the experience of working on this coding project?
- Do you think it is easy / hard? If you find it to be hard, what is the main missing piece that you think the instructor / TA should cover in the lectures / discussion sections.
- Have you taken CS165B? Was there a similar project in CS165B? What are the key differences?

I have taken CS165B and CS180. I have some knowledge in computer vision and machine learning. There was a similar project in the deep learning class where we had to use LLMs to predict the following word from given seqeunce of words.