Can a Machine Learn to Teach?

Brandon Rule 05356629

December 16, 2011

1 Introduction

Computers have the extraordinary ability to recall a lookup table with perfect accuracy given a single presentation. We humans are not so fortunate. To learn, we must see the entries of a lookup table many times. However, it is not sufficient, nor efficient, to simply see the entries several times in one sitting. We must repeatedly be reminded of an entry at spaced intervals. The spacing is not arbitrary: if we wait too long, we forget the entry; not long enough, and we waste time with a familiar entry. The spacing is also not constant: more difficult entries must be reviewed more frequently. The process of learning a lookup table in this manner is called *spaced repetition*.

The goal of spaced repetition is to maximize the number of lookup table entries stored in a student's memory at a given time. However, it is not possible to have complete confidence that any particular entry is known, so we consider two alternatives:

Goal 1: Maximize the expected number of entries known at a given time.

Goal 2: Maximize the number of entries that we are highly confident the student knows at a given time.

It is not clear that either goal is superior in all circumstances. If the student wants to score well on a simple knowledge retrieval test, then we might argue that we should target the first goal, because this would maximize the expected score on the exam. On the other hand, if the lookup table consisted of the vocabulary for a language, then it might be superior to target the second goal, since the first may be prone to leaving the student with a vocabulary of partially known words across a range of topics, rendering her unable to speak fluently about any single one.

2 Our Model

To clarify the problem, we specify a probabilistic model. We are given a set of students $S = \{a, b, c \ldots\}$ and a lookup table $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Here the set S is arbitrary, and the x_k and y_k are also arbitrary. The reader may take x_k and y_k to be numbers, words, names, or any other objects that a person might be interested in committing to memory. Associated with each student and entry is a history consisting of times $H = (t_0, t_1, \ldots) \in \mathbb{R}^{\mathbb{N}}$ (more accurately, a sequence of elements of an affine space acted on by $\mathbb{R}$), indicating when the student is exposed to the given entry. For example, suppose Adam is trying to learn Spanish, and has seen the flashcard

$$\text{Hello} \mapsto \text{Hola},$$

at 7:00pm, 8:00pm and 10:00pm. In this case, we could represent Adam as student a, the flashcard as entry ("Hello", "Hola"), and the history as $(7, 8, 10)$.

We model the experiment of testing student a on entry (x_k, y_k) at time t given history H using a Bernoulli random variable X whose probability is a function of a, k, t and H. We set $X = 1$ if student a knows entry (x_k, y_k) at time t given history H, and 0 otherwise. We denote the probability that $X = 1$ by $f(a, k, t, H)$. In symbols, we have

$$f(a, k, t, H) \overset{\text{def}}{=} \Pr(X = 1; a, k, t, H).$$

With our new definitions, we see that the task is to construct for each student and entry a history H, given our knowledge of the outcome of a series of Bernoulli experiments. We thus restate goal 1 as follows. Given student a, vocabulary T, and time t, find

$$\arg\max_H \sum_{k=1}^{n} E[X; a, k, t, H] = \arg\max_H \sum_{k=1}^{n} f(a, k, t, H).$$

Goal 2 can be stated using an additional parameter γ, indicating what we mean by "highly confident". For example, we might say we're highly confident a student knows an entry if we believe there is at least a 90% chance that she knows the entry. In this case, we'd set $\gamma = .9$. Given γ, a, k and t, our goal is to find

$$\arg\max_H \sum_{k=1}^{n} 1\left\{f(a, k, t, H) \geq \gamma\right\}.$$

For this project, we focus on the latter goal.

3 Existing Solution

Our data was collected by a program used by a single student to learn the language Xhosa. In this case, the entries of the lookup table consisted of pairs of words indicating the translation from English to Xhosa, for example ("Dog", "Inja"). The program uses a simple algorithm intended to maximize the number of entries with confidence greater than 90%. For each word, the program keeps track of the the student's past performance. For example, if at a given point in time, the student has been presented with a given word five times, answering incorrectly the first two and correctly the last three, the student's performance on the word would be $(0, 0, 1, 1, 1)$. The algorithm associates with a given history a feature called the word's *streak*, defined to be the value and length of the longest constant suffix of the history. For example, the history $(0, 0, 1, 1, 1)$ would have streak $(1, 3)$. Intuitively, this says that the student has answered the word correctly the past three times in a row. The history $(1, 0, 0)$ would have streak $(0, 2)$, indicating she has answered incorrectly the past two times in a row.

Associated with each type of streak that has occurred, the program stores a number indicating the number of milliseconds that it should wait before presenting the student with any word with the given history. For example, if the student has history $(0, 1, 1)$ for a particular word, the student last saw the word at 8:00pm, and the program has a time of 1 hour associated with the streak type $(1, 2)$, then the the student will be scheduled to see the given word again at 9:00pm. Note that the repetition interval selected by the algorithm is purely a function of the streak of a particular word, taking no other features of the word or its history into account.

In order to target the goal of maximizing the number of words with confidence above 90%, the program tunes the times associated with the various streak types as follows. Whenever the student answers correctly after a given streak type, the time associated with the streak type is multiplied by 1.01. When the student answers incorrectly, the streak type is multiplied by 1.01^{-9}. Thus, if a student is answering correctly after a given streak type 90% of the time, then on average, out of 10 answers, 9 will be correct, and 1 will be incorrect. Thus, the time will be multiplied by

$$1.01^9 \cdot 1.01^{-9} = 1,$$

causing the time to oscillate. If she is answering more than 90% of the words correctly, the time will increase until it starts to oscillate. Similarly, it will decrease if she answers correctly less than 90% of the time. The data demonstrates that this technique appears to work well: in a history consisting of 16,744 answers, we observed that the student answered words correctly 88.3% of the time, on average.

However, this model takes into account only a single feature of the word: its current streak. It makes no distinction between histories (0) and $(1, 1, 1, 1, 0)$. We decided to investigate the impact of other features on the probability of answering a word correctly.

4 Testing other features

Given our data of 16,744 answers across 964 words, with times selected by the algorithm described in the previous section, we trained a logistic regression algorithm to predict whether the student will answer a word correctly given the word's history, testing the predictive capabilities of various features. However, it wasn't possible to treat all histories uniformly, because the way that times were selected was not uniform. For instance, we initially attempted to find a correlation between time since last seeing a word and probability of answering correctly. It was difficult to find any correlation. However, this was to be expected, because the times were carefully selected by an algorithm to target a 90% probability of answering correctly.

To overcome this bias, we split the data according to streak types. This way, within a single streak type, there is no bias as to how the time was selected. We then tried various features to determine which might have an impact on the probability of answering correctly. Although we tried more than a dozen features, only a few ended up being predictive. We give seven here, though as we'll see in the data, not all of them were particularly predictive.

- The time since the student last saw the word
- The number of times the student has answered the word incorrectly
- The number of times the student has answered the word correctly

- The longest streak of incorrect answers the student has had for the given word
- The number of times the student has answered the given word incorrectly after a streak of the current type
- An exponentially weighted count of times the student has answered the current word correctly. Answering correctly the previous time counts for 1, the time before for γ, before that γ^2, etc. We found $\gamma = .8$ to be most effective.
- An exponentially weighted sum of the total amount of time the student has gone between seeing the word while still getting it correct.

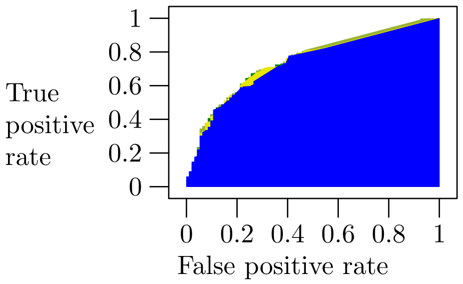
We tested the features using 70%/30% hold-out cross validation, using the area under the ROC curve as our metric. To select features for a particular streak type, we used forward search.

5 Results

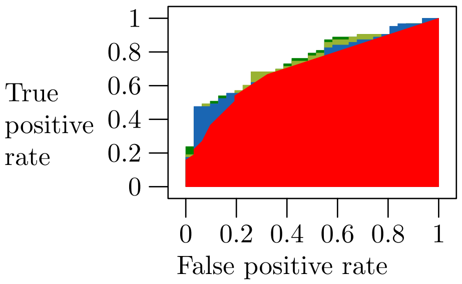
We present our results in the Figure 5.1. We note that for different streak lengths, different features tend to be more predictive. For short correct or incorrect streaks, we see that the exponentially weighted count of correct answers, as well as the longest wrong streak, tends to be indicative, while for long correct streaks, the simple count of total wrong answers for the word tends to be most indicative.

6 Future work

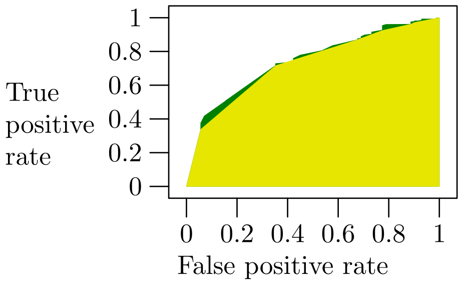
In future work, we'd like to try to incorporate the features we tested into a new model for selecting times to show a word. It would also be interesting to attempt to come up with a model that optimizes goal 1, the expected number of words known. It would also be useful to collect data that is not influenced by a selection algorithm, since this would allow us to test whether the streak length itself was a good feature to use.



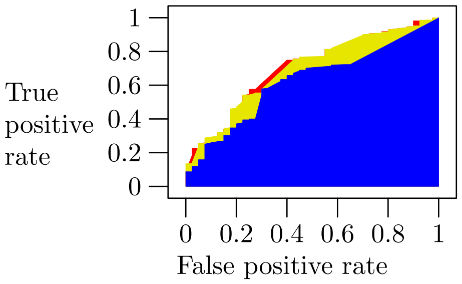
(a) Wrong streak of 1



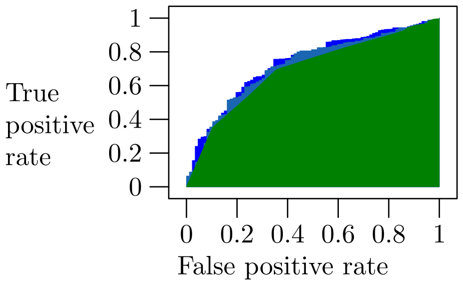
(b) Wrong streak of 2



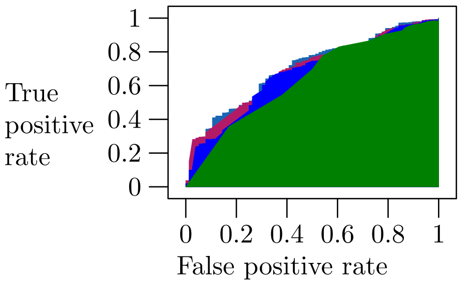
(c) Right streak of 1



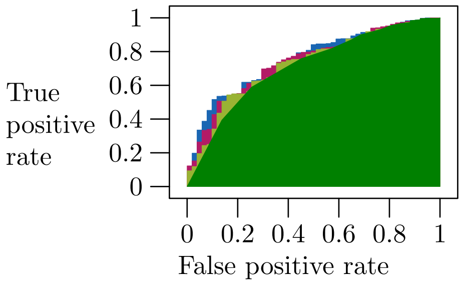
(d) Right streak of 2



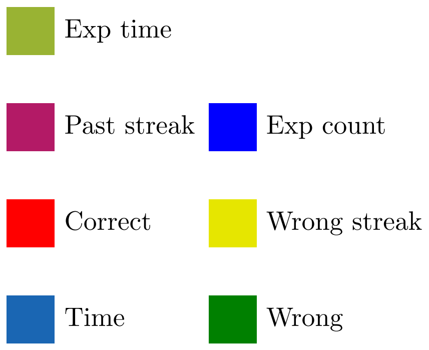
(e) Right streak of 3



(f) Right streak of 4



(g) Right streak of 5



(h) Legend

Figure 5.1: ROC curves for different types of streaks