COMP0090 19/20 Assignment 2: "Half-bag"

Pontus SAITO STENETORP
p.stenetorp@cs.ucl.ac.uk

1 Instructions

For this assignment the maximum number of points obtainable is 100, which maps one-to-one for the mark given for the assignment. The assignment is to be carried out in groups of *up to five students*, which may or may not be the same groups as those for past and future assignments. For the assignment you will use Colaboratory[1] and any programming language or framework of your choice. However, boiler plate code and supervision will only be provided for the Julia programming language.[2] The assignment is *due by Friday, December 13th 2019 at 12:00 (UTC+00:00)*.

Your submission *shall*:

1. Take the form of *a single cover page* PDF document, to which you will *append* a printout of your Colaboratory notebooks (you can achieve this using for example PDFtk[3]). The final document is to be submitted via Moodle.
2. List all group members clearly on the cover page and provide their UCL e-mails.
3. Provide hyperlinks on the cover page to *up to five* Colaboratory notebooks, that *must not* be edited after the submission deadline (it may be wise to make a separate copy of your final notebooks and name them something akin to "Submission for COMP0090 Assignment 2").
4. Describe briefly on the cover page (about a paragraph) which group member contributed to which part of the assignment. It is expected that each group member contributes towards at least one of the tasks.

Kindly report errors (even typos) and ask for clarifications when needed, this assignment is to be an exercise in deep learning, not mind reading. Corrections to this assignment post-release will be listed in Section 4.

[1] `https://colab.research.google.com`
[2] `https://julialang.org`
[3] `https://www.pdflabs.com/tools/pdftk-the-pdf-toolkit`

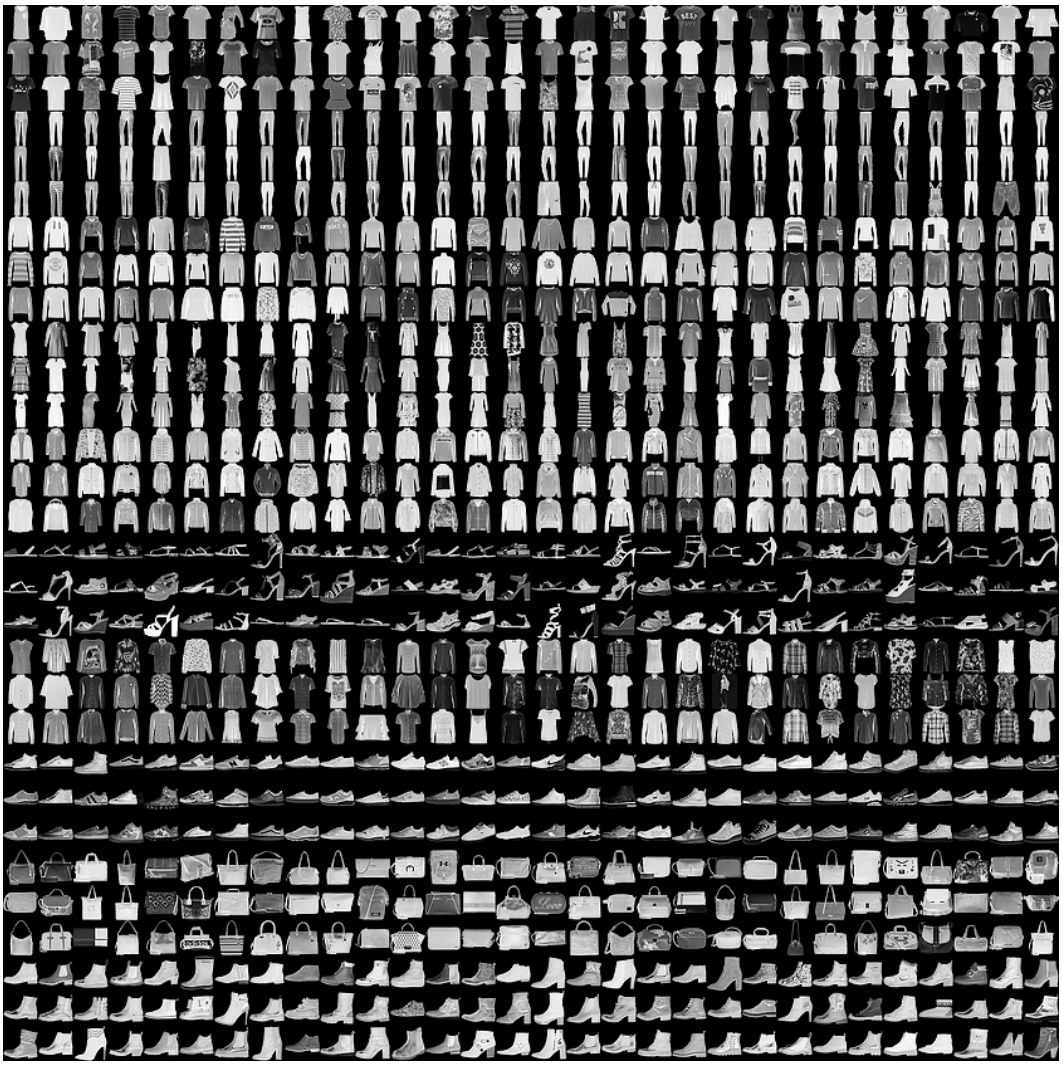

Figure 1: Examples from each of ten classes from Xiao et al. (2017), each class occupies three rows.

2 Data

For this assignment we will make use of two datasets, one of which you should be familiar with.

2.1 Fashion-MNIST

A standard dataset in machine learning is the MNIST dataset of handwritten digits that has for many years been used to train and evaluate machine learning algorithms. However, even simple algorithms can perform relatively well on MNIST and its usage has increasingly come into question. Zalando[4] is a German e-commerce company and that has released a dataset "Fashion-MNIST" (Xiao et al., 2017) which instead of digits from 0 to 9, contains 28-by-28 pixel, greyscale images of ten distinct fashion items (see Figure 1 for an example subset).

Code for loading the data can be found in the Colaboratory notebook for the assignment: `https://colab.research.google.com/drive/1_9ly0mVfbnU-hViXVdck4xk8yv2MAYAP`.

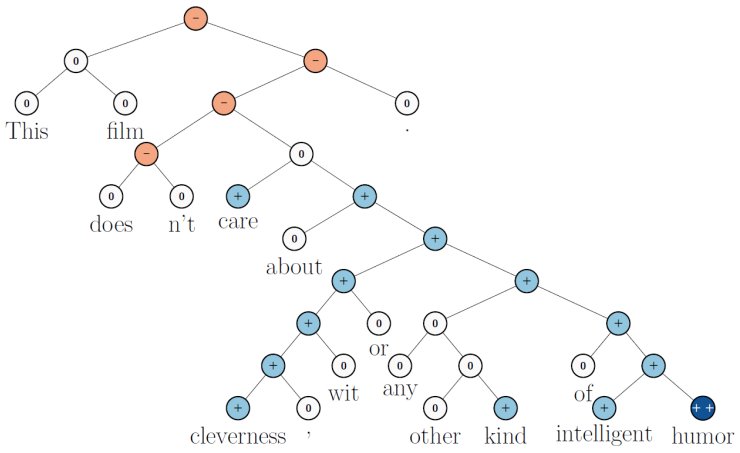

Figure 2: An example of negation from Socher et al. (2013).

2.2 Stanford Sentiment Treebank

A common task in natural language processing is sentiment classification, where given a sentence or a document you are tasked with predicting the overall sentiment of the input. Understandably, this is a fairly popular task in terms of industrial applications of natural language processing, after all, who would not like to know the overall emotional impact of a recent marketing campaign or what the voting population is currently feeling regarding a given topic?

Socher et al. (2013) introduced the Stanford Sentiment Treebank, which not only provides sentiment labels for sentences as a whole, but also for each node inside of a constituency parse tree,[5] providing annotations for phenomena such as negation (as seen in Figure 2). However, for this assignment we will only be concerned with sentence-level annotations, as parse trees and the likes are more suitable for a natural language processing class.

Code for loading the data can be found in the Colaboratory notebook for the assignment: `https://colab.research.google.com/drive/1_9ly0mVfbnU-hViXVdck4xk8yv2MAYAP`.

[4] `https://www.zalando.de`

[5] `https://en.wikipedia.org/wiki/Phrase_structure_grammar`

3 Tasks

3.1 A multi-class, multi-layer perceptron *(20 points)*

Now that we have moved from binary to multi-class classification, it is time to take on the Fashion-MNIST dataset in its true form.

To complete this task you are expected to:

1. Implement a multi-class, multi-layer perceptron with cross-entropy loss for the Fashion-MNIST data.
2. Explore model variants with different layer sizes, depths, etc. to find what works well on the validation set and describe the process through which you arrive at your final model.
3. Train your final model to convergence on the training set using an optimisation algorithm of your choice.
4. Provide a plot of the loss on the training set and validation set for each epoch of training.
5. Provide the final accuracy on the training, validation, and test set.
6. Analyse the errors of your models by constructing a confusion matrix.[6] Which classes are easily "confused" by the model? Hypothesise why.

[6] `https://en.wikipedia.org/wiki/Confusion_matrix`

3.2 Denoising autoencoder *(20 points)*

A denoising autoencoder is only a minor modification of a "vanilla" autoencoder. However, as opposed to simply reconstructing its input $\mathbf{x}$, it seeks to reconstruct it after a noise function has corrupted it. Doing so allows the model to learn to recover from the noise introduced by the noise function – or, "denoising".

$$f(\text{noise}(\mathbf{x})) = \mathbf{x}' \tag{1}$$

For this task we will use a noise function that removes one or two quadrants of the image, an implementation of it can be found in the Colaboratory notebook for the assignment: `https://colab.research.google.com/drive/1_9ly0mVfbnU-hViXVdck4xk8yv2MAYAP`.

To complete this task you are expected to:

1. Implement a denosing autoencoder with mean squared error loss for the Fashion-MNIST data.
2. Explore model variants with different layer sizes, depths, etc. to find what works well on the validation set and describe the process through which you arrive at your final model.
3. Train your final model to convergence on the training set using an optimisation algorithm of your choice.
4. Provide a plot of the loss on the training set and validation set for each epoch of training.
5. Provide a selection of 32 images in their original, "noisy", and "denoised" form from the test set and comment on what the model has been able to learn and what it appears to find challenging.

3.3 Sequence prediction *(20 points)*

A *language model*, predicts the next output in a sequence given a number of initial observations. Recurrent neural networks have been popular for language modelling for some time and for this task and we will use one to predict the next *character* in order to keep the output space significantly smaller than if we were to predict the next word.[7]

To complete this task you are expected to:

1. Implement a character-level recurrent neural network model with a cross-entropy loss for the textual portion of the Stanford Sentiment Treebank data, where the model is to predict the next character based on the character previously observed for a given sentence.
2. Explore model variants with different recurrent units ("vanilla", LSTM, and GRU), number of layers, different depths and layer sizes for the multi-layer perceptron, etc. to find what works well on the validation set and describe the process through which you arrive at your final model.
3. Train your final model to convergence on the training set using an optimisation algorithm of your choice.
4. Provide a plot of the loss on the training set and validation set for each epoch of training.
5. Provide the final accuracy on the training, validation, and test set.
6. Provide 5 sentences completed by your final model, after you have provided a first few words. For these generated sentences, please mark which part was provided by you and which part was generated by the model.

[7] Recall the hirerchical SoftMax and friends from the lectures, these address a large output space.

3.4 Bag of vectors *(20 points)*

A common baseline before the advent of deep learning in natural language processing, was a "bag of words" model where one would take an input of a sequence of words and create a one-hot vector of the size of all words in the dataset and mark the presence of a word in the input by binary setting an index associated with that word in the vector. This accomplishes the task of turning a variable-length input such as a sentence, into a fixed-size input that is easier to process using for example a logistic-regression layer. For many years this proved to be a surprisingly – sometimes even frustratingly – strong baseline for many tasks, despite the fact that it creates a model that is oblivious to the order of the words in the input.

Nowadays it is more common to use a "neuralised" version of this model, where instead of creating a binary vector, each word is first mapped to a fixed-size embedding – commonly these embeddings are pre-trained, using a method such as "word2vec". Then, one or several pooling operations is applied to the embeddings, to turn them into a fixed-size representation. Popular choices for pooling include element-wise mean, max, and min. If more than one pooling operation is used, the resulting vector for each pooling operation is concatenated, this representation is then passed to a multi-layer perceptron to provide a classification decision. Just like its ancestor the "bag of words", this "bag of vectors" can be surprisingly difficult to beat, despite it also being completely ignorant of the word order in its input.

To complete this task you are expected to:

1. Implement a bag of vectors model with a cross-entropy loss for the Stanford Sentiment Treebank data, based on the pre-trained word2vec embeddings provided in the Assignment Colaboratory notebook.
2. Explore model variants with different pooling operations, keeping the word embeddings fixed during training or updating them, different depths and layer sizes for the multi-layer perceptron, etc. to find what works well on the validation set and describe the process through which you arrive at your final model.
3. Train your final model to convergence on the training set using an optimisation algorithm of your choice.
4. Provide a plot of the loss on the training set and validation set for each epoch of training.
5. Provide the final accuracy on the training, validation, and test set.
6. Provide a selection of the classification decisions of the final model for 5 opinionated sentences from movies reviews that you find online – perhaps for a movie you liked, or did not like? Remember to provide a links to the reviews from which you selected the sentences.

3.5 Sequence encoding *(20 points)*

Our "bag"-based model from Section 3.4 ignores word order and it is time to remedy this. Recurrent neural network models are by their very nature capable of keeping a state based on previous observation, and can thus "encode" a variable-length input sequence into a fixed-size output to then be passed to a downstream layer for a classification decision. This downstream layer would preferably be some form of multi-layer perceptron and you can turn the sequence of input words into fixed-size vectors using the code provided for Section 3.4.

To complete this task you are expected to:

1. Implement a recurrent neural network model which encodes a sequence of words into inputs to a multi-layer perceptron with a cross-entropy loss for the Stanford Sentiment Treebank data.
2. Explore model variants with different recurrent units ("vanilla", LSTM, and GRU), number of layers, keeping the word embeddings fixed during training or updating them, different depths and layer sizes for the multi-layer perceptron, etc. to find what works well on the validation set and describe the process through which you arrive at your final model.
3. Train your final model to convergence on the training set using an optimisation algorithm of your choice.
4. Provide a plot of the loss on the training set and validation set for each epoch of training.
5. Provide the final accuracy on the training, validation, and test set.
6. Provide a selection of the classification decisions of the final model for 5 opinionated sentences from movies reviews that you find online[8] – perhaps for a movie you liked, or did not like? Remember to provide a links to the reviews from which you selected the sentences.

[8] These can be the same as those used for Section 3.4.

4 Errata

Empty, for now…

References

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017.