Review Scheduling for Maximum Long-Term Retention of Knowledge

Stephen Barnes (stbarnes) ◇ Cooper Gates Frye (cooperg) ◇ Khalil S Griffin (khalilsg)

Many people, such as language-learners and medical students, currently use spaced repetition software to learn information. However, it is unclear *when* the best time to repeat is. In order to improve long-term recall, we used machine learning to find the best time for someone to schedule their next review.

Prior Work

The mathematical investigation of human memory was first undertaken by Ebbinghaus, who in 1885, while experimenting with memorizing pairs of nonsense syllables, discovered a mathematical description of memory he called the "forgetting curve" (see en.wikipedia.org/wiki/Forgetting_curve). This knowledge was first applied by Sebastian Leitner, who devised the Leitner flashcard scheduling system (see en.wikipedia.org/wiki/Leitner_system).

Leitner's system was improved on by Piotr Wozniak, a researcher in neurobiology, who wrote the first spaced repetition software (see en.wikipedia.org/wiki/Piotr_Wozniak_(researcher)) as part of his quest to learn English. Wozniak has been profiled by Wired (see archive.wired.com/medtech/health/magazine/16-05/ff_wozniak).

There are now several open-source spaced repetition implementations, such as Anki, Mnemosyne, and Pauker. These are very popular among people learning languages, and have also found a following among medical students. Wozniak and others have continued to attempt to improve their scheduling algorithms, though there is no agreement as to the optimal scheduling algorithm; current implementations use different algorithms.

More information on the practice of spaced repetition can be found in Wozniak's essays (www.supermemo.com/english/contents.htm#Articles). For information on the theory of spaced repetition, see the following publications:

- Toppino, T.C., & Bloom, L.C. (2002). The spacing effect, free recall, and two-process theory: A closer look. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 28*(3), 437-444.
- Greene R. L. (2008). Repetition and spacing effects. In Roediger H. L. III (Ed.), Learning and memory: A comprehensive reference. *Cognitive Psychology of Memory, 2*, 65-78. Oxford: Elsevier.
- Cull, W. L. (2000). Untangling the benefits of multiple study opportunities and repeated testing for cued recall. Applied Cognitive Psychology, 14, 215–235.
- Woźniak, P. A., & Gorzelańczyk, E. J. (1994). Optimization of repetition spacing in the practice of learning. *Acta Neurobiologiae Experimentalis*, 54, 59-62.

Dataset

We used a data provided by the open-source Mnemosyne Project. This data tracked many (on the order of millions) of flashcards and people reviewing them up to hundreds of times over years. After each time they reviewed a card, the students rated themselves on a score of 0-5 with how comfortable they were. The dataset also included other information for each review, such as the time at which it was performed, and the amount of time it took between seeing the card and getting the answer ("thinking time"). For example entries, see the attached data.zip.

Problem Definition

Given a student's history on a flashcard, we wanted to give the best time for them to next review it in order to maximize their long-term recall. In order to do this, we use the following assumption based on the prior work: The best time to review something is right before you would forget it. Thus, we want to find the time at which the information will be "forgotten." We split this task into two parts:

1. Write a function that takes the history of reviews for a single card, and a time in the future, and returns the most likely possible scores at the given time for that card.
2. Write a function that uses the function from part (1) to find the time at which the expected score crosses a certain threshold that we define as "forgetting."

Methods

For the prediction task, we used a feature vector and logistic regression. They operated on a review history, and classified it into one of the score buckets (0-5). We used the scikit-learn library in order to perform these tasks.

We used 7,563 features, including:

- Number of times reviewed, together with a discretization feature template (buckets for 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, and 512 reviews)
- Average score, and frequency of previous scores
- The scores for every previous review, along with the index of that, for example: "A score of 5 on the most recent review" would be encoded as (HISTORY_BACK_SCORE, 1, 5)
- Average thinking time, together with a discretization feature template
- A combination of thinking time and score for each previous review
- The average score of entries that fall into each time bucket
- Combinations of some of the above features

For part 2, we used an unbounded binary search over different times, finding the projected score at those times using part 1. We returned the time at which we predicted the score had a high likelihood of being below a threshold of "forgetting."

Evaluation

Since part 1 was the bulk of the project, we focused our evaluation on that. For our test data, we took existing histories from the database, removed the last score, and tried to predict what that score was. For example, if a history consisted of a score of 1 at time t1, 2 at time t2, and 5 at time t3, we would try to predict the score at t3 given history [(t1, 1), (t2, 2)].

Over the course of the project, we used three different evaluation functions. The first was a Bayesian score based on the probability we assign a result, using the function -100log(1/p), where p is the probability we assigned the correct result. This score could range from negative infinity to zero.

For additional insight, we decided to use a modified F_1 score. We created a confusion matrix, and then calculated and F_1 score for each of the classes. Then, we calculated a weighted sum of these F_1 scores with the weights for each score being the fraction of the data that had that score. A perfect result would get a score of 1, and a perfectly wrong result would get a score of 0.

This F_1 score revealed another problem. Upon looking at our confusion matrix, we noticed that there were many times when we were close, but not quite correct. For example, we often guessed 4 when the correct result was 5. Under the F_1 score, this counted as a miss, and was just as bad as guessing a 1 when the correct result was 5.

To fix this problem, we used a score that was simply the negative of the average square error from all of our data. For example, if we predict a 4 and the actual is a 5, then the score is $-(4-5)^2 = -1$. However, if we predict a 1, the score is $-(1-5)^2=-16$. This way, we penalize scores that are very wrong more than scores that are close. As a concrete example: if we have the following pairs of (guess, actual): [(0,0), (3, 1), (5, 3)], the score would be $-((0-0)^2 + (3-1)^2 + (5-3)^2) = -(0+4+4) = -8$. A perfect result would get a 0, and the theoretical minimum would be -25.

Results and Analysis

We compared our results to a baseline and an oracle. The baseline was a simple model that predicted a probability distribution based on the counts of scores in the history, with LaPlace smoothing. For example, if the history had scores [1, 1, 2, 3], it would predict 0 with 10% probability, 1 with 30%, 2 with 20%, 3 with 20%, 4 with 10%, and 5 with 10%. The oracle predicted the correct answer all of the time.

Regression	**Correct:0**	**Correct:1**	**Correct:2**	**Correct:3**	**Correct:4**	**Correct:5**
Guess:0	**8**	2	2	1	3	5
Guess:1	1	**16**	3	1	5	1
Guess:2	72	149	**330**	118	116	54
Guess:3	8	24	22	**66**	44	8
Guess:4	209	243	637	451	**1455**	559
Guess:5	65	50	81	63	203	**433**

Figure 1.1: Confusion matrix from a run on 5557 test histories

Train Errors	-100log(1/p)	F_1 Score	$-(\text{guess-actual})^2$
Baseline	-235.233237218	0.362754967242	-3.76099028228
Our Regression	-233.10247254	0.388815248766	-2.59941385161
Oracle	0	1	0

Figure 2.1: Scores for Baseline, Oracle, and our function for training examples

Test Errors	-100log(1/p)	F_1 Score	$-(\text{guess-actual})^2$
Baseline	-234.108947535	0.357605298634	-3.68292244017
Our Regression	-239.110819526	0.356793523134	-2.62623717833
Oracle	0	1	0

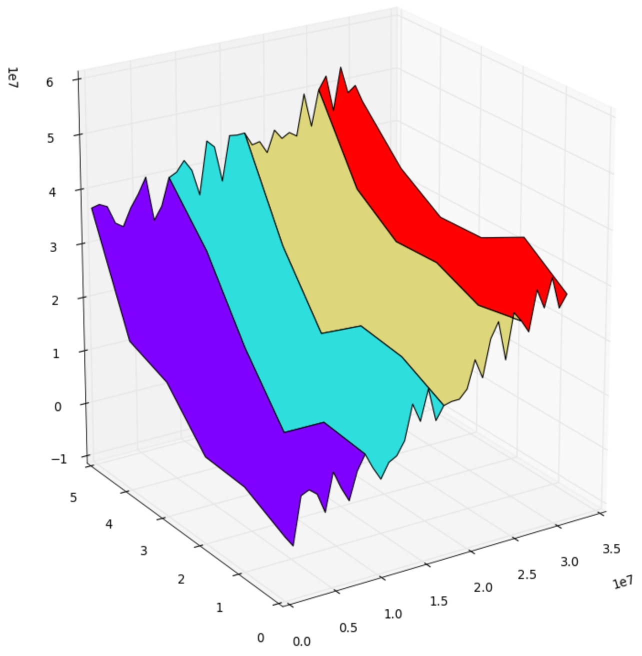
Figure 2.2: Scores for Baseline, Oracle, and our function for training examples

The training and test results are close together, showing that we did not overfit.

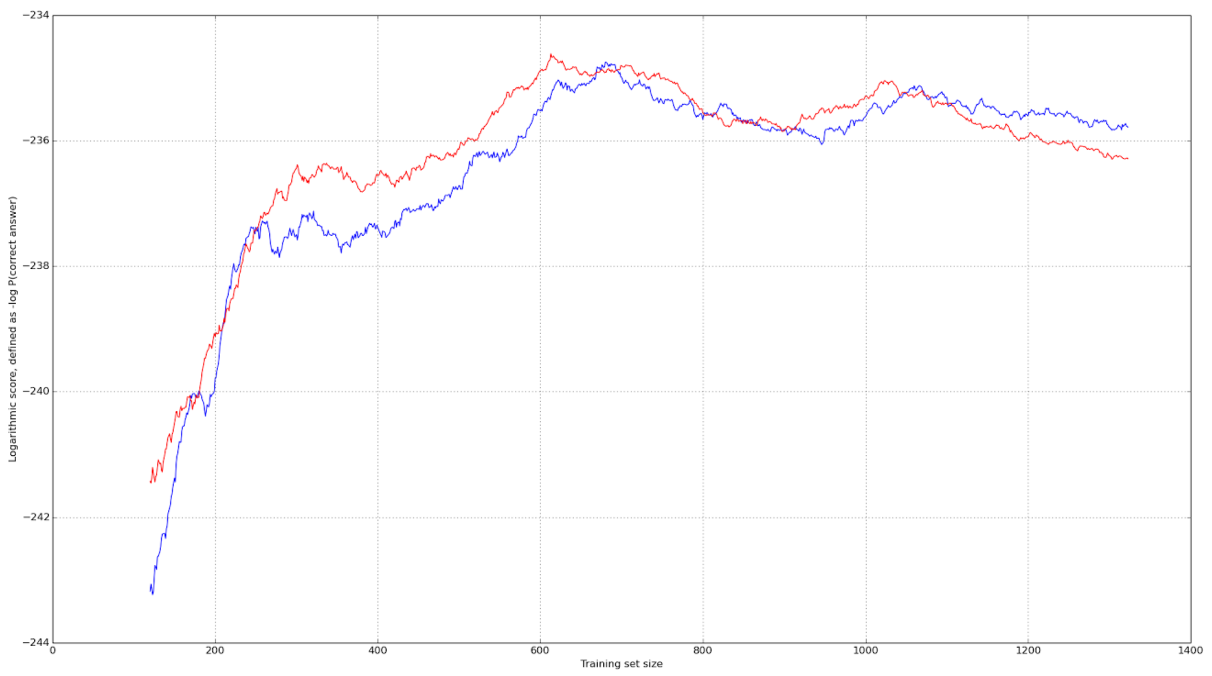
Unfortunately, our model was not able to do much better than the baseline. With both the log score and the F_1 score, we were very close to the baseline. This implies that for both the probability distribution, and the exact classification tasks, our features were not able to to perform the task well.

However, we were able to make noticeable improvement on the score based on the magnitude of our error. This shows that, even if we weren't able to get exactly correct more often, we were able to get *closer* to the correct answer than the baseline. For this application, that improvement is important; we want to be able to predict as close as possible to the actual score, getting closer, while not as good as getting exactly correct, is helpful.

Data Trends



To build intuition, we used visualizations of our algorithms' behavior. For instance, this 3-dimensional graph shows the recommended time to the next review (on the vertical axis, in seconds) against the time taken for the previous review (on the bottom-right axis, in microseconds) and the grade of the previous review (on the bottom-left axis). As expected, greater grades generally increase the inter-review time, though the surface is not entirely smooth. The plot below is a learning curve for our linear regression model (the first nontrivial model we tried), showing test score (blue) and training score (red) on the vertical axis, plotted against the dataset size on the horizontal axis. As can be seen, the score generally increases as more data is added. This graph also shows that adding more data does not substantially improve the algorithm's score past a threshold at around 600 data points. This shows that the main problem with the linear regression model is *bias*, not *variance*. For this reason, we decided to work on better features, which reduces bias by increasing the size of the hypothesis space. We also decided to use a better model (logistic regression), which further reduced the bias.



Insights

The feature weights provided some insight into the importance of various features:

- The time each review took had little predictive value as to the next grade.
- The model generally predicted improvement: if the previous score was a 0, 1, or 2, it gave a high likelihood of scoring a 3. If the previous score was a 3, 4, or 5, it gave high likelihoods for 4 and 5. However, if the second-most-recent score was a 0, it gave a very high likelihood of staying a 0.
- The previous scores only mattered a short ways back. For example, the scores from 50 reviews ago had very small weights.
- The previous score frequency mattered very little. Even users who score fifty 0's previously didn't have an increased likelihood of scoring a 0 next time.

Conclusion

In conclusion, we were able to improve significantly on our baseline, in terms of the predicted grade as well as the overall distributions we predicted. However, our baseline was not too sophisticated, so we suspect there is still a fair amount of room for improvement. The second component of our project is currently based on some assumptions from prior research. While that makes it more well-founded, it also doesn't give us an independent method to compare to the previous ones.

For future work, there are a couple of areas/approaches that seem particularly promising given more time and/or resources.

- Experiment with multiclass classification methods other than logistic regression
- Find ways to evaluate our performance on part 2 indirectly. For instance, by developing a way to quantify how similar a card's real review history was to what we would have suggested. Then, given a particular proposal for a scheduling algorithm, you could see whether ones similar to its proposal did better on average, and to what extent.