Exercises
Computational Intelligence Lab
SS 2019

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

Series 6, March 28-30, 2019 (Word Embeddings and Text Classification)

1 Theoretical Questions

Problem 1: Stochastic Gradient Descent (SGD)
We consider minimization of function f that is an average of functions $\{f_1, \ldots, f_n\}$ as

$$f(\mathbf{w}) = \frac{1}{n} \sum_{i=1} f_i(\mathbf{w}). \tag{1}$$

Stochastic gradient descent (SGD) optimizes f through the following recurrence:

$$\mathbf{w} \leftarrow \mathbf{w} - \gamma \nabla f_r(\mathbf{w}), \tag{2}$$

where r is selected uniformly at random from $\{1, \ldots, n\}$ (independently from $\mathbf{w}$).

i. Prove that $\nabla f_r(\mathbf{w})$ is an unbiased estimator of $\nabla f(\mathbf{w})$, namely

$$\mathbf{E}_r \left[\nabla f_r(\mathbf{w})\right] = \nabla f(\mathbf{w}) \tag{3}$$

holds. Compare the computational complexity of one gradient descent step with one SGD step.

ii. Suppose f is L-smooth, i.e. for all $\mathbf{v}$ and $\mathbf{w}$ the following holds:

$$f(\mathbf{w}) \leq f(\mathbf{v}) + \langle \nabla f(\mathbf{v}), \mathbf{w} - \mathbf{v} \rangle + \frac{L}{2} \|\mathbf{w} - \mathbf{v}\|^2 \tag{4}$$

Using the above inequality and the result from section i. prove that the following holds

$$\mathbf{E}_r \left[f(\mathbf{w} - \gamma \nabla f_r(\mathbf{w}))\right] \leq f(\mathbf{w}) - \gamma \|\nabla f(\mathbf{w})\|^2 + \frac{\gamma^2 L}{2} \mathbf{E}_r \left[\|\nabla f_r(\mathbf{w})\|^2\right] \tag{5}$$

iii. Given the result of Eq. (5) in section ii., explain why SGD requires small stepsize to guarantee a decrease in $f(\mathbf{w})$ in expectation.

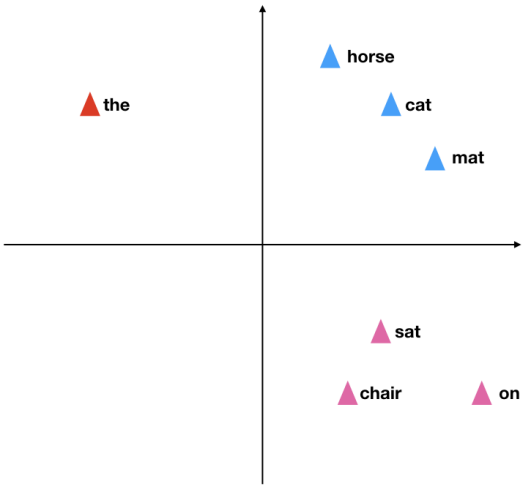
Problem 2: Latent Vector Model
Consider the following log bilinear model for word embedding:

$$\log(p(w|w')) = \langle \mathbf{x}_w, \mathbf{x}_{w'} \rangle + \beta \tag{6}$$

where $p(w|w')$ is the probability that w occurs in context window w'.

i. Suppose w' is an adjective and w is a noun. Explain why we do expect that $p(w|w') > p(w'|w)$ for context window of size 1. Why the model of Eq. (6) can not capture this? How we can refine the model to solve this issue?

ii. Consider the following 2D embeddings of words (i.e. $\mathbf{x}_w$ in Eq. (6)).



Given the above embeddings, we want to complete the following sentence using one of words "horse", "mat", or "chair".

The cat sat on the ...

According to the skip-gram model with window size 3 and 4, which words are more probable to complete the sentence?

Window size	predicted word (horse, mat, or chair)
3	.....
4	.....

Problem 3: Negative Sampling

Recall $\mathbf{z}_w$ and $\mathbf{x}_v$ denote the embedding of the context w and the word v, respectively. We consider the following model the probability of observing v given the context of w, namely

$$\log P(v|w; \mathbf{x}, \mathbf{z}) = \langle \mathbf{x}_v, \mathbf{z}_w \rangle + c \tag{7}$$

using embedding $\mathbf{z}_w$ and $\mathbf{x}_v$. To Suppose that we are given sample set Δ_+ is generated from an unknown distribution p.

i. **Inference issue** To infer the embeddings using maximum likelihood approach, we need to the following minimization problem:

$$\arg\min_{\mathbf{x},\mathbf{z}} \left(-\log L = -\sum_v \left(\sum_{w \in C(v)} \langle \mathbf{x}_v, \mathbf{z}_w \rangle - \log \left(\sum_{w \in C(v)} \exp(\langle \mathbf{x}_v, \mathbf{z}_w \rangle) \right) \right) \right) \tag{8}$$

where $C(v)$ denote the set of context words of v. Explain why SGD on the above objective is not practical?

ii. **Noise contrastive model** We generate noises from an arbitrary distribution p_n (we called it pdf. of negative samples) . Hence we can classify our samples to two sets: the set of positive samples Δ_+ generated from p and the set of negative samples Δ_- from p_n. Let $D_{v,w} \in \{0, 1\}$ be a random variable where $D_{v,w} = 1$ if pair (v, w) is generated from p, otherwise it is zero. Given embeddings, we define

$$P(D_{v,w} = 1|v, w) = \sigma(\langle \mathbf{x}_w, \mathbf{z}_w \rangle), \quad \sigma(\alpha) = 1/(1 + \exp(-\alpha)). \tag{9}$$

Prove that the above reconstructs the bi-linear model as

$$\log P(v|D_{v,w} = 1, w) = \langle \mathbf{x}_v, \mathbf{z}_w \rangle + c. \tag{10}$$

iii. **Derivation of the likelihood and log-likelihood function:** Using the conditional probability distribution, given in the last part, derive the log-likelihood for positive and negative samples as

$$L(\Delta_+, \Delta_-; \mathbf{x}, \mathbf{z}) = \sum_{(v,w)\in\Delta_+} \log \sigma(\mathbf{x}_v^\top \mathbf{z}_w) + \sum_{(v,w)\in\Delta_-} \log(\sigma(-\mathbf{x}_v^\top \mathbf{z}_w)). \tag{11}$$

iv. **Optimization:** Compare the plain log-likelihood in Eq.(8) with those of Eq. (11) for negative sampling. Explain which is easier to optimize. **Hint:** For which objective function SGD is applicable?

Problem 4: GloVe and SGD

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

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

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

1) Assume $f(\cdot) \equiv 1$ 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 $\mathbf{y}_j$.

2) Show that GloVe with $f(n_{ij}) := \begin{cases} 1 & \text{if } n_{ij} > 0, \\ 0 & \text{otherwise,} \end{cases}$ solves the *matrix completion* problem

$$\min_{\mathbf{X},\mathbf{Y}} \sum_{(i,j)\,:\,n_{ij}>0} \big(m_{ij} - (\mathbf{X}^\top \mathbf{Y})_{ij}\big)^2.$$

3) Analogous to part 1), derive the gradient and the stochastic gradient of $\mathcal{H}$ with respect to $\mathbf{x}_i$ and $\mathbf{y}_i$ for a generic weighting function f.

2 Practical Questions

Problem 5: 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 of the three project tasks proposed. Here, we introduce the problem and competition rules.

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

Submission system environment setup:

1. The data and the template code are available at

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

   The kaggle web page for the 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-2019.

2. Download the provided dataset `train_pos.txt`, `train_neg.txt`, `test_data.txt`, and the submission example `sampleSubmission.csv` from the competition web page.

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 predictions is arbitrary, but make sure the tweet ids and predictions match. Your submission is evaluated according to the classification error of your predictions (the number of misclassified tweets).

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

Below is an example of the solution pipeline:

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 the RAM available), and construct a vocabulary, the list of words appearing at least 5 times in the subset chosen.

   Compute the GloVe word embeddings for each word from the 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 from the tweet.

3. **Train a linear classifier:** Train a linear classifier (e.g. logistic regression or SVM) on your features constructed, 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 quality of your solution). Try to tune your system to achieve the 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 6: GloVe Implementation
In this part, you will implement the GloVe models and train your own word vectors with stochastic gradient descent. We provide a set of tokenized tweets for which you have to learn the 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 the 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` .

   Once 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 to achieve a 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 as follows:

a) find some similar interesting configurations of embedded words from tweets,

b) perform PCA to project the word vectors to a 2-dimensional space, and check whether the linguistic features are learned for the word embeddings.