# Writing Scientific Papers and Software

Cheng Soon Ong
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—A critical part of scientific discovery is the communication of research findings to peers or the general public. Mastery of the process of scientific communication improves the visibility and impact of research. While this guide is a necessary tool for learning how to write in a manner suitable for publication at a scientific venue, it is by no means sufficient, on its own, to make its reader an accomplished writer. We also describe the rules for submission in the computational intelligence laboratory. This guide should be a starting point for further development of writing skills.**

## I. Introduction

Recommender systems have gained increasing popularity in the last 10-15 years, mainly due to their immediate market applications. The huge selection of products online offer retailers the opportunity to attract a wide-range of clients. However, it also means that users can no longer be expected to browse through everything available until they find the film, song or book they like the most. Good recommendations are key to both making the correct products available to the appropriate audience, but also to creating a unique, personalised experience which will attract new clients. These ideas are fundamental for many leading online businesses [ref netflix, ref amazon, ref something else].

In this paper, we concentrate on the collaborative-filtering approach to recommender systems, which relies on previous user experiences (ratings), rather than pre-defined user profile [some ref about the two types]. We use the Netflix dataset with ratings from 10000 users on 1000 films, which has driven a lot of the innovation in the area and allows us to compare our solutions to the state-of-the-art.

Our main contribution consists of combining latent factor models (SVD, SGD SVD, RBM) and neighbourhood models (user- and item-based kNN) to produce a robust recommender system which optimises the RMSE of predicted ratings on X[how many films] from Blaa to Bllaa. Furthermore, we show that the addition of a more complex learning rate algorithm improves the score significantly, to X.

In Section II we describe the idea and the implementation of our systems. We present the results from the local evaluation of the models and from the validation set on Kaggle in Section III. Finally, in Section IV we interpret the improvements we achieved and discuss potential weakness and future work.

## II. Models and Methods

All work is done on the Netflix dataset provided for the project. The data is a collection of *(user, film, rating)* tuples. For training, we have 1176953 ratings available. The online evaluation on Kaggle is performed on another 1176953 ratings. We were able to see the prediction accuracy on 50% of that online dataset and our goal was to improve that as it is believed to be very similar to the other 50% of the data.

### A. Data Model

The provided dataset can be most naturally seen as an $NxM$ matrix, with $N$ being the 10000 unique users, and $M$ - 1000 films. If every user had rated every film, the matrix would have more than $10^7$ entries. However, in this realistic scenario, this is not the case and our matrix is quite sparse. From here stems the difficulty of the problem - we attempt to predict more than a million ratings based on very incomplete dataset.

There are two main views which define the methods of tackling this problem.

***Latent factor models**:* This approach tries to explain the ratings as influenced by a set of unknown factors. Both users and items can be characterised by those latent factors and they are represented as feature vectors in this latent feature space. The rating of a film is approximate by the dot product between the user and the film feature vectors.

The different methods then aim to infer the underlying film and user feature vectors from the available ratings, most successfully through matrix factorisation techniques and approximations. We discuss SVD, SGD SVD and RBM.

***Neighbourhood-based models**:* Another approach to the collaborative filtering problem is to view the ratings as features of either the users (user-based) or the films (item-based). In the former case, a user A is represented as a sparse feature vector of their ratings and we would base our prediction on other users who have rated those films similarly. In the latter scenario, we would use the similarity between the new film and the films already rated by A to estimate the new rating. Each film is a feature vector of all users' ratings. We review an implementation of both in the next subsection.

### B. Algorithms

We investigated three algorithms which concentrate on solving the latent-factor model problem and two which work

on the neighbourhood model:

***SVD**:* Given the latent factor model described above, the rating matrix is produced by the multiplication of the users matrix and films matrix which are all on the same feature space. The most natural way to infer the users and films representation is using a matrix factorisation approach so SVD comes directly to mind[some refs here]. The main issue is that SVD works on full matrices and as mentioned above, this is not the case. Hence, we need to impute the missing values before we can proceed. This introduces noise and leads to sup-optimal results. We hope to minimize the noise and the overfitting by looking for a low-rank ($k$) approximation of the rating matrix.

We experimented with imputation strategies, different number of latent features and values for k using a local validation set. We concluded that the best results were accomplished using XX features and Equation 1 for imputation:

$$R_ij = 0.5 * AvgUser[i] + 0.5 * AvgFilm[j] \quad (1)$$

A missing rating of user $i$ for film $j$ is calculated as a linear combination of the average rating given by user $i$ and the average rating received by film $j$ from all other users. If any of those values is missing due to a film with no ratings or user with no rated films, the other values is used.

We have used the Numpy build-in SVD algorithm for this. Despite the optimisation strategies, we expect the results achieved with SVD to be suboptimal due to the large number of missing values, and the inaccuracy introduced by the imputation of those values. We use this algorithm as a baseline and we try to improve it with better strategies for matrix factorisation or alternative approaches of finding the latent factor representation.

***SGD for SVD**:* The main idea of the algorithm is to avoid the need of filling in the missing values in the k-rank approximation of the rating matrix.

Instead of doing the standard SVD which requires a full matrix, we use only the available ratings in a Stochastic Gradient Descent method. The function we minimize is :

$$equationforSGD \quad (2)$$

The approximation should be much more representative of the data as it only uses real ratings. The algorithm was very sensitive to hyper-parameters so those were optimised on both a local validation set and the online Kaggle score.

***RBM**:* stuff stuff

***user-based kNN**:* Find the top k most similar users using a Pearson distance metric. The rating is the weighted average of the top k userś rating for the film.

***item-based kNN**:* Calculate the similarity between the films a user has already rated and the film in question. The predicted rating is then a weighted average of their ratings using the similarities as weights.

### C. Bold Driver Learning Rate

One of the biggest boosts to our system was the incorporating of the bold drive technique in the calculation of the learning rate in both SGD and RBM. This is an adaptive learning rate given by: [some formula]

### D. Parameter optimisation and validation

The general protocol for training and testing followed the steps below:

1) Choose a random subset of the test samples for a local validation set if not already chosen. This was in most cases 5% of the data, so xxxx samples and was only done by the first algorithm, so that all algorithms are trained on the same data.
2) Train each individual algorithm on the leftover test dataset. In some cases, such as in the item-base kNN, the validation set was used to automatically choose the optimal k in the predefined range. In the iterative algorithms like SGD and RBM, we used the validation set as an early indication on whether the error is really falling with every step. We would often optimise certain parameters using this validation data, such as number of features to user or algorithm iterations (see results for details).
3) Each algorithm generates predictions for the local validation set and the test set which would be uploaded on Kaggle.
4) The predictions from all algorithms are combined using linear regression. The regressor is trained on the predictions generated for local validation set for which we have ground truth. The coefficients which fit the results best are then used to generate a final prediction from all individual predictions for the online test dataset.

## III. Results

Use examples and illustrations to clarify ideas and results. For example, by comparing Figure 1 and Figure 2, we can see the two different situations where Fourier and wavelet basis perform well.

## IV. Discussion

The models and methods section should describe what was done to answer the research question, describe how it was done, justify the experimental design, and explain how the results were analyzed.

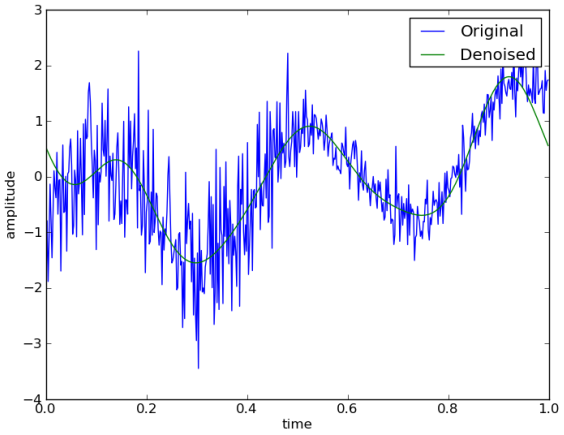


Figure 1. Signal compression and denoising using the Fourier basis.

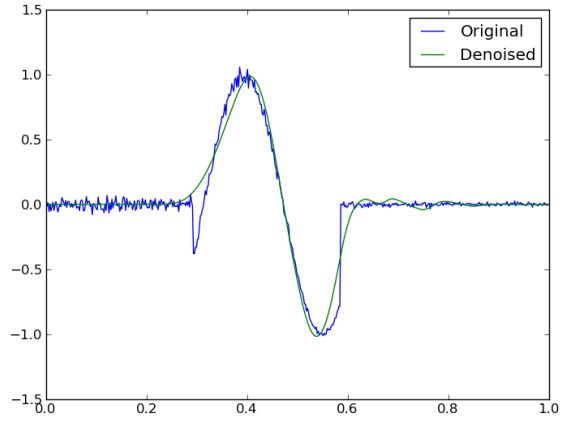


Figure 2. Signal compression and denoising using the Daubechies wavelet basis.

The model refers to the underlying mathematical model or structure which you use to describe your problem, or that your solution is based on. The methods on the other hand, are the algorithms used to solve the problem. In some cases, the suggested method directly solves the problem, without having it stated in terms of an underlying model. Generally though it is a better practice to have the model figured out and stated clearly, rather than presenting a method without specifying the model. In this case, the method can be more easily evaluated in the task of fitting the given data to the underlying model.

The methods part of this section, is not a step-by-step, directive, protocol as you might see in your lab manual, but detailed enough such that an interested reader can reproduce your work [**?**], [**?**].

The methods section of a research paper provides the information by which a study's validity is judged. Therefore, it requires a clear and precise description of how an experiment was done, and the rationale for why specific experimental procedures were chosen. It is usually helpful to structure the methods section by [**?**]:

1) Layout the model you used to describe the problem or the solution.
2) Describing the algorithms used in the study, briefly including details such as hyperparameter values (e.g. thresholds), and preprocessing steps (e.g. normalizing the data to have mean value of zero).
3) Explaining how the materials were prepared, for example the images used and their resolution.
4) Describing the research protocol, for example which examples were used for estimating the parameters (training) and which were used for computing performance.
5) Explaining how measurements were made and what calculations were performed. Do not reproduce the full source code in the paper, but explain the key steps.

### A. *Results*

Organize the results section based on the sequence of table and figures you include. Prepare the tables and figures as soon as all the data are analyzed and arrange them in the sequence that best presents your findings in a logical way. A good strategy is to note, on a draft of each table or figure, the one or two key results you want to address in the text portion of the results. The information from the figures is summarized in Table I.

When reporting computational or measurement results, always report the mean (average value) along with a measure of variability (standard deviation(s) or standard error of the mean).

*1) Installation:* There are various different packages available for processing LaTeX documents. On Windows, use the MikTeX package (http://miktex.org/), and on OSX use MacTeX (http://www.tug.org/mactex/2009/). Alternatively, on OSX, you can install the `tetex` package via Fink[1] or Macports[2].

*2) Compiling LaTeX:* Your directory should contain at least 4 files, in addition to image files. Images should be in `.png`, `.jpg` or `.pdf` format.

- IEEEtran.cls
- IEEEtran.bst
- groupXX-submission.tex
- groupXX-literature.bib

Note that you should replace groupXX with your chosen group name. Then, from the command line, type:

```
$ pdflatex groupXX-submission
```

[1]http://www.finkproject.org/
[2]http://www.macports.org/

| Basis | Support | Suitable signals | Unsuitable signals |
|---|---|---|---|
| Fourier | global | sine like | localized |
| wavelet | local | localized | sine like |

Table I
CHARACTERISTICS OF FOURIER AND WAVELET BASIS.

```
$ bibtex groupXX-literature
$ pdflatex groupXX-submission
$ pdflatex groupXX-submission
```

This should give you a PDF document `groupXX-submission.pdf`.

*3) Equations:* There are three types of equations available: inline equations, for example $y = mx+c$, which appear in the text, unnumbered equations

$$y = mx + c,$$

which are presented on a line on its own, and numbered equations

$$y = mx + c \tag{3}$$

which you can refer to at a later point (Equation (3)).

*4) Tables and Figures:* Tables and figures are "floating" objects, which means that the text can flow around it. Note that `figure*` and `table*` cause the corresponding figure or table to span both columns.

### B. Grading

There are two different types of grading criteria applied to your project, with the corresponding weights shown in brackets.

Competitive

The following criteria is scored based on your rank in comparison with the rest of the class.

- time taken for computation (10%)
- average rank for all other criteria relevant to the task, for example reconstruction error and sparsity (20%)

The ranks will then be converted on a linear scale into a grade between 4 and 6.

Non-competitive

The following criteria is scored based on an evaluation by the teaching assistants.

- quality of paper (30%)
- quality of implementation (20%)
- creativity of solution (20%)

### C. Submission System

The deadline for submitting your project report is Friday, 22 June 2012. You need to submit:

- PDF of paper.
- Archive (`.tar.gz` or `.zip`) of software. Please do not forget to include author information in the source archive.

**Important:** Please check the submission instructions on the webpage as it is the most updated instructions.

## V. SUMMARY

The aim of a scientific paper is to convey the idea or discovery of the researcher to the minds of the readers. The associated software package provides the relevant details, which are often only briefly explained in the paper, such that the research can be reproduced. To write good papers, identify your key idea, make your contributions explicit, and use examples and illustrations to describe the problems and solutions.

## ACKNOWLEDGEMENTS

The author thanks Christian Sigg for his careful reading and helpful suggestions.