Name: Nebiyou Daniel Hailemariam
Andrew ID: nhailema
Nickname: Gradient Ascent

Machine Learning with Graphs

Homework 2 – Template

1. Statement of Assurance

I certify that all the material that I have submitted is my original work that was done by only me.

2. Data Preprocessing

(1) **[5 pts]** After your finishing the data preprocessing, report the top 9 frequent tokens and corresponding counts in the report.

Rank	Token	Count	Rank	Token	Count	Rank	Token	Count
No. 1	good	717886	No. 2	place	705901	No. 3	food	668636
No. 4	great	568882	No. 5	like	541612	No. 6	just	516881
No. 7	time	432549	No. 8	service	402826	No. 9	really	384510

(2) **[5 pts]** Before continuing to the next step, another interesting problem is to check the star distribution of training samples. Report the count of training samples for each star (i.e., 1 to 5).

Star	1	2	3	4	5
# of training data	128038	112547	178215	373469	463084
Percentage	10.19%	8.96%	14.19%	29.75%	36.88%

Do you find something unexpected from the distribution (e.g., whether the dataset is balanced)? Will this be a problem in training the model? If so, could you give some idea about how to address it and explain why your idea should work?

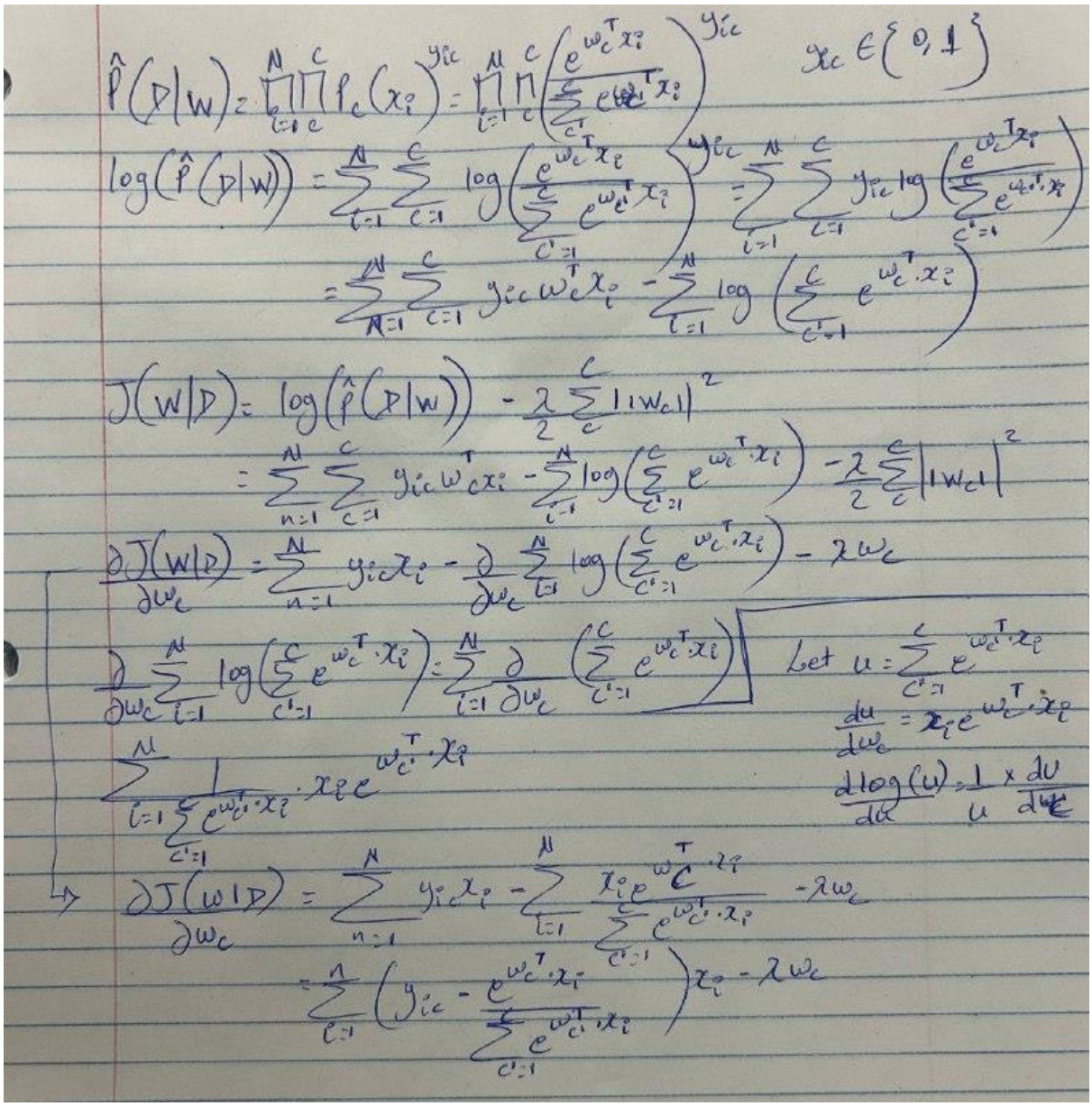
Answer: The distribution of training samples shows that the dataset is not balanced. There is a significant imbalance in the number of samples across different star ratings. This imbalance might pose a challenge during model training, as the model could be biased towards the majority classes (e.g., star ratings 5 and 4). To address this, techniques such as oversampling the minority classes and using class weights can be employed. Oversampling the minority classes helps the model get more instances of those classes, aiding it in learning their features better. Additionally, using class weights helps assign different weights to classes during training to ensure that the model is more penalized for misclassifying the minority class than the majority class.

3. Model Design

(1) **[5 pts]** Show that the gradient of regularized conditional log-likelihood function with respect to the weight vector of class c (i.e., $\frac{\partial l(W)}{\partial w_c}$) is equal to

$$\sum_{i=1}^{n}\left(y_{ic}-\frac{e^{\mathbf{w}_c^T\mathbf{x}_i}}{\sum_{c'=1}^{C}e^{\mathbf{w}_{c'}^T\mathbf{x}_i}}\right)\cdot\mathbf{x}_i-\lambda\mathbf{w_c}$$

Notice that the gradient of log-likelihood function with respect to a vector w_c is itself a vector, whose i-th element is defined as $\frac{\partial l(W)}{\partial w_{ci}}$, where w_{ci} is the i-th element of vector w_c.



(2) **[5 pts]** Let the learning rate be α, outline the algorithm (Batched-SGD) for implementation. You should cover how would you like to update the weights in each iteration, how to check the convergence and stop the algorithm and so on.

Answer: The aim is to maximize the log-likelihood function, which has a maximum value of 0 when our model assigns a probability of 1 to the correct class and a value less than 0 when our model assigns a small value to the correct class. Therefore, after computing the $\nabla \boldsymbol{w}$L, we will get a small change in the direction of the steepest increase. We will need to move in the same direction.

$$\boldsymbol{w}(t) = w(t-1) + \alpha \nabla \boldsymbol{w} L$$

How fast does GD converge?

- Theorem

If ℓ is both convex and differentiable [1]

$$\ell\left(w^{(k)}\right) - \ell\left(w^*\right) \leq \begin{cases} \frac{\|w^{(0)} - w^*\|_2^2}{2\eta k} = O\left(\frac{1}{k}\right) & \ell \text{ is convex} \\ \frac{c^k L \|w^{(0)} - w^*\|_2^2}{2} = O\left(c^k\right) & \ell \text{ is strongly convex} \end{cases} \quad (4)$$

where k is the number of iterations and $c \in (0, 1)$.

- In general, to achieve $l(w^{(k)}) - l(w^*) \leq \rho$, GD needs $O\left(\frac{1}{\rho}\right)$ iterations;
- With strong convexity, it takes $O\left(\log\left(\frac{1}{\rho}\right)\right)$ siterations[2]

2: Convex Optimization, S. Boyd & L. Vandenberghe, Ch 9.3

2/13/2024 @Yiming Yang, lecture on SGD & EVAL 6

We multiply the steps we take with a small value of less than 1. One reason is to take small steps to move past the maximum but rather take a slow and measured step. To check for convergence, we can look at the following theorem shown above. The theorem applies to the convex and strongly convex functions. The negative log-likelihood is a convex function. In this problem, we're trying to maximize the log-likelihood function, which is not convex but rather concave. But we see the similarity between the two approaches. Maximizing the log-likelihood is the same thing as minimizing the negative log-likelihood function. According to the theorem, to achieve a precision of $L(w^{(k)}) - L(w^*) \leqslant \rho$ for a convex function (minimizing the negative log-likelihood), GD needs $O(1/\rho)$ of iterations. Therefore, since minimizing the negative log-likelihood and maximizing the log-likelihood function are comparable, to achieve a precision of $L(w^{(k)}) - L(w^*) \leqslant \rho$, GD needs $O(1/\rho)$ of iterations.

(3) **[10 pts]** After implementing your model, please use these two types of prediction to calculate and report the Accuracy and RMSE (See definition in Evaluation part) on the entire training set with the two features designed in Task 2.

Feature	CTF		DF	
Dataset	Training	Development	Training	Development
Accuracy	54.13%	54.01%	56.26%	55.84%
RMSE	0.8542	0.8557	0.8730	0.8813
Parameters Setting	Learning Rate alpha=0.1? Regularization Parameter lambda=1 How many iterations used? 10 epochs. Batch size=512?			

[10 pts] Multi-class Support Vector Machine

After you figure them out, report only the accuracy on the training and development set using the two features designed in Task 2.

Feature	CTF		DF	
Dataset	Training	Development	Training	Development
Accuracy	57.78%	57.44%	41.91%	37.06%
Parameters Setting	All parameters you used to run SVM. If you run SVM in terminal, include your command line here.			

Answer: I used scikit-learn's SVM module, which implements Liblinear. The SVM model is initialized using l2 penalty, and the loss function is squared hinge loss.

4. **Feature Engineering**

[10 pts] Describe in details your most satisfying design and the corresponding considerations, use formula to illustrate your idea if necessary. Besides, report the evaluation results on training and development set here (The reported result here should match the record on the leaderboard).

Answer:

The Neural Network approach used for sentiment analysis on Yelp reviews involves a three-layer neural network with ReLU activation functions, trained using the Adam optimizer with a learning rate of 0.01, batch size of 32, and CrossEntropyLoss loss function. This approach has demonstrated better performance compared to other traditional machine learning models.

ReLU activation is used to introduce non-linearity. It enhances the model's ability to capture complex relationships within the data. The evaluation results on the development set show an accuracy of 58.93%, outperforming the performance of all other models. Additionally, for soft prediction evaluation on the development set, the root mean squared error (RMSE) was 0.7736, indicating better star ratings estimation.

The model, implemented using PyTorch, is structured as an MLP class. Custom dataset classes, YelpDataset and YelpTestDataset, were defined to load the Yelp reviews dataset

for training. Moreover, the feature used in this sentiment analysis model were **CTF**, which was found to yield far better results compared to **DF**.

5. **One sentence of your feeling for this homework**

   Is that good or not? Why?

   The homework was quite intense for me. I wasn't able to load the dataset in my computer. Students should have been made aware that we would buy a compute.