COMP9418 Assignment 3

Advanced Topics in Statistical Machine Learning, 18s2, UNSW Sydney

Last Update: Friday 12th October, 2018 at 15:41

Submission deadline: Sunday October 28th, 2018 at 23:59:59

Late Submission Policy:
20% marks will be deducted from the total for each day late, up to a total of four days. If five or more days late, a zero mark will be given.

Form of Submission: You should submit your solution with the following files:

1. `solution.pdf`: Technical report;
2. `solution.py`: Python code; and
3. `predictions.txt`: Model's prediction on the test data.

No other formats will be accepted. There is a maximum file size cap of 20MB so make sure your submission does not exceed this size.

Submit your files using give. On a CSE Linux machine, type the following on the command-line:
`$ give cs9418 ass2 solution.pdf solution.py predictions.txt`
Alternative, you can submit your solution via the course website
https://webcms3.cse.unsw.edu.au/COMP9418/18s2/resources/21405

Please note that this is a group assignment. See §6 below for details.

Recall the guidance regarding plagiarism in the course introduction: this applies to this homework and if evidence of plagiarism is detected it may result in penalties ranging from loss of marks to suspension.

[100 Marks] Structured Probabilistic Models

In this assignment you will make use of the dataset from the *CoNLL-2000 shared task* on **text chunking** (Tjong Kim Sang and Buchholz, 2000). Text chunking is concerned with dividing text into syntactically-related chunks of words, or *phrases*. These phrases are non-overlapping in the sense that a word can only be a member of one phrase. For example, consider the sentence:

He reckons the current account deficit will narrow to only #1.8 billion in September.

The segmentation of this sentence into chunks and their corresponding labels is shown in table 1. The chunk label contains the type of the chunk, e.g. `I-NP` for noun phrase words and `I-VP` for verb phrase words. Most chunk types have two kinds of labels to delineate the boundaries of

He	B-NP
reckons	B-VP
the	B-NP
current	I-NP
account	I-NP
deficit	I-NP
will	B-VP
narrow	I-VP
to	B-PP
only	B-NP
#	I-NP
1.8	I-NP
billion	I-NP
in	B-PP
September	B-NP
.	O

Table 1: Example sentence and chunkings.

the chunk, `B-CHUNK` for the first word of the chunk and `I-CHUNK` for every other word in the chunk.
While all the necessary information to carry out this assignment is contained within this assignment specification, you may also find out more about this task at

https://www.clips.uantwerpen.be/conll2000/chunking/

1 Data

Instead of providing you with raw text data, we have preprocessed and extracted features from this dataset for you. These are given in the the compressed files "`conll_train.zip`", and "`conll_test_features.zip`". When extracted, you will find files "i`.x`" and "i`.y`" consisting of the features and chunk labels for the ith sentence, respectively.

Train/test split. The training examples ("`conll_train.zip`") consist of the files with $i \leq 8,936$. The remaining 2,012 examples, i.e. the files with $8,936 < i \leq 10,948$ in the compressed file "`conll_test_features.zip`" are to be used as test examples for making predictions on. You should only train on examples in "`conll_train.zip`" and the test examples must not used for training in any way. Note that only "i`.x`" files are provided for the test data ("`conll_test_features.zip`"), as we have withheld the labels for evaluation of your algorithm performance.

Schema. Let T_i be the length of the ith sentence, the number of words/tokens it contains. There is a D-dimensional binary feature vector for each word/token in the sentence, where $D = 2,035,523$. Due to the high-dimensionality of the feature space, the "i`.x`" file provides a *sparse* representation of the feature vectors for the ith sentence. A row entry with the value

```
j k 1
```

indicates that the kth feature for the jth word/token in the sentence has value 1. Next, the "i.y" file contains the label $c \in \{1, \ldots, 23\}$ of each of the T_i words/tokens in the sentence.

2 Main Task: Classifying Chunk Tags with Gaussian Process Models

Your task is to build a probabilistic classifier based on *Gaussian processes* for predicting the class probability of the label for every word/token in a sentence. You are required to submit (i) a technical report describing your solution; (ii) the code implementing your solution; and (iii) the predicted class log probabilities for each word/token of the test examples.

3 Technical Report: `solution.pdf`

In this report you will describe your solution to the problem above. The maximum length of the report is 4 pages excluding references and appendix. Keep in mind that your assessor reserves the right to read your appendix. The report must contain only the following sections:

Abstract [≈ 1 paragraph] A short summary of your approach and the results of your method.

Introduction [≈ 1/2 page] An introduction to the problem, the basic approach you have taken and your contributions.

Model [≈ 1/2 – 1 page] A mathematical and conceptual description of your model.

Inference [≈ 1 page] A description of how your inference method works. For example, posterior inference (if applicable) and how predictions are done.

Parameter Estimation [≈ 1/2 page] A description of how parameter estimation is carried out (if applicable). Examples of this can be cross-validation, MAP, MLE or full Bayesian inference.

Results [≈ 1 page] Here you need to describe an evaluation methodology that convinces the reader that your approach is sound. For this, you need to split your training set into training and validation and show performance metrics with respect to a sensible baseline that uses a softmax classifier. The performance metrics that you need to report are the error rate (ER) and the mean negative log probability (MNLP) defined as:

$$\text{ER} = 1 - \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{k=1}^{T_i}\mathbb{I}[\hat{\mathbf{y}}_k^{(i)} = \mathbf{y}_k^{(i)}], \tag{1}$$

$$\text{MNLP} = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{k=1}^{T_i}\sum_{j=1}^{C}\mathbb{I}[\mathbf{y}_k^{(i)} = j]\log p_{\text{model}}(\mathbf{y}_k^{(i)}|\mathbf{x}^{(i)}), \tag{2}$$

where $\mathbb{I}[\cdot]$ is an indicator function that is 1 if the condition inside the brackets is satisfied and 0 otherwise; $\hat{\mathbf{y}}_k^{(i)}$ is your model's prediction and $\mathbf{y}_k^{(i)}$ is the true label for the kth word/token of sentence $\mathbf{x}^{(i)}$ in the validation set; $p_{\text{model}}(\mathbf{y}_k^{(i)}|\mathbf{x}^{(i)})$ is your model's predicted probability on class $\mathbf{y}_k^{(i)}$ for datapoint $\mathbf{x}^{(i)}$; and N is the number of datapoints in the validation set. In addition to the performance metrics above, you can report other analysis/insights of the data or results in this section.

References A full list of previous work that you used or is relevant to your report.

Appendix Additional material such as derivations or extra analysis of results.

The length of each section is provided as a guide only and you may deviate from this as long as you do not exceed the page limit of the report.

4 Code: `solution.py`

This is a Python file that implements your solution. It must be well-documented, self-contained and able to generate your predictions in the next section. The program should load the file "`conll_test_features.zip`" from the current directory and its output should be the file `predictions.txt`.

5 Predictions: `predictions.txt`

This file should contain the log probabilities $\log p_{\text{model}}(\mathbf{y}_k^{(i)} = j|\mathbf{x}^{(i)})$ for each word/token, for each sentence in the test set, in the order they were provided. There should be 2012 blocks, separated by a blank line. Within each block, there should be T_i lines. Every line must contain comma-separated log probabilities for all classes $j = 1, \ldots, 23$.

6 Group Submission

This is a group assignment with the minimum group size of 2 people and a maximum of 3 people. It can be submitted from one of the group members' account. Authorship should be stated in the technical report `solution.pdf` and the code `solution.py`.

7 Hard Constraints

While you have freedom on the method, inference machinery and coding techniques you use, these are the constraints your submission must satisfy. Failure to meet these requirements will yield an overall mark of zero.

(i) The maximum length of your technical report `solution.pdf` is 4 pages excluding references and appendix (for which there is no limit).

(ii) The minimum font size of your technical report `solution.pdf` is 11pt.

(iii) Your solution must make use of *Gaussian processes* and (optionally) one or more inference techniques as explained in the course. It can be an extension of one of the methods described in the lectures. However, methods that are completely unrelated to the course material or do not use Gaussian process in any meaningful way are not acceptable. If in doubt, please contact the course lecturer.

(iv) Your code `solution.py` must be executable on a standard architecture (Linux and Mac OS) and if non-standard Python packages (i.e. packages that are not hosted on PyPI) are required it must advise the user to install them.

(v) The prediction file `predictions.txt` must be in the format specified in §5.

(vi) Although you are given the test examples for making predictions, under absolutely no circumstances may they be used during training.

(vii) Only group submissions of 2 or 3 people are accepted.

8 Assessment

Your submission will be assessed based on the quality of the technical report and the performance of your predictions. Note that, although the code does not have a specific weight in the assessment, penalties will be applied for unsuitable documentation, unreproducible results or failure to execute (with the latter yielding an overall mark of zero). This is a breakdown of the marks:

- [**50 Marks, technical report**] The techical report must satisfy the constraints above and the marks will take into account the following criteria:
  - [10 Marks] Overall clarity of presentation. This includes clarity, formatting, organisation, language use, correct spelling and grammar. Note that rambling or waffling to fill space unnecessarily will be penalised. Your report may well be under 4 pages if it is sufficiently clear and descriptive.
  - [30 Marks] Technical description of your solution (sections Model, Inference, and Parameter Estimation of your report). This includes clarity, technical difficulty and innovation.
  - [10 Marks] Sound evaluation of your technique (section Results of your report). This includes presentation and analysis of the results.
  - A well-written appendix that expands on the technical description of your solution or on the analysis of the results may increase your overall mark. However, as stated above, your asessor reserves the right to read the appendix in detail.
- [**50 Marks, predictive performance**] Predictive performance on the test data will be evaluated using the error rate (ER) and the mean negative log probability (MNLP) as defined in equation 1 and equation 2 respectively. In order to make point-predictions for computation of the error rate, we will assume a max-probability approach, i.e. predict the class with the maximum predicted log-probability.

8.1 Additional Notes on Assessment

- Performance will be evaluated using *both measures the ER and the MNLP* and interpolating between the baseline performance and the best submission. The best submission wil be given full marks in the performance category.
- The predictions in `predictions.txt` must be generated using the proposed method, even if you found that it was not better than a simple baseline that does not use Gaussian processes. Failure to do so, will yield zero marks in the performance category.
- Nonsensical predicted log-probabilities (i.e. NaNs or positive values) will be highly penalized.

9 Software

You can use any software package for developing and evaluating your method. In particular, you may find useful the following Gaussian process packages:

- GPflow: `https://github.com/GPflow/GPflow`
- AutoGP: `https://github.com/ebonilla/AutoGP`
- GPy: `https://sheffieldml.github.io/GPy`

References

Tjong Kim Sang, E. F. and Buchholz, S. (2000). Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics.