Exercises
Computational Intelligence Lab
SS 2017

Machine Learning Institute
Dept. of Computer Science, ETH Zürich
Prof. Dr. Thomas Hofmann
Web http://cil.inf.ethz.ch/

Series 6, April 6-7, 2017
(Word Embeddings and Text Classification)

Problem 1 (Theory of Word Embeddings):

Recall the GloVe objective that consists in weighted least squares fit of log-counts, written as

$$\mathcal{H}(\theta;\mathbf{N}) = \sum_{i,j} f(n_{ij}) \Big(\underbrace{\log n_{ij}}_{\text{target}} - \underbrace{\langle \mathbf{x}_i, \mathbf{y}_j \rangle}_{\substack{=\log \tilde{p}_\theta(w_i|w_j) \\ \text{model}}} \Big)^2 ,$$

where $\tilde{p}_\theta(w_i|w_j) = \exp \langle \mathbf{x}_i, \mathbf{y}_j \rangle$ (note that we here ignore the bias terms for simplicity).

1) Assume $f(.) := 1$ for all arguments, and write $m_{ij} := \log n_{ij}$.
   a) Derive the derivative (gradient) of $\mathcal{H}$ with respect to the embedding vector $\mathbf{x}_i$, and the same for $\mathbf{y}_j$.
   b) Derive a stochastic gradient of $\mathcal{H}$, given by the contribution of just the term for (i,j) as part of the sum, again with respect to the embedding vector $\mathbf{x}_i$, and the same for $\mathbf{y}_j$.
   A *stochastic gradient* of a function is a vector which in expectation is the true gradient. For a function of sum structure $g = \frac{1}{|K|} \sum_{k \in K} g_k$, a stochastic gradient is given by the gradient of an element of the sum, i.e., ∇g_k, so that we have $\mathbf{E}_k[\nabla g_k] = \nabla g$.
2) Show that GloVe with $f(n_{ij}) := \begin{cases} 1 & \text{if } n_{ij} > 0, \\ 0 & \text{otherwise.} \end{cases}$ solves a *matrix completion* problem
$$\min_{\mathbf{X},\mathbf{Y}} \sum_{ij\,:\,n_{ij}>0} \big(m_{ij} - (\mathbf{X}^\top \mathbf{Y})_{ij} \big)^2 \ .$$
3) Analogous to part 1), derive the gradient as well as stochastic gradient of $\mathcal{H}$ with respect to $\mathbf{x}_i$ and $\mathbf{y}_i$ for any weighting function $f()$.

Problem 2 (Project Text Sentiment Classification):

In this assignment, we will use word embeddings as one possible approach to build our own text classifier system. The text sentiment classification task is the second one of the three possible project tasks, and we introduce the task and competition format here.

The task of the competition is to predict if a tweet message used to contain a positive :) or negative :(smiley, by considering only the remaining text.

Submission system environment setup:

1. The data and template code for the competition are available here:

   `https://github.com/dalab/lecture_cil_public/tree/master/exercises/ex6` .

   The kaggle web page for the text sentiment classification competition can be found here. To join the competition, create a kaggle account using your `....ethz.ch` email address.

   https://inclass.kaggle.com/c/cil-text-classification-2017.

2. Download the provided dataset `train_pos.txt`, `train_neg.txt` and `test_data.txt`, as well as the example `sampleSubmission.csv`, from the competition webpage.

To submit your solution to the online evaluation system, we require you to prepare a ".csv" file of the same structure as `sampleSubmission.csv` (the order of the predictions does not matter, but make sure the tweet ids and predictions match). Your submission is evaluated according to the classification error (number of misclassified tweets) of your predictions.

Working with Twitter data: We provide a large set of training tweets, one tweet per line. All tweets in the file `train_pos.txt` (and the `train_pos_full.txt` counterpart) used to have positive smileys, those of `train_neg.txt` used to have a negative smiley. Additionally, the file `test_data.txt` contains 10'000 tweets without any labels, each line numbered by the tweet-id.

Your task is to predict the labels of these tweets, and upload the predictions to kaggle. Your submission file for the 10'000 tweets must be of the form `<tweet-id>, <prediction>`, see `sampleSubmission.csv`.

Note that all tweets have already been pre-processed so that all words (tokens) are separated by a single whitespace. Also, the smileys (labels) have been removed.

Classification using Word-Vectors. For building a good text-classifier it is crucial to have a good feature representation of the input text. Here we will start by using the word vectors (word embeddings) of each word in the given tweet. For simplicity, we will construct the feature representation of the entire text by simply averaging the word vectors.

Below is a solution pipeline with an evaluation step:

1. **Compute Word Embeddings:** Load the training tweets given in `train_pos_full.txt` and `train_neg_full.txt` (or a suitable subset depending on RAM requirements), and construct a a vocabulary list of words appearing at least 5 times.

   Compute GloVe word embeddings for each word of your vocabulary. (See previous exercise).

2. **Construct Features for the Training Texts:** Load the training tweets and the built GloVe word embeddings. Using the word embeddings, construct a feature representation of each training tweet (by averaging the word vectors over all words of the tweet).

3. **Train a Linear Classifier:** Train a linear classifier (e.g. logistic regression or SVM) on your constructed features, using the `scikit learn` library. Recall that the labels indicate if a tweet used to contain a :) or :(smiley.

4. **Prediction:** Predict labels for all tweets in the test set.

5. **Submission / Evaluation:** Submit your predictions to kaggle, and verify the obtained misclassification error score. (You can also use a local separate validation set to get faster feedback on the accuracy of your system). Try to tune your system for best evaluation score.

Extensions: Naturally there are many ways to improve your solution, both in terms of accuracy and computation speed. More advanced techniques can be found in the recent literature.

Problem 3 (GloVe Implementation):

In this part you will implement the GloVe models and train your own word vectors with stochastic gradient descent (SGD). We provide a set of tokenized tweets for which you have to learn word embeddings.

1) Construct the co-occurrence matrix $\mathbf{N} = (n_{ij}) \in \Re^{|\mathcal{V}| \cdot |\mathcal{C}|}$ where n_{ij} is the number of occurrences of the word $w_i \in \mathcal{V}$ in context of $w_j \in \mathcal{C}$.

2) Implement the GloVe algorithm using Stochastic Gradient Descent. To do so, you can fill in the implementation of the cost and gradient functions in

   `https://github.com/dalab/lecture_cil_public/tree/master/exercises/ex6` .

   When you are done, test your implementation by running on the data provided in the Text Sentiment Classification task, and tune the best-step-size parameters and dimensionality in order to get good performance.

3) The word vectors typically capture strong linguistic regularities, for example vector operations on the embeddings $w_{Paris} - w_{France} + w_{Italy}$ results in a vector that is very close to w_{Rome}, and $w_{king} - w_{man} + w_{woman}$ is close to w_{queen}. Your task is to use a) find some similar interesting configurations of embedded words from tweets, and b) Perform PCA to project the word vectors to a 2-dimensional space, and check what linguistic features are learned for the word embeddings.