KNEWTON DATA CHALLENGE

JOHN WANG

1. Problem Formulation

Let there be n individuals taking an exam each year, each of whom take k of the K total questions available. We require that L of the K questions be offered to at least one student, and also that $0 < k < K$ be satisfied. In this paper, I will examine strategies for finding a ranking of students given a previous year's results as training data.

2. Measuring Question Difficulty

In order to do this, we want to formulate some measure of the difficulty of each question j. This motivates our examination of r_j, the probability that a student will get question j correct. To estimate r_j, one could naively use the sample mean from the training data for each question:

$$r_j \approx \frac{1}{n_j} \sum_{i=1}^{n} x_{ij} \tag{1}$$

Where x_{ij} denotes whether or not individual i answered question j correctly and n_j is the number of times question j was asked in the training data. This scheme works as long as the questions were assigned uniformly at random. However, as soon as there exists dependence among the questions, then this technique no longer works. If some set of questions q_1 are assigned to a group of students s_1 with higher probability than other questions and s_1 has a higher intelligence level than the average student, then r_j for $j \in q_1$ will be biased upwards.

Therefore, it is necessary to introduce a new variable θ_i which captures the intelligence level of student i. The probability that student i answers question j correctly will now have an additive factor of θ_i and will be given by $r_j + \theta_i$. If $r_j + \theta_i \geq 1$, then student i always answers j correctly, and if $r_j + \theta_i \leq 0$, then student i never answers question j correctly. One can see that $\theta_i = 0$ implies that student i is average, whereas $\theta_i > 0$ implies above average intelligence and vice versa for $\theta_i < 0$. Let $\gamma_{ij} = 1$ if student i was assigned question j and $\gamma_{ij} = 0$ otherwise. We will try to minimize the distance to the probability prediction:

$$\sum_{i=1}^{K} |x_{ij} - (r_j + \theta_i)|\gamma_{ij} \tag{2}$$

Using this distance function, we will attempt to estimate r_j for all j and θ_i for all i with the following algorithm. We start by initializing $\theta_i = 0$ for all i and $r_j = \frac{1}{n_j} \sum_{i=1}^{n} x_{ij}$ for all j. Now will will loop for T iterations. Suppose we are in iteration $t > 0$ and $t \leq T$. We will first update all intelligence values θ_i for all i by estimating $\sum_{j=1}^{K} |x_{ij} - (r_j + \theta_i)|\gamma_{ij}$. We will also estimate this with θ_i replaced with $\theta_i + \Delta_t$ and $\theta_i - \Delta_t$. The algorithm then replaces θ_i with $\theta_i - \Delta_t, \theta_i, \theta_i + \Delta_t$ depending on which one gives the smallest distance. We do this for all i and do the same for r_j by calculating $\sum_{i=1}^{n} |x_{ij} - (r_j + \theta_i)|\gamma_{ij}$ for $r_j - \Delta_t, r_j, r_j + \Delta_t$. We do this for all j and finish the iteration.

To calculate the distance offset Δ_t at each iteration t, we will use the function $\Delta_t = \frac{1}{2}e^{-t}$. This allows our distance to decrease at each iteration in an exponential manner. Intuitively this allows the algorithm to explore the sample space at the beginning and narrow down the search in later iterations.

To prove that the resulting r_j and θ_i will be good approximations of their actual values, we will show that our distance to the optimal result is bounded. Let r_j^* be the optimal value of r_j and θ_i^* be the optimal value of θ_i for all j and i. The distance using these values is given by:

$$E\left[\sum_{j=1}^{K} |x_{ij} - (r_j^* + \theta_i^*)|\gamma_{ij}\right] = round(k(r_j^* + \theta_i^*)) - k(r_j^* + \theta_i^*) \tag{3}$$

Here, $round(.)$ denotes the function that rounds . to the nearest integer. The expression follows because there will be k questions for which $\gamma_{ij} = 1$ for each student i, and since there can only be an integer number

of $x_{ij} = 1$. Now, we will show that the above algorithm obtains values that are no worse than kn distance from the optimal. We compute:

$$\sum_{i=1}^{n}\sum_{j=1}^{K}|x_{ij} - (r_j + \theta_i)|\gamma_{ij} \quad \leq \quad \left|\sum_{i=1}^{n}\sum_{j=1}^{K}x_{ij}\gamma_{ij}\right| - \left|\sum_{i=1}^{n}\sum_{j=1}^{K}(r_j + \theta_i)\gamma_{ij}\right| \tag{4}$$

Using the triangle inequality. Since we always decrease our distance function as the iterations occur over time, all that needs to be shown is that the initial value is bounded by kn. We will use the initial values of r_j and θ_i, which means we can write the above expression as:

$$\sum_{i=1}^{n}\sum_{j=1}^{K}x_{ij}\gamma_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{K}\frac{1}{n_j}\sum_{i=1}^{n}x_{ij}\gamma_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{K}x_{ij}\gamma_{ij} - 2\sum_{n=1}^{n}\sum_{j=1}^{K}\frac{x_{ij}\gamma_{ij}}{n_{ij}} \tag{5}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{K}x_{ij}\gamma_{ij}\left(1 - \frac{2}{n_{ij}}\right) \tag{6}$$

$$\leq \sum_{i=1}^{n}\sum_{j=1}^{K}x_{ij}\gamma_{ij} \tag{7}$$

$$= kn \tag{8}$$

From the fact that $n_{ij} > 0$ so that $(1 - \frac{2}{n_{ij}}) < 1$. Therefore, we have bounded how far our algorithm can be from the optimal, since the optimal weights can only have a minimum distance of 0. In practice, however, $k(r_j^* + \theta_i^*)$ will not be an integer for all j and i which means that the distance for the optimal values of r_j^* and θ_i^* will be positive and proportional to n.

This analysis shows that we have found an approximation algorithm that allows us to find the difficulty of questions, r_j, while accounting for student's intelligence. We will now use r_{ij} as a probability measure for selecting the questions to use.

3. Entropy and Question Selection

The main idea of the algorithms to be presented is to maximize the amount of entropy in the set of questions to be asked to students. We will call the pairing of all n students each with k questions to be an examset. The goal will then be to maximize the entropy of the examset. The reason to maximize entropy is because by doing this we maximize the amount of information available. Intuitively, we want to ask questions which will best separate good and bad students.

The information entropy will be a good measure of how separated the questions are. Thus, if we maximize the entropy of the examset, we will have the most information that is possible for ranking students. Let $p_i(s)$ be the probability that student i achieves a score of s. We will define the information entropy of an examset as:

$$\mathrm{H} = \sum_{i=1}^{n}\sum_{s=0}^{k}p_i(s)\log\frac{1}{p_i(s)} \tag{9}$$

Thus, we sum over all possible scores on a test and obtain the entropy for student i's scores, then take the sum over all students to obtain the total entropy of the examset. Maximizing entropy will intuitively produce exam scores which are as close to uniformly distributed as possible, which will provide the most information to use when ranking students. To compute $p_i(s)$, a dynamic programming solution will be used.

Notice that $p_i(s)$ when computed by brute force requires the enumeration of all possible combinations of getting s questions correct. This is exponential in the number of questions. However, one can notice that one can use previously computed probabilities to compute new probabilities. Thus, given a student i and his k questions, we can compute $p_i(s)$ for $s \in [0, k]$ in the following manner. Let $p_i(s, q)$ be the probability of getting a score of s through the first q questions, where $q \leq k$ and the questions are ordered arbitrarily. Then, we can compute $p_i(s, q)$ using the recursion:

$$p_i(s,q) = \begin{cases} p_i(s, q-1)(1 - r_q) + p_i(s-1, q-1)r_q & \text{if } i \leq j \\ 0 & \text{if } i > j \end{cases} \tag{10}$$

Assume that $p_i(s, q) = 0$ if $s \leq 0$ or $q \leq 0$ so that the $p_i(s-1, q-1)r_q$ term falls away if $s - 1 \leq 0$.

4. Greedy Approximation

For the first attempt at a solution to this maximization problem, a greedy solution will be proposed. Since it is possible to calculate the entropies of each question, and it is likely that high entropy questions individually lead to high entropy totals, the greedy algorithm will calculate the top entropies for all questions and select the highest L. From there, a variety of methods will be developed to actually assign the questions to students.

Note that this is only an approximation to the optimal because the highest entropy questions individually do not necessarily provide the highest entropy examset since we are looking for entropy of scores. Thus, high entropy questions will contribute significantly to a higher entropy total score for each student, but it may be possible that questions are dependent, in which case a particular combination of questions could lead to very high entropy.

To compute the entropy H_j of each question j, we use the equation:

$$\mathrm{H}_j = r_j \log \frac{1}{r_j} + (1 - r_j) \log \frac{1}{1 - r_j} \tag{11}$$

Since r_j has already been calculated for each question j, calculating H_j is straightforward. Sorting the questions and taking the top L requires $O(K \log K)$ time. Now, a number of different strategies for assigning questions to students is employed.

4.1. **Randomized Probabilistic Assignment.**

For the randomized probabilistic assignment (RPA) algorithm, probabilities will be given to each question and the questions for each student will be chosen randomly based on the probabilities provided. The crux of the matter is assigning probabilities which lead to valid examsets of high entropy. Assigning very large probabilities to the highest entropy questions means that this is a lower probability that the eventual examset is valid (i.e. has more than L questions assigned to at least one person). If an assignment is not valid, the algorithm will try again to reassign all questions. Keeping the expected number of retries a constant will prevent the algorithm from having too large of a runtime.