UNIVERSITY OF CAMBRIDGE

PART II PROJECT PROPOSAL

ANDREW WELLS (AW684) - QUEENS' COLLEGE

Deep Learning for Music Recommendation

Name:
Andrew WELLS

Supervisor:
Catalina CANGEA

Overseers:
Dr Hatice GUNES
Dr Robert WATSON

Director of Studies:
Dr Alastair BERESFORD

October 18, 2017

Introduction and Description

With the influx of users moving towards music-streaming services such as Spotify, Apple Music, and Pandora, content providers are now, more than ever, pushing the boundaries of music classification and recommendation. Features like Discover Weekly on Spotify and Apple's automated song suggestions are just two examples of this fascinating area of Computer Science. A plethora of Machine Learning techniques have been and are used for classifying songs by genre, mood, instrument and era. Many content-rich, data-driven music-streaming applications have fundamentally changed how users listen to and discover new music.

Collaborative filtering [4] is one of the methods used for recommending new songs and artists and is based on historical usage data, clustering users together based on the songs they've listened to. At first sight, this seems like a perfectly suitable method of recommending songs, but it ignores new songs, new artists and bands that haven't yet been listened to frequently. Music discovery should be oriented towards giving users the best possible recommendations, a possible approach to this being to analyse the audio signal itself.

Many Music Information Retrieval Systems have a two-stage approach, starting with feature extraction from audio signals, and subsequently using these features as input to a classifier or regressor such as logistic regression or support vector machines. The problem with this approach, however, is that *any* audio signal must first be analysed using complex DSP techniques to find such features.

This project will explore the use of CNNs for music tagging, which requires only an audio spectrogram of the audio signal, and as such vastly reduces the time spent on pre-processing. I will be generating spectrograms of the audio files, and using these as input to a Convolutional Neural Network (CNN), a model which has been actively used in a number of music classification tasks. Spectrograms will be provided as input because CNNs are in general well-suited to extracting features from images, and they provide perhaps the best representation of the audio signal for CNN architectures.

The CNN architecture I have chosen to implement [3] consists of 5 convolutional layers followed by 2 fully-connected layers. The output of the convolutional layers are fed into the fully connected layers, which act as the classifier, outputting the probability that each of the top-50 tags can be associated with the song. Sigmoid functions are used as activation at output nodes to facilitate multi-label classification.

Starting Point

I will be starting my Part II project with the following prior experience and knowledge:

- basic machine learning concepts from the Part IB Artificial Intelligence course
- basic knowledge of Python language

The project will thus require that I research and explore the following:

- the theory and background to ANNs with a focus on CNNs
- Python programming for TensorFlow's API (the only version that has stability guarantees)
- content from the Digital Signal Processing course (Michaelmas 2017) on spectrograms of audio-signals
- audio signal processing frameworks

Project Structure and Substance

Key concepts

The project will involve substantial research and knowledge acquisition in the theory of neural networks, an understanding of CNNs for music classification, and the network architectures that can be used. In addition, a ground-level knowledge of Digital Signal Processing techniques will be needed to understand the implementation of pre-existing libraries for audio-signal processing, which will be used in generating the spectrograms.

Major work items

My project is to build a music classification system using CNNs. This will involve classifying songs according to the top-50 relevant tags, acquired from the *last.fm* subset of the Million Song Dataset [2]. Sample audio will be fetched from services such as *7Digital*. The tags include genre, mood, era and instruments. The classification system (CS) will be implemented through the use of Convolutional Neural Networks which will use the $2D$ kernel, $1D$ convolution model *k2c1* described in the paper [3]. This model has been chosen specifically as it is motivated by structures for music tagging and genre classification.

The CNN system I will be implementing takes as input a spectrogram of an audio clip and outputs the probability that each of the top-50 tags is associated with the song.

The implementation of the CNN will require extensive knowledge of the TensorFlow API. It was decided that the implementation language be Python,

as TensorFlow only offers API stability guarantees for Python. The training and testing of the CNN will require use of a GPU, which will be obtained from the Computational Biology Group at the Computer Laboratory.

It is worth noting that the substantial part of the project consists of the CNN implementation, the choice of network architecture, and the training of the network. The recommendation system itself relies purely on the output of the CNN, so it is essential that the CNN is tested and evaluated before a recommendation system is built.

Using the top-50 tags for songs in the Million Song Dataset, a t-SNE plot can be made using songs in the dataset, to evaluate the potential of the system for clustering similar songs and being part of a recommendation system.

t-distributed stochastic neighbour embedding (t-SNE) is a machine learning algorithm for dimensionality reduction, which will project the top-50 tags of a song into a 2D space. The more similar two songs are, the closer they are to one another in some n-dimensional space. t-SNE preserves distances between points in higher dimensional space, so those points that are close together (i.e. similar songs) in a higher dimension will map to points that are close together in 2D space. The clustering of songs in a 2D space would then allow recommendations to be made, based upon a simple distance metric between points.

Methods of Evaluation

The implementation can be evaluated using the last.fm tags included in the Million Song Dataset. The majority of the evaluation will be purely quantitative; I have chosen to use AUC-ROC, as it is widely used for classification problems and will allow a direct comparison between the implementation found in [3] and my own.

AUC-ROC (Area Under the Receiver Operating Characteristic curve) is a metric actively used for assessing the performance of classifiers. Counting false-positives and true-positives will indicate incorrectly generated tags and correctly generated tags respectively. To allow for a comparison of performance between my implementation and that of [3], I will be reproducing the results for AUC-ROC per tag, and then averaging the AUC-ROC over all tags to obtain a single measure for the classifier's performance. Using this metric will allow a comparison between my implementation and the paper's implementation of the CNN.

For all models, evaluation will be carried out using the holdout method, which splits the dataset into training and test partitions, training on the former and testing on the latter.

Success Criteria

The core project will be considered successful upon completing the following:

1. A working implementation of a Convolutional Neural Network
2. The ability of the network to perform better-than-chance prediction (accuracy above 50% for each tag and the lower bound of its confidence interval above 50%) of the top-50 tags for a given audio file.

Extensions

Should I complete the core objectives, I will aim to do the following, in order of preference:

1. Analyse the high-level representations learned by the model by producing t-SNE plots for various representative subsets of the songs in the dataset, and explore its use in music recommendation.
2. Implement other models in the paper [3] to improve the performance of the system and enable further opportunity for comparison between my implementation and that of the paper
3. Implement an attention mechanism [1] for the CRNN model above to improve results even further

Project Plan

Week 0 (21st October—27th October)
Begin researching the theory behind convolutional neural networks, and the theory behind DSP techniques for spectrograms.

Weeks 1 & 2 (28th October—10th November)
Continue researching the theoretical aspects, and look at libraries to be used for the audio-signal processing.
Milestone: The theory topics have been covered and can be applied to the project.

Weeks 3 & 4 (11th November—24th November)
Start writing code snippets and gain familiarity with the libraries to be used. Implement a simple ANN with TensorFlow, and learn about how CNNs are implemented with TensorFlow.
Milestone: Familiar with the libraries and implemented the audio-signal pre-processing module.

Weeks 5 & 6 (25th November—8th December)
Implement the CNN
Milestone: A working implementation of the CNN has been completed.

Weeks 7 & 8 (9th December—22nd December)
Slack time for delays and unexpected issues with the CNN implementation. Can begin training and testing the CNN using the Million Song Dataset.

Weeks 9 & 10 (23rd December–5th January)
Finish the training and testing of the network's performance.
Milestone: The network provides a better-than-chance prediction of the top-50 tags for a given song.

Weeks 11 & 12 (6th January—19th January)
Perform hyperparameter optimisation to improve the accuracy of the network. Start implementing other models.
Milestone: Completion of training and testing. The CNN now has improved accuracy.

Weeks 13 & 14 (20th January—2nd February)
Additional slack time for delays in training and testing. Begin to write progress report and evaluate the project.
Milestone: Progress report finished and submitted before 2nd February. Completed implementation of other models.

Weeks 15 & 16 (3rd February—16th February)
Revise the evaluation based on feedback, and write the presentation. Continue testing other models.

Weeks 17 & 18 (17th February—2nd March)
Analyse the potential of the system for recommendation, using t-SNE.
Milestone: t-SNE plots are finalised. Evaluation of other models completed.

Weeks 19 & 20 (3rd March—16th March)
Start writing up the dissertation, focussing on the Introduction chapter.
Milestone: Introduction chapter has been completed.

Weeks 21 & 22 (17th March—30th March)
Write up the Preparation and Implementation chapters of the dissertation.
Milestone: Preparation and Implementation chapters have been completed.

Weeks 23 & 24 (31st March—13th April)
Write up the Evaluation and Conclusion chapters of the dissertation. Amend any changes suggested by my supervisor and/or DoS.
Milestone: The Introduction, Preparation, Implementation, Evaluation and Conclusion chapters of the dissertation are completed and ready for submission to supervisor.

Weeks 25 & 26 (14th April—27th April)
Send dissertation to supervisor by 16th April and amend any changes from comments received.

Weeks 27 & 28 (28th April—11th May)
Slack time for changes to write-up.
Milestone: Dissertation has been completed and submitted by 11th May to my DoS for approval.

Week 29 (12th May—18th May)
Slack time for any final comments received from my DoS.
Milestone: Final Dissertation submitted before 16th May.

Resource Declaration

My Macbook Pro (2.5 GHz Intel Core i7, 16 GB 1600 MHz DDR3 RAM, AMD Radeon R9 M370X 2048 MB) running macOS 10.12 Beta (16A313a) will be used for the project implementation and initial testing. Should the training take too long, I will use a GPU acquired from the department. I accept full responsibility for the equipment used and will use MCS machines should failure arise.

Regular back-ups of the project data will be undertaken, with the project implementation being stored locally, on-disk with usage of BitBucket which provides free private repositories and git for version control. The write-up will be synced with iCloud Drive and also be stored in a repository on BitBucket, using git for version control.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv* (2014).

[2] Thierry Bertin-Mahieux et al. "The Million Song Dataset". In: *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*. 2011.

[3] K. Choi et al. "Convolutional Recurrent Neural Networks for Music Classification". In: (2016). URL: https://arxiv.org/pdf/1609.04243.pdf.

[4] Sander Dieleman. "Recommending music on Spotify with deep learning". In: (2014). URL: http://benanne.github.io/2014/08/05/spotify-cnns.html.