Name: João Coelho
Andrew ID: jmcoelho
Nickname: joao

Machine Learning for Text Mining

Homework 5 – Template

1. Statement of Assurance

I certify that all of the submitted material is original and done by 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	742807	No. 2	place	716146	No. 3	food	691499
No. 4	great	585143	No. 5	like	546150	No. 6	just	521348
No. 7	time	442451	No. 8	service	431304	No. 9	really	387324

(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	10.20	8.97	14.20	29.75	36.89

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?

The data does not follow an uniform distribution between classes, showing biases towards the higher ratings of 4* and 5*. It *may* be problematic if the data used to test the model follows a much different distribution. To assess that I'll leverage a separate validation set during training, and if necessary, employ techniques such as under or over sampling to overcome the imbalance.

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.

Starting by simplifying the likelihood expression:

$$\begin{aligned}
l(W) &= \sum_{i=1}^{n}\log P(y_i|x_i,W)-\frac{\lambda}{2}\sum_{i=1}^{C}||w_i||^2\\
&= \sum_{i=1}^{n}\log\prod_{c=1}^{C}\left(\frac{e^{w_c^\top x_i}}{\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}}\right)^{y_{ic}}-\frac{\lambda}{2}\sum_{i=1}^{C}||w_i||^2\\
&= \sum_{i=1}^{n}\sum_{c=1}^{C}\log\left(\frac{e^{w_c^\top x_i}}{\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}}\right)^{y_{ic}}-\frac{\lambda}{2}\sum_{i=1}^{C}||w_i||^2\\
&= \sum_{i=1}^{n}\sum_{c=1}^{C}y_{ic}\log\left(\frac{e^{w_c^\top x_i}}{\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}}\right)-\frac{\lambda}{2}\sum_{i=1}^{C}||w_i||^2\\
&= \sum_{i=1}^{n}\sum_{c=1}^{C}y_{ic}\left(\log\left(e^{w_c^\top x_i}\right)-\log\left(\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}\right)\right)-\frac{\lambda}{2}\sum_{i=1}^{C}||w_i||^2\\
&= \sum_{i=1}^{n}\sum_{c=1}^{C}y_{ic}\left(w_c^\top x_i-\log\left(\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}\right)\right)-\frac{\lambda}{2}\sum_{i=1}^{C}||w_i||^2\\
&= \sum_{i=1}^{n}\left(w_{c_i}^\top x_i-\log\left(\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}\right)\right)-\frac{\lambda}{2}\sum_{i=1}^{C}||w_i||^2
\end{aligned}$$

The last step comes from the fact that y_{ic} is 1 for a single $c=c_i$ and 0 otherwise. Now computing the gradient, let's leverage that same fact, notating $\mathbb{1}$ as the indicator function:

$$\begin{aligned}
\nabla_{w_c}l(W) &= \sum_{i=1}^{n}\left(\nabla_{w_c}(w_{c_i}^\top x_i)-\nabla_{w_c}\log\left(\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}\right)\right)-\frac{\lambda}{2}\sum_{i=1}^{C}\nabla_{w_c}||w_i||^2\\
&= \sum_{i=1}^{n}\left(x_i\mathbb{1}(c=c_i)-\frac{\nabla_{w_c}\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}}{\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}}\right)-\lambda w_c\\
&= \sum_{i=1}^{n}\left(x_i\mathbb{1}(c=c_i)-x_i\frac{e^{w_c^\top x_i}}{\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}}\right)-\lambda w_c\\
&= \sum_{i=1}^{n}\left(\mathbb{1}(c=c_i)-\frac{e^{w_c^\top x_i}}{\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}}\right)x_i-\lambda w_c\\
&= \sum_{i=1}^{n}\left(y_{ic}-\frac{e^{w_c^\top x_i}}{\sum_{c'=1}^{C}e^{w_{c'}^\top x_i}}\right)x_i-\lambda w_c
\end{aligned}$$ ■

This finalizes the proof, leveraging the fact that, by definition, if $c=c_i$ then $y_{ic}=1$, and if $c\neq c_i$ then $y_{ic}=0$.

(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.

Batched Stochastic Gradient Ascent:

Input: X, y, N, w, α, bs

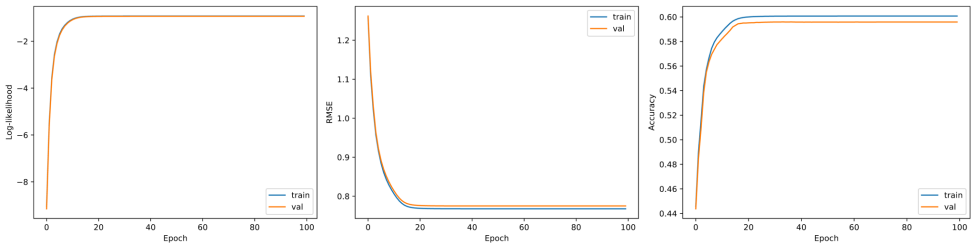
Initialization: w~Gaussian(0,1); Split (X, y) in batches of size bs

For N epochs:

For each batch (Xi, yi):

$w \leftarrow w + \alpha \nabla f(Xi, yi; w)$

Gradient ascent is used because we want to maximize the log-likelihood. Model trains for a fixed set of epochs (N, default 100). Model is validated on a subset of the provided train data every epoch. Checkpoint with best accuracy on that subset is kept as final model (which was usually the last one). Convergence is verified by looking at train/val curves for likelihood, accuracy, and rmse. Early stop was not implemented since this was already running very fast (~1s/epoch).



(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	0.5987	0.5945	0.5986	0.5946
RMSE	0.7919	0.7942	0.7921	0.7945
Parameters Setting	Learning Rate alpha=0.5 Regularization Parameter lambda=0.01 How many iterations used? 100. Batch size=5000			

[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	0.5913	0.5867	0.5918	0.5874
Parameters Setting	I used the python bindings. Only set parameters were "-s 2", i.e., L2-regularized L2-loss support vector classification (primal), and "-q" to suppress outputs to stdout. Everything else was default.			

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).

I build on top of the DF experiment, picking the top-5000 words with higher df to form the vocabulary. However, I change the weighting scheme from term frequency to BM25-based weighting. Given the document d for which we want to compute the 5000-dimensional vector (w_1, w_2, …, w_{5000}), each w_i is computed as follows:

$$w_i = w_{\text{BM25}}(t_i, d) = \text{idf}(t_i) \times \frac{\text{tf}(t_i, d) \times (k_1 + 1)}{\text{tf}(t_i, d) + k_1 \times \left(1 - b + b \times \frac{|d|}{\text{avgdl}}\right)}$$

In the previous equation, t_i is the term associated with the ith dimension of the vector. Tf and idf are the term frequency and inverse document frequency metrics, respectively. K1 and b are hyperparameters set to 1.2 and 0.75. This metric considers document length normalization by leveraging the ratio between |d|, the number of tokens of the document, and avgdl, the average document length in the collection.

This metric is simple to implement, does not imply significant time increase when compared to the previous approaches, and improves the previous results, with a larger impact on the soft metric:

Feature	DF + BM25	
Dataset	Training	Development
Accuracy	0.6176	0.6076
RMSE	0.7251	0.7327

5. **One sentence of your feeling for this homework**
   Is that good or not? Why?

   It was nice. Although I did not explore it in the last exercise, there is a strong alignment between this HW with the previous ones. E.g., apply matrix decomposition from HW2 to the document/term matrix; or use the LSTM/CNN from HW3 directly on top of the text, to compare feature extraction vs representation learning.