Assignment 2: Incorporating Temporal Dynamics for Personalized Beer Rating Prediction

Abstract

Recommender systems require large amounts of user data in order to adequately understand user preferences. User preferences and item popularity can also change over time. Static models may miss important trends such as a general increase in item rating over time, or a periodic change in rating dependent on summer versus winter months. This work uses the *rateBeer* dataset and incorporates temporal dynamics to improve rating prediction models. We use a latent factor model and incorporate both periodic and yearly temporal dynamics to lower overall mean squared error. The temporal latent factor model outperforms static latent factor models and bias-only models, showing temporal dynamics should be utilized for modern day recommendation systems.

Keywords

recommender systems, latent factor models, temporal data

1 Introduction

Rating prediction is one of several ways to recommend an item to a user. We look at rating prediction in the context of supervised learning, or learning from labeled review data. A model is first trained using known star ratings as labels. The model internally has learnable parameters that are updated depending on the mean squared error between the ground truth star rating and the predicted rating. The unseen item with the highest predicted star rating can then be recommended to a user.

In order to accurately recommend an item to a user, we must fully understand their personal preferences. These preferences often change over time as a user encounters diverse items and expands their taste. We look at recommendation in the context of beer reviews. We use the review website *rateBeer* [5, 6] to improve upon standard rating prediction models in order to recommend better products. For example, a user who enjoys very "hoppy" beers will most likely give a lower rating to lighter beers such as miller lite. Ratings also increase or decrease over time due to changes in item quality or user preference. If temporal data is available, it must be used to understand how ratings change over time. Modern day rating predictors must account for all of this information to give recommendations tailored to a user's current taste.

2 Related Work

2.1 Latent Factor Models

Koren et al. [3] introduced latent factor models (LFM) as an effective approach to collaborative filtering, emphasizing their superiority over unsupervised nearest neighbor techniques. Their work, primarily motivated by the Netflix Prize competition, demonstrated the power of matrix factorization techniques in capturing latent factors that drive user preferences and item popularity. The matrix factorization model introduced in their work decomposes the user-item interaction matrix into a product of lower-dimensional matrices, with latent factors representing the underlying characteristics of users and items. The model also incorporated regularization to prevent overfitting and improve generalization.

The equation for the latent factor model (LFM) is given by Equation 1, where $f(u, i)$ represents the predicted rating for user u and item i, α is the global bias, β_u and β_i are user and item biases, and $\gamma_u \cdot \gamma_i$ is the interaction between latent user and item factors. This model was highly effective, and its success in the Netflix competition laid the foundation for subsequent research in recommender systems.

$$f(u, i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i \tag{1}$$

2.2 Temporal Dynamics in Collaborative Filtering

Temporal dynamics are crucial in collaborative filtering, as user preferences and item popularity evolve over time. Koren [2] expanded upon the latent factor model by incorporating temporal dynamics to better capture the evolution of user and item preferences. Through analysis of the Netflix movie rating dataset, Koren identified significant temporal trends, such as sudden shifts in ratings or gradual increases in ratings as items age. To account for these changes, Koren introduced time-based adjustments to the bias terms for both users and items. Specifically, item biases were split into time-based bins, allowing the model to capture shifts in item perceptions over time. In his work, the number of bins was dataset-specific, with each bin corresponding to approximately 10 weeks of data. The model to capture temporal item bias is given by Equation 2, where $\beta_i(t)$ is the temporal bias for item i at time t, and $\beta_{i,\text{Bin}(t)}$ represents the item bias within the specific time bin.

$$\beta_i(t) = \beta_i + \beta_{i,\text{Bin}(t)} \tag{2}$$

For user bias, Koren chose not to use a binned approach due to the sparsity of user reviews over time. Instead, he used a linear function or spline based on the time distance from the user's mean rating date to model gradual shifts in user preferences. This method effectively captures how user biases drift over time, providing more accurate rating predictions.

Our work differs slightly from [2] for a few reasons. First, we use the *rateBeer* dataset instead of Netflix movie ratings. Secondly, we incorporate periodic effects, a temporal effect mentioned but not used in [2]. We model monthly and weekly periodic effects, which account for any possible cyclic shifts that occur. For example, maybe ratings are lower on Mondays because users are upset about having to work after a long weekend. Another cyclic shift could be due to the change in season. These are important aspects of the *rateBeer* dataset that must be modeled.

2.3 Temporal Effects and Expertise in Rating Prediction

McAuley et al. [5] applied temporal latent factor models to the *rateBeer* dataset, incorporating user expertise as a key factor in predicting ratings. They showed that the similarity between two users' ratings of an item is influenced not only by their preferences but also by their expertise. Users with higher expertise tend to provide more consistent ratings over time. The authors found that expertise-based models performed particularly well when reviewing products at similar stages of expertise, even if there was significant time separation between reviews.

Unlike McAuley's approach, which directly models user expertise, our work focuses solely on capturing periodic and temporal drift without explicitly modeling user expertise. While user expertise can certainly influence ratings, we chose to focus only on time-of-week or time-of-year effects.

2.4 Other Collaborative Filtering Approaches

Many other approaches exist that were not used in this work. Salakhutdinov and Mnih [7] proposed a Bayesian approach to matrix factorization, which introduced a probabilistic framework for modeling uncertainty in the learned latent factors. The advantage of the Bayesian approach is that it allows for modeling uncertainty in both user and item latent factors, which leads to more robust predictions and the ability to quantify the confidence in the model's outputs.

A more recent development in collaborative filtering uses neural networks for learning complex latent factors. He et al. [1] introduced neural network based collaborative filtering models, which extend traditional matrix factorization by using deep neural networks to model the interactions between users and items ($\gamma_u \cdot \gamma_i$). These models learn nonlinear latent representations of users and items, which can capture more complex patterns in the data than traditional linear latent factor models.

3 Dataset

We demonstrate the effectiveness of incorporating temporal information by using the beer-rating website *RateBeer*. The *RateBeer* dataset, used by [5, 6], consists of nearly 3 million ratings across the years 2000-2012. An example rating is shown in Figure 1. Overall reviews are out of 20, however, we normalized all reviews to 5 stars in order to match more typical star ratings. The normalization from 0 to 5 was also used by [5, 6] when using this dataset. A review also contains natural text and star ratings for different aspects of the beer including appearance, aroma, palate, and taste. This work opts for simplicity and only looks at overall rating. We predict overall ratings by analyzing the user, beer, and timestamp of the review only and ignore the text review.

The *rateBeer* dataset has 40,213 unique users, 110,419 unique beers, and 2,924,127 ratings [5]. We split the dataset into train, validation, and test sets to experiment with different rating prediction models and parameters. Figure 1 shows metrics for random training validation, and test splits of the review data. 90% of the data was used for training, while 5% of the data was used for validation and test. The mean rating for training, validation, and test is 3.3 out of 5 stars.

John Harvards Simcoe IPA, reviewed by **hopdog**

September 6th, 2006

On tap at the Springfield, PA location. Poured a deep and cloudy orange (almost a copper) color with a small sized off white head. Aromas or oranges and all around citric. Tastes of oranges, light caramel and a very light grapefruit finish. I too would not believe the 80+ IBUs - I found this one to have a very light bitterness with a medium sweetness to it. Light lacing left on the glass.

Appearance: 4 Aroma: 3 Palate : 3 Taste: 3 **Overall: 3.25**

Figure 1: An example review from *rateBeer* with normalized ratings from [0, 5). We predict overall ratings by analyzing the user, beer, and timestamp of the review.

	# Users	# Beers	# Ratings	Mean Rating
Train	28244	107765	2631747	3.300
Validation	10068	35935	146207	3.303
Test	9994	35867	146209	3.300

Table 1: Train, validation, and test statistics on the rateBeer [5] dataset. Reviews are randomly sampled and ratings are normalized (0, 5].

Table 2 shows the number and percentage of new users and beers on a per-rating basis for the validation and test sets. These users and beers are known as cold-starts [4]. It can be difficult to give an accurate recommendation if the user or beer isn't in the training corpus when feature data isn't used. Cold starts are kept to a minimum for unseen data. Out of over 140,000 ratings, only 0.36% are from new users. Similarly, less than 1% of ratings are for new beers.

	# New Users	# New Beers
Validation	533 (0.36%)	1382 (0.95%)
Test	526 (0.36%)	1380 (0.94%)

Table 2: New user and beer data in the validation and test set of the rateBeers [5] compared to the training corpus.

Figure 2 highlights many important temporal aspects of the *rateBeer* dataset. To begin, the years 2000 and 2001 have extremely noisy and inconsistent ratings. This is likely because the *rateBeer* website was less known in its early years, causing less reviews and higher variance in ratings. Additionally, the years 2002-2012 follow a positive linear trend. In 2002, the average rating was around 3.16 while in 2012, the average rating increased to roughly 3.41.

We fit a linear regression model over the training data and found the line of best fit as shown in Equation (3). In this equation, r is the predicted rating and x is the normalized year from 2002-2012. The year 2002 corresponds to $x = 0$ and 2012 corresponds to $x = 1$. The years 2000 and 2001 were ignored due to the large observer variance in Figure 2. This observation suggests that incorporating temporal information into a mode may improve rating prediction results.

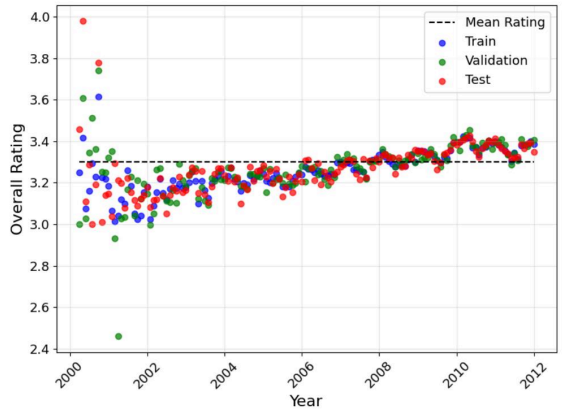

Figure 2: Mean overall beer rating as a function of time. Reviews are averaged using a sliding window of one month.

$$r = 0.2484 * x + 3.1595 \tag{3}$$

Another important temporal aspect of the data is the periodic nature of reviews over a calendar year. In order to visualize trends over a calendar year, we removed all noisy review data from years 2000 and 2001 and detrended the rating predictions for years 2002-2012 by subtracting off the linear model prediction in Equation (3). This produced Figure 3, which shows a small decrease in overall beer ratings in the months of June, July, and August. November and December tent to have higher overall ratings for all beers compared to all other months.

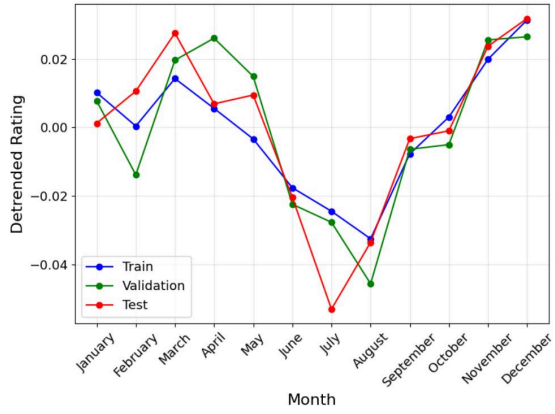

Figure 3: Yearly periodic trends for the *rateBeer* dataset. Ratings are first detrended to account for the increase in overall rating over time.

Similar to looking at periodic trends over each month, we can also look at average reviews each day of the week. Figure 4 displays weekly cyclic trends for the *rateBeer* dataset. The overall ratings are detrended similar to the yearly periodic ratings. Reviews are highest on Fridays on average. This could be because users are generally more happy when their weekend starts and more likely to give a higher rating.

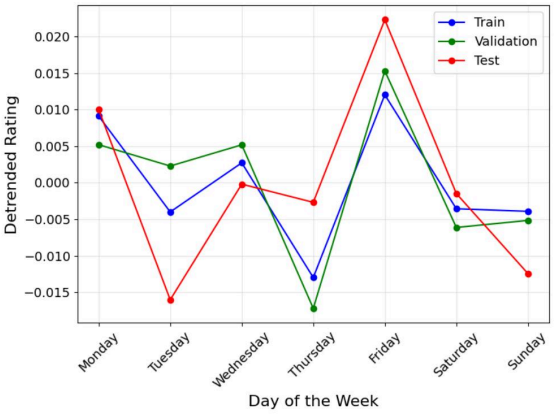

Figure 4: Weekly periodic trends for the *rateBeer* dataset. Ratings are first detrended to account for the increase in overall rating over time.

4 Predictive Task

This work seeks to predict the overall star rating for reviews in the *rateBeer* dataset in order to recommend better products for users. As mentioned in Section 3, a beer review contains many different features. These include the user ID, beer ID, timestamp, and plain text review. Only the user ID, beer ID, and timestamp are used in this work. Additionally, only the overall rating out of 5 stars is used as a label. All other sub-category ratings are ignored. Figures 2, 3, and 4 motivate the idea that a model will benefit by incorporating temporal data. We model rating drift as well as monthly and cyclic trends directly in our model using learnable parameters. Equation 4 provides the input and output information for the temporal model f, u, i, and t are the user, item, and timestamp and $\hat{r}_{ui}$ is the predicted rating for a review.

$$f(u, i, t) = \hat{r}_{ui} \tag{4}$$

Baselines. Several different baselines are used to compare with the temporal model proposed in this work. The most trivial baseline is to use the mean rating (3.3 stars) as the prediction. More sophisticated baseline models are static bias-only and latent factor models. In this work, bias-only models mean that only the bias terms in Equation 1 are used (α, β_u, β_i). These models ignore the timestamp from the data and make predictions solely based upon user and item trends. A concise description of all models used is provided in the list below. Exact methods for our work (models that incorporate temporal dynamics) are described in the next section.

- **Trivial baseline**: The mean rating of the training set is used as a predictor.
- **Bias-only model**: Only bias terms α, β_u and β_i are used as learnable parameters from Equation 1

- **Latent factor model**: User and item latent factor parameters γ_u and γ_i are used as learnable parameters in addition to the bias terms as described in Equation. 1.
- **Temporal bias-only model**: Bias terms evolve as a function of time using binning techniques. Exact methods are described in Section 5.
- **Temporal latent factor model**: User latent factors (γ_u) evolve as a function of time in addition the bias terms. Exact methods are described in Section 5.

Model Evaluation. All models are evaluated using mean squared error (MSE) over the reserved test set, which contains over 140,000 ratings randomly sampled from the *rateBeer* dataset. MSE is defined in Equation 5, where $\mathcal{D}$ is the reserved test dataset, r_{ui} is the ground truth overall rating out of 5 stars, and $\hat{r}_{ui}$ is the predicted overall rating.

$$\text{MSE}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{r_{ui} \in \mathcal{D}} (\hat{r}_{ui} - r_{ui})^2 \tag{5}$$

5 Model

Our work analyzes two separate temporal models. The first is a temporal bias-only model (Equation 6), which utilizes user and item bias terms as a function of time. The second is a temporal latent factor model that evolves the user latent factors as a function of time in addition to the bias terms (Equation 7). All temporal terms are modeled using binning as mentioned by [2] in Equation 2. Learnable parameters are added to address temporal drift as well as periodic weekly and yearly trends.

$$f(u, i, t) = \alpha + \beta_u(t) + \beta_i(t) \tag{6}$$

$$f(u, i, t) = \alpha + \beta_u(t) + \beta_i(t) + \gamma_u(t) \cdot \gamma_i \tag{7}$$

Temporal Drift. We noticed a temporal drift in overall rating after visualizing the data as seen in Figure 2. To model this, two learnable parameters was added to the model: $\beta_{u,drift}$ of size $|u| \times 13$ and $\beta_{i,drift}$ of size $|i| \times 13$, where u and i are the users and items respectively. The number 13 is used because there are 13 total years in the dataset (2000-2012). A rating from the year 2000 would fall into the first bin while a rating from the year 2012 would fall into the 13th bin. The temporal drift terms seek to model how much an individual user and item bias shifts on a yearly basis.

Yearly Periodic Trends. Analyzing the dataset also showed several interesting periodic trends over a calendar year, namely that summer months tend to have slightly lower ratings. We incorporated this into our model by binning a rating according to the month of the review. There are 12 months in a calendar year, so naturally the size of the bins for $\beta_{u,yearly}$ and $\beta_{i,yearly}$ is also 12. Ratings in the month of January fall into the 1st bin while ratings in the month of December fall into the 12th bin.

Weekly Periodic Trends. Finally, analyzing the dataset showed other interesting periodic trends over a weekly period. Reviews tend to be higher on Fridays and lower on Tuesdays. We incorporated this into our model by binning a rating according to the day of the week of the review. There are 7 days in a week, so naturally the size of the bins for $\beta_{u,weekly}$ and $\beta_{i,weekly}$ is are also 7. Ratings on Mondays fall into the 1st bin while ratings on Sundays fall into the 7th bin.

These definitions for temporal data now fully explain our temporal bias only and latent factor models. Parameters for the temporal user latent factor γ_u were found in the exact same way as for user bias.

$$\beta_u(t) = \beta_u + \beta_{u,drift} + \beta_{u,yearly} + \beta_{u,weekly} \tag{8}$$

$$\beta_i(t) = \beta_i + \beta_{i,drift} + \beta_{i,yearly} + \beta_{i,weekly} \tag{9}$$

$$\gamma_u(t) = \gamma_u + \gamma_{u,drift} + \gamma_{u,yearly} + \gamma_{u,weekly} \tag{10}$$

Regularization. To prevent overfitting and improve the generalizability of our model, we apply L_2 regularization to all learnable parameters in the model. The regularization term penalizes large parameter values and is added to the overall loss function. The regularized mean squared error (MSE) loss is defined in Equation 11, where where λ is the regularization coefficient, $\|\cdot\|_2^2$ represents the squared L_2 norm of the parameters, and $\mathcal{D}$ is the training dataset. Each term in the summation represents the contribution of a specific parameter to the overall regularization penalty. Three separate regularization coefficients were used. λ_1 penalizes the bias terms, λ_2 penalizes the latent factors, and λ_3 penalizes the temporal terms. After experimentation, it was found that $\lambda_1 = 0.005$ worked best with $\lambda_2 = K * \lambda_1$. In this case, the latent dimension K was chosen to be 10 also after experimentation. The MSE did not seem to decrease much over the test set after $K = 10$. λ_3 was chosen to be 0.02 after also experimentation and was only used for the temporal models.

$$\begin{aligned}\mathcal{L}(\mathcal{D}) = \text{MSE}(\mathcal{D}) &+ \lambda_1 \left(\|\beta_u\|_2^2 + \|\beta_i\|_2^2 \right) + \lambda_2 \left(\|\gamma_u\|_2^2 + \|\gamma_i\|_2^2 \right) \\ &+ \lambda_3 \left(\|\beta_{u,drift}\|_2^2 + \|\beta_{u,yearly}\|_2^2 + \|\beta_{u,weekly}\|_2^2 + \|\beta_{i,drift}\|_2^2 + \|\beta_{i,yearly}\|_2^2 \right) \\ &+ \lambda_3 \left(\|\beta_{i,weekly}\|_2^2 + \|\gamma_{u,drift}\|_2^2 + \|\gamma_{u,yearly}\|_2^2 + \|\gamma_{u,weekly}\|_2^2 \right)\end{aligned} \tag{11}$$

Training Procedure. All models, except the trivial baseline, were trained using PyTorch with mini-batch gradient descent. The training procedure optimizes the regularized loss function (Equation 11) by updating the model parameters iteratively. The Adam optimizer was used in this work, which calculates a different learning rate for each parameter for faster convergence. Additionally, the *OneCycleLR* learning rate scheduler was used in PyTorch [8] which helped speed up model convergence rate using a higher learning rate. All models were trained for 50 epochs with a batch size of 10000 and a learning rate of 0.02 on a single CPU. The training time took about 1-1.5 hours depending on the complexity of the model.

6 Results

This section presents the results of the experiments conducted on the *rateBeer* dataset. The models were evaluated using mean squared error (MSE) on the test set, as described in Equation 5. Additionally, insights into the most biased beers and users were derived from the learned parameters of the temporal latent factor model.

6.1 Overall Performance

The overall MSE results for the different models are presented in Table 5. As expected, the trivial baseline, which predicts the mean rating for all items, performed the worst with an MSE of 0.704. The bias-only model significantly improved the MSE to 0.300 by incorporating user and item bias terms. The latent factor model further reduced the error to 0.286 by capturing latent user and item interactions.

Introducing temporal dynamics improved the performance of both the bias-only and latent factor models. The temporal bias-only model achieved an MSE of 0.292, while the temporal latent factor model outperformed all other approaches with an MSE of 0.283. These results clearly demonstrate that incorporating temporal dynamics, such as temporal drift and periodic trends, improves rating prediction accuracy.

Method	MSE
Trivial baseline	0.704
Bias-only model	0.300
Latent factor model	0.286
Temporal bias-only model	0.292
Temporal latent factor model	**0.283**

Table 3: Overall rating prediction mean square error (mse) results on the rateBeer test set. Ratings are normalized (0, 5].

6.2 Bias Analysis

We analyzed the most biased beers and users by looking at the highest and lowest 3 bias terms for β_i and β_u. The beer known as *3 Fonteinen Framboo* seemed to consistently be rated highly, making it have a higher than average score regardless of the user's preferences. *General Generic Beer* had the most negative item bias of -2.060, indicating that there were consistent low ratings by all users.

Beer Name	β_i
3 Fonteinen Framboos (Framboise)	1.267
De Dolle Stille Nacht Reserva 2000	1.066
Kuhnhenn Bourbon Barrel American Imperial Stout	1.049
PC 2.5 g Low Carb	-1.814
Pabst NA	-1.896
General Generic Beer	-2.060

Table 4: Most biased beers (top and bottom 3) based upon β_i from the trained temporal latent factor model.

The user known as *hfekry99* tended to consistently hand out high ratings of the beers they drank with a bias of 1.902. This high user bias could mean many things. One plausible explanation for such a high bias is that the user only drank types of beers that they knew they loved and did not go out of there comfort zone often. *DrinkForFun* had the most negative user bias of -1.795, indicating that they often rated beers critically. This user may have consistently gone out of their comfort zone and tried beers they didn't like. Another explanation could be that they simply have high standards for beers and hardly give out reviews that are above average.

User	β_u
hfekry99	1.902
lllChirslll	1.773
bunz	1.706
woodsboy	-1.618
thehammer	-1.650
DrinkForFun	-1.795

Table 5: Most biased users (top and bottom 3) based upon β_u from the trained temporal latent factor model.

The inclusion of temporal dynamics contributed to the superior performance of the temporal latent factor model. By modeling temporal drift, yearly trends, and weekly patterns, the model captures evolving user preferences and item popularity over time.

7 Conclusion

This work uncovered several temporal trends in the *rateBeer* dataset and showed that rating prediction can be improved by accounting for these trends. We used a latent factor model and accounted for temporal drift and periodic temporal dynamics to lower overall mean squared error on a reserved random subset of *rateBeer* reviews. The temporal latent factor model outperforms static latent factor models and bias-only models, showing temporal dynamics should be utilized for more accurate modern day recommendation systems.

Acknowledgments

Thank you to Professor McAuley and all of the TAs for their guidance this quarter.

References

[1] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *CoRR* abs/1708.05031 (2017). arXiv:1708.05031 http://arxiv.org/abs/1708.05031

[2] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Knowledge Discovery and Data Mining*. https://api.semanticscholar.org/CorpusID:3022077

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. https://doi.org/10.1109/MC.2009.263

[4] Julian McAuley. 2024. Recommender Systems Slides Week 3-5. https://cseweb.ucsd.edu/classes/fa24/cse258-b/slides/recommendation.pdf

[5] Julian McAuley and Jure Leskovec. 2013. From Amateurs to Connoisseurs: Modeling the Evolution of User Expertise through Online Reviews. arXiv:1303.4402 [cs.SI] https://arxiv.org/abs/1303.4402

[6] Julian McAuley, Jure Leskovec, and Dan Jurafsky. 2012. Learning Attitudes and Attributes from Multi-Aspect Reviews. arXiv:1210.3926 [cs.CL] https://arxiv.org/abs/1210.3926

[7] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems*, J. Platt, D. Koller, Y. Singer, and S. Roweis (Eds.), Vol. 20. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2007/file/d7322ed717dedf1eb4e6e52a37ea7bcd-Paper.pdf

[8] Leslie N. Smith and Nicholay Topin. 2018. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. arXiv:1708.07120 [cs.LG] https://arxiv.org/abs/1708.07120