IMPROVING THE STUDENT EVALUTION BY PREDICTING STUDENT PREFERENCES

Arbaz Khan (Y9128)
Advised By : Prof. Harish Karnick

May 2, 2012

Abstract

While preparing oneself for an examination, it is always helpful if one could get an estimate of the topic-wise toughness in his/her course of study. Such an estimate would enable to plan the strategy of preparations to maximize the performance for an examination. One can decide upon what areas to study more or which require special attention. Also it could help improve student evaluation in an exam based scenario where users would be tested on their online performances. In order to help get such an estimate, we attempt to predict the probability of answering a question correctly given the past performance of students who have underwent similar tests and answered questions on the same topics. This will not only help students figure out what areas they are weak in but also design tests which help better measure what a student actually knows based on his grade hence evaluated. The data in our research comes from students studying for GMAT, SAT, and ACT at an online test portal, www.grockit.com . Starting with few state-of-the-art techniques. We would attempt to predict, for each question attempted in the test set, the probability of student answering the question correctly. We would introduce an online unsupervised learning algorithm based on the paradigms of Dynamic Bayesian Networks.
Keywords : Topicwise toughness, User Grade, Online Unsupervised Learning, Dynamic Bayesian Networks

1 Overview

1.1 Student Evaluation

By student evaluation, we mean classifying students on the basis of their skills/strengths. Now, there are a wide range of areas of education. Strength of a student in one of them may be very impressive while he may be relatively very weak at the same time on the other. Therefore arises a need of a topicwise student evaluation wherein a student gets assigned a grade for every possible independent fields of his education paradigm (e.g. Commerce, Science) just like how the education institutes function.

1.2 Exploring the problem

Not only does student evaluation as mentioned help a student in getting aware of his standards but such a plan could help designing a model which could give a good measure of quality of a student and his topicwise strengths/toughness. Since its evaluations that we are talking about, the model could be extended to the fields of tutoring too. Now, how to go about to achieve the target? How can one decide upon a student's class as early as possible with sufficient accuracy as well as in the least time of exposure to student? We do it by predicting a student's answer on every question instance posed for hims. If we can somehow get an estimate of what is the probability of his answering the question correct or incorrect, then it could help us get a measure of how well equipped he is in the areas related to that question. The more probable of him answering, the better his grade is.

1.3 Approach

At prior , what approach we wish to conquer is :
Feed a certain number of questions to the user keeping a record on the regulation scoring and predictions of his answers. For quicker evaluation use the concept of related topics in a question. Performance in one would help infering about the others. Using the records mentioned, once we get an estimate

of his class in each such topic, we can go on and deal thereon with the results depending upon our application. E.g. We could stop there and report him qualified or ineligible if the environment was an exam or we could go on making him practice upon his weaker fields if this was a tutoring session.

2 Data Format for Learning

We now introduce the practical details of the problem in the form of the data available for training and testing the algorithm. We have around 5 million answers by as many as 10K users on an online coaching website www.grockit.com. We need to test the data on users of which prior histories of any kind maynt be known. There are some instances in the training set for a good number of users. Therefore the problem stands as a semi supervised learning problem. Also, each data item has labels of specific kind distinguishing it from the others, so we have labelled data.
Each question has few important fields: Question_id, question_type(Free Response/Objective), group_name(ACT, SAT), track_name(ACT Maths, SAT English) , subtrack_name (Comprehension, Problem solving), tag_string(Arithmetic, Geometry, Pronouns).
Each field has certain range of discrete values. Tag strings the lowest and the most important denomination. It basically denotes the most specific fields to which the question belongs. Like knowing that a user is strong in Maths is not as informative as he is strong in Algebra. This is what we would require while classifying the students on the basis of their topicwise grades.
Hereafter, we would explore the following techniques for dealing with the problem and analyse the possibilities with each of these along with what merits and demerits could one get out of each of them. We would finally present our own modification to a state of the art technique.

3 Techniques Explored

1. Naive Bayes
   - Binning Naive Bayes
2. Inductive Logic Programming
   - ESCAPE
3. Bayesian Networks
   - Dynamic Bayesian Networks
     - Improvised Dynamic Bayesian Networks

3.1 Naive Bayes

As the name suggests, its a naive classification algorithm working with the conditional probabilistic relationships using the Bayes Rule. With Naive Bayes, it is all about computing the posterior probability P for classes in C as:

$$P = P(C = Ci \mid V_1 = v_1, V_2 = v_2, ..., V_n = v_n)$$

Here V_K is the field attribute and v_k is its value which belongs to the test example The class which gains the maximum probability is assigned as the class of the test example.
Assuming conditional independence with Bayes rule on P :

$$P(C = C_i | V_1 = v_1, V_2 = v_2, ..., V_n = v_n) = T \times P(C = C_i) \prod_{j=1}^{n} P_j$$

where,
$\boldsymbol{T}$ – a factor constant over the classes
$P_k = P(V = v_k \mid C = c_i)$
P_k is computed as:

$$P(V = v_k \mid C = c_i) = \frac{((P(V = v_k \wedge C = c_i) + l)}{(P(V = v_k) + 1)}$$

,
where $\boldsymbol{l}$ and 1 are balancing factors just to avoid the situation where the probability values are null

which in turn would cause the product of Pi's to be null. In our case, the class C takes values correct and incorrect and V contains the field attributes pertaining to a question as defined previously. Applying Naive Bayes requires the preprocessing on the total answers corresponding to each attribute seperately. We cant apply Naive Bayes classification directly to our problem as we can only get a bernouli output for each prediction. We need to calibrate the algorithm to output probabilities for each class rather than assigning bernouli outputs to the probabilities.

For obtaining calibrated probability outputs from the Naive Bayesian classifier :

1. Start by sorting the training examples according to the probabilities computed by the Bayes rule in a fashion similar to what we did above to classify a test case.

2. Divide the sorted training set into bins of equal size. The bins are nothing but a term for the subsets of the sorted data. For each of this bin we compute and label the range of probability outputs i.e. the minimum and the maximum probability estimates from that bin.

3. When needed to classify a test example , it is assigned a bin according to the range in which its probability output lie and the labels for the bins that we have. It then belongs to class C_i with a probability equaling the fraction of training examples in the bin that actually belonged to C_i.

Intuitive idea is to estimate membership in a class of a test example from the training examples which actually had nearabout the same probability score using Naive Bayes. ie. how did the probabilistically same examples performed earlier.

3.1.1 Advantages:

Forms characteristically an intuitive and a simple approach. It only requires the probability of correctness for each attribute for proceeding with probability estimates.

3.1.2 Disadvantages:

There are few vital disadvantage behind applying Naive Bayes classification to our problem. There occurs high computation cost while preprocessing and sorting while calliberation. This effectively increases the learning time of the algorithm. Also importantly, in Naive Bayes, the criteria for extracting the relations in the data sets is weak. Chance occurences of an instance are given high priority even if no relationship can be inferred from such an occurence. What only affects is the probabilistic statistics derived from the training sets.

3.2 INDUCTIVE LOGIC PROGRAMMING

Inductive logic programming techniques use first order logic for data representation with their numerous algorithms help gain the incontext information from a large range of features amongst which prior relations are unknown. The task of an ILP program is briefly to generate a set of hypotheses that could help explain the structure by covering the positive examples and maintaining consistency requirements by eliminating the negative examples. Amongst the many techniques of ILP, we studied rule induction systems for the purpose of our problem. Rule induction systems help infering rules from a set of observations. Most common strategy being sequential covering. The sequential algorithms which I studied to apply them to the problem were $ESCAPE$ and $CN2$. Every sequential covering strategy works on similar terms as in it does rule learning in a sequential fashion. Learning the most promising rules first which cover the training examples and then iterating over the rest of the examples. Rule learned in the sequence then help classifying a test example by searching for a rule in the sequence in an ordered fashion which classifies it first. We'll analyse here ESCAPE algorithm as a representative of the sequential covering strategies. (Time complexity $O(nAlognA)$ is the best)

3.2.1 IMPLEMENTING ESCAPE

Pruning For processing over large number of attribute value pairs in our data set, one would start with pruning the attribute value pairs based on some entropy measure. As intuition suggests, this should depend upon the frequency of occurence of an attribute value pair in the positive (P_{pos}) and the negative examples (P_{pos}). One such simple entropy measure ($E(A)$) of an attribute value pair would be :

$$E(A) = P_{pos}(A) - kP_{neg(A)}$$

where $\boldsymbol{k}$ is a parameter of degradation

We require a lot of preprocessing to be done here. We have a good number of attribute value pairs and we need to record their occurences in the training set based on the correct and incorrect outcome of the question instance they belong to. $P_{pos}(A)$ is as large as 0.79 and as small as 0.35. $(0.35 < P_{pos} < 0.79)$
One way to represent E in terms of P_{pos} is : $E(A) = P_{pos(A)}(1 + k) - k$
An appropriate threshhold could be set to decide the number of attribute value pairs that could be just enough to achieve a good final result. Parameters in all to be adjusted now become 2. One is the degradation parameter, other is the threshhold for criteria for elimination.

Building binary matrix In the next step, we would build a binary matrix wherein we start sorting the attribute value pairs based on the entropy measure and order them as the columns of this matrix. We would then sort the training examples based on the number of high performing pairs contained in them which become the rows of the binary matrix. This comes from the fundamentals of sequential covering strategies which requires the best rules to be put ahead of the sequence. Building up the matrix like this provides us to make these fundamentals to be met.

Rules are extracted from this binary matrix by a top down traversal taking intersections of the performing pairs till you get an empty set wherin you store the conjuction hence obtained.

An Illustration :
Suppose the snapshot given below which was obtained from the original matrix from the problem were the binary matrix. Then *ESCAPE* algorithm would form the below mentioned rule sequence for explaining the 5 records given below. Note that the columns are ordered by the entropy measure (P_pos written in bold on the top of each column) and the rows are ordered according to the occurences of high performing pairs. If the occurence of an attribute value pair be treated as a bit then the record with a larger binary number comes first.

Table 1: Escape Model Illustration

$P_{pos} \rightarrow$	**0.79**	**0.72**	**0.69**	**0.68**	**0.68**
UserId, QId	Tag = 99	GameType = 0	Tag = 244	Tag = 248	Tag = 60
85104,5367	1	1	0	0	0
42198,5367	1	0	0	0	0
60146,16	0	1	1	0	0
16975,1753	0	1	0	1	0
163150,620	0	1	0	1	0

$$Class = Correct \text{ if } (Tag = 99) \vee (GameType = 0)$$

3.2.2 Advantages:

Help representing the data in a natural way as first order logic is used for representation. Helps finding relations between the class and the field attributes by generating rules and knowing the structure in the data. Very much desirable in our case as though an expert can grant his views on student types but being data specific, its a boon to have the algorithm itself discover the structure. Such strategies could explain a data set in the form of disjunction of a few rules

3.2.3 Disadvantages:

Both rule learning algorithms involve high computational cost. Log linear in terms of the product of number of training examples and the attribute value pairs after pruning. ESCAPE involves sorting the attribute value pairs (columns) based on a performance measure and then sorting on all the examples (rows) according to the those which are contained under decreasing orders of columns. In our case 422 columns and 5M rows. Thats almost 2GB size of memory if no attribute value pairs are pruned and one requires to process the binary matrix in one go. The time complexity for sequential covering algorithms is $n|A|log n|A|$ where $|A|$ is the number of attribute value pairs. $n|A|$ is 2 * 109 in our case. Practical application is quite a tedious task. Classification can be achieved but inferring the probabilistic relations poses another challenge to the ILP techniques. Stress is upon explaining the the data set rather than learning a classifier

3.3 Bayesian Networks

A Bayesian network is a Directed acyclic graph (DAG) form of representation to indicate a relationship that exists between the random variables of a data set.

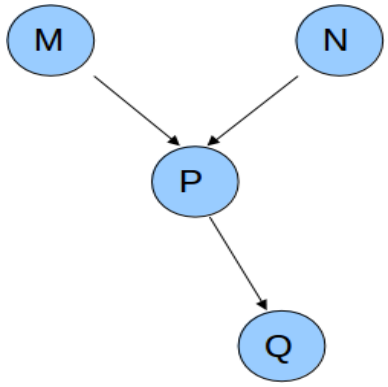

Figure 1: A Simple Bayesian Network

Nodes in the pictorial representation of the Bayesian network represent random variables amongst whom the relationships are to be studied Edges between these nodes represent the conditional dependencies between these variables in the form of probability distributions. If there does not exist an edge between two nodes, then the variables can be said to be in a state of conditional independence, as in the probability distribution of one doesnt affect the other's. These relationships and distributions structure the data in a form wherin complex causal dependencies can be inferred. When one needs to acquire knowledge about a problem domain or needs to make future predictions about some event, he needs to know the degree of causal dependencies in the events associated with it.
Even if the structure of dependencies are unknown, or in other terms one cant figure out the direction of edges in the DAG or the edges themselves, Bayesian Networks still can help in learning the structure as well as the degree of the dependencies. This forms one of the two problems Bayesian Networks are designed to address. Its called learning a Bayesian network and the other is inference. When we have the structure learnt and the dependencies estimated, a lot of useful inferences can be made applied to wide varying domains. We can think about various real world applications wherin we dont have much data or its not feasible to arrange for it beforehand, one could build a network to make useful predictions. e.g. Stock analysis. A lot of variables, substantial theory for expertise in this domain makes it liable for Bayesian Networks for its data extraction.

Using Bayesian Networks for the Problem

To answer this question, we need to look back at the task of our problem. What we eventually want is evaluating students based on their performances and attain some judgements about his weakness and strengths. Evaluation could mathematically mean assigning some grade to a student on a finite relative scale.
Evaluation would be an online process and hence the grade of a user should start with some default value which could be universally same or it could be derived from the past information about the user. e.g. Secondary and Senior Secondary School performances. In the online process, the student would be posed a series of questions for which his response would in turn affect his pre-evaluated grade. For the questions he faces, we cant keep them on similar terms as in answering a question should not necessarily have the same effect as answering some other.
Now to make the Bayesian network, we need to collect the variables. This fortunately is a domain from wherin one would not have much difficulty in deducing the structure. What is visible to us is

his whether he answers a question correct or incorrect. What his answer depends upon is his current grade and the question he faced. We need to parametrize a question instance to fit into our model. What could be best is to allot each question a toughness coefficient on again a finite scale. This also helps when we would decide for the dependencies between the variables. User Grade should keep changing as he keeps on answering the questions correctly or incorrectly. With this, the value of user grade needs updation after every response. This updation depends upon what our prediction for his answering the question was and what the actual outcome is. The greater the difference, the higher magnitude of updation should be. Also, it depends upon the question he answered as incorrect answer for a tough question should have lesser effect on an intermediate student as compared to an incorrect answer of an easy question. And most importantly, the next grade for an instance depends on the current grade as well. You may have noticed the term 'intermediate' in the above sentence which clearly denotes what effect the current grade has on the evaluation scheme.
Also the toughness coefficient shouldnt be kept static ideally as questions do lose their toughness with time due to sufficient theories made available with time but the effect can be considered very gradual but not static.
Given the non-conventional behaviour our variables exhibit, should we really go for simply a Bayesian network to model the domain knowledge. We definitely require a time series modelling for these dynamic variables. Fortunately, we do have a way out.

3.4 Dynamic Bayesian Networks

Dynamic bayesian networks help us in time series modelling of variables. It operates in sequences of slices which are in themselves a bayesian network. There are edges intra to the slices as well as inter amongst the slices. Inter edges represent a phase of transition and the edges intra to the slices represent a state in operation. We allow edges only within consecutive slices assuming a markov process for simplification i.e. each state of a variable depends on the state in immediate past and affects immediate future. Each state is a set of observed variables and latent variables. For our problem, the outcome of the question is an observed variable and rest are inherently latent as in we dont have any information visible from the problem. A simple illustration of a dynamic bayesian network is given below:

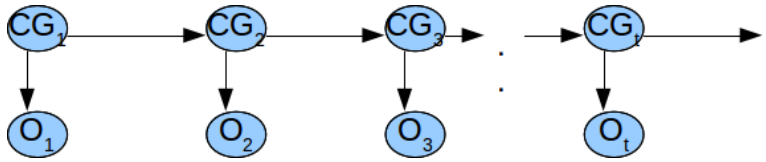

Fig. A simple Dynamic Bayesian Network

Each O_t in every time slice is an observed variable and depends upon the latent variable CG_t which in turn depends upon CG_{t_1}.
This is a fairly simple Dynamic Bayesian network. We need to figure out the complexities related to our structure before we could model it.

3.4.1 Exploring the dependencies :

Given a response of a user occured at a time t_1. For grade updation, user Grade CG_t depends upon the previous grade CG_{t_1}, the toughness coefficient QT_{t_1} of the question at that response and the actual response O_{t_1} of the user to help us find the difference between our prediction and the actual response. Outcome Ot at a particular time depends upon the user answering it, hence the user grade and also the toughness coefficient of the question being answered QTt. This is what forms the prediction strategy for a given question and this is what we are actually aiming to excel in.
However, the variable QT, would exhibit a strange behaviour if we try it to model as a node in our network. A user in his time series goes on to answer a sequence of questions and its almost sure that he wont come across the same question again. Updation of QT at a time t would occur from its current toughness, the actual response occured for this instance of the question and the user who went about answering it. The last two variables are clearly Ot and CGt. The dependence of future toughness on current toughness cant be given as a representation in a simple Dynamic Bayesian Network as QT would have dependency edges from the time series modelling of different variables. It would be like edges hovering over various simple Dynamic Bayesian Networks representing the dependency of just

one variable QT. It can only be labelled as QT_{prev} , as the previous occurrence of this question cant be generalized in every case to some t.
This abnormal behaviour of QT and the fact that the length of sequence i.e. tmax can not be defined in our case as a student evaluation can go on for long and indefinitely, makes us to deviate from the standard norms of a Dynamic Bayesian network, However the dependencies inferred and the model in simulation appeals us to come up with a probability function as required and exploit the properties of the structure in a fashion similar to Dynamic Bayesian Networks. To use the structure and sample the data from it to make predictions for the observed, we need to provide it a probability function for the variables involved.

3.4.2 Designing the probability functions

Before going further to design the probability functions, we need to ask ourselves what domain knowledge can we use? Since we are in a state of an unsupervised online learning, as in we would make predictions without any training data available about the grade and outcome relationship or the toughness and outcome relationship, domain knowledge would play an important role, also in the data samplings and inferences.
We start with the earlier mentioned scalings of user grade and the toughness coefficient. User grades can be scaled on a scale of 1 to 100 and toughness coefficient in 1 to 10. User grades are provided a finer tuning as we would want that each outcome have some effect on the user grade and that it should not be discarded just to round it off to the nearest scale point. On the other hand, toughness coefficients could be managed in a coarser detail. This intuition comes from the fact that there are less varying toughness levels of questions than the the varying quality of students at some local platform where we would use our algorithm.
Since we are the one who would decide the evaluation criteria, if we can assign a toughness coefficient to a given question, than we can always put forward the expectation that the user of this grade must have this probability of answering it.
One would always want to assign a question as the easiest if maximum number of users can solve it and have round about equal probability of answering it. Similarly the toughest question is the one which almost nobody could be expected to solve and hence again asks for equal probabilities. A moderately difficult question is the one which can be approximated to have a linear relationship with the probability of answering the question. With these in mind, we can now formularise our heuristics.

For Outcome O_t :
Since the outcome, probability of correctness**(P)** as mentioned depends upon user grade and the toughness coefficient in a manner described above, we came up with a mixture of cumulative beta distributions that could capture the above intuitions. There cant be any theory for generalizing the best distribution, we can only observe the results and evaluate the performances. And this is what exactly we do here. Our function for estimating the probability function :

```
if toughness < 5:
.   y = special.btdtr (int(toughness)/div, 5/div, float(x)/100)
else:
.   ŷ = special.btdtr (5/div, (11 - int(toughness))/div, float(x)/100)
```

where,
y is the probability of a correct outcome (**P**) , **x** is User Grade
special.btdtr is a special function representing the integral from 0 to x of beta distribution function
div is the flattening parameter

Given below is a sample output of the above function for $div = 8$.

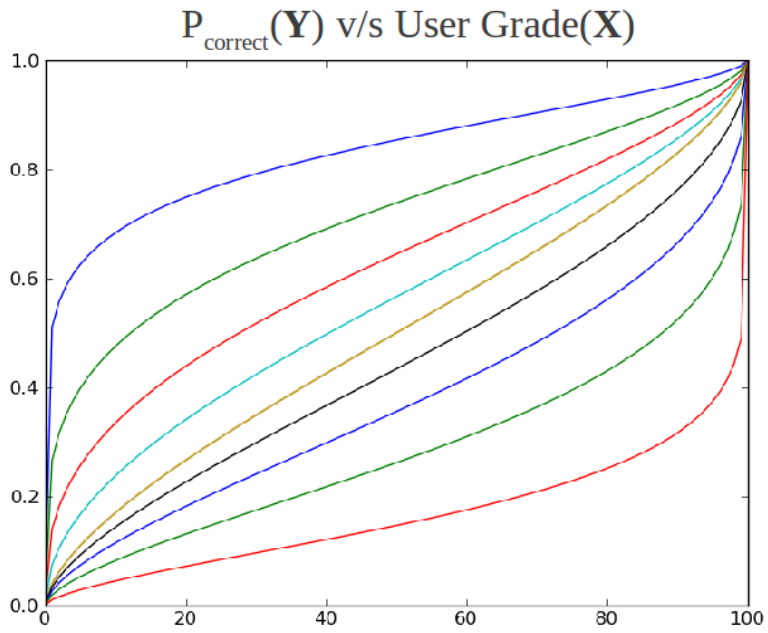

Fig. Modelling the Real World

The curves at the top of the graph are for the probability of answering a question of low QT values correct as a function of User Grade. While the curves at the bottom represent the same for questions with higher QT values. In brief, QT increases as we move to the bottom of the graph.
As we can see, what the above graph basically captures as that very low QT values as well as very high QT values individually offer almost similar probabilities for most of the User Grades. While for the intermediate values, QT increases almost linearly with User Grades. Thus, it quite desirably captures our intuitions based on real world scenarios.
We ran the final algorithm for various adjustments of parameters trying to visualize the nature of performance with these parameters. The best results appear to form a local optima with flattening parameter $div = 5$ as shown below.

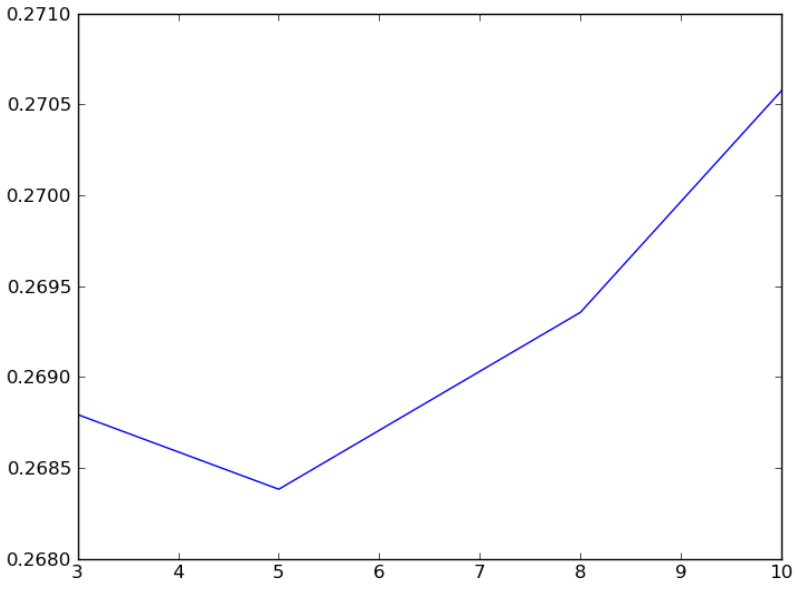

Fig. An analysis on the *div* parameter(**X**) v/s binomial capped deviance of the predictions (**Y**)

For User Grade:

Since we would not always have enough exposure to a user by posing him a good enough number of questions,we should aim for converging to his grade as quickly as possible. This clearly calls for simulated annealing wherin we would provide good enough magnitude of fluctuations in the start, to quickly move a user to a point around his actual grade. As the number of questions answered increases

with time, we would reduce this magnitude exponentially to ensure convergence.
As for the first attempt to such a function, we came up with :

$$CG = CG + a \times (delta) \times e^{\frac{-t}{c}}$$

where,

delta is the difference coefficient linearly proportional to the difference in prediction and actual outcome.

t is the number of questions answered till now

a is what we call factor coefficient governing the difference magnitude

c being another constant which determines rate of convergence

Introducing further enhancements, we achieved better results for if we smoothened the curve from the ends by changing the exponential term to :

$$CG = CG + a \times (delta) \times \left(1 - \frac{1}{1 - e^{-1}}\right) \times \left(1 - e^{\frac{-\sqrt{t}}{c}}\right) \ \ldots. \ (\mathbf{1})$$

The number of responses after which the grade would converge to a static point would directly depend upon c.
Best results were obtained for : $c = 40$, $a = 40.0$.
Since such a procedure suffers from the problem of convergence to a local optima, it could be harsh in a tutoring environment wherin a student may be classified as a weak student in first few responses but he actually may gather strength with time to achieve a better grade. The adjustment in such a scenario could be made if cut down the exponential decrease in magnitude after a predefined number of responses to allow some magnitude of fluctuations which could have scope for long term effects.
We actually achieved reduced binomial capped deviance when we cut down the exponential decrease after 25 answers (i.e using $t = 25$ in (**1**) $\forall t > 25$). Again, these constants would greatly depend upon what environment are we exposing the students. An examination environment may have no cut down point while a tutoring session could cut down quite early.

3.4.3 Learning Question Toughness

For the problem of online learning question toughness based on user's response :
This is different from the problem of updation of user grade as here we are guaranteed enough exposure to a question as compared to the case of a user's responses which could be very sparse. Slow convergence in this case shouldnt hurt in this case as we can't risk a question toughness getting stuck into a local optima as its a universal and a prime factor for our evaluation scheme as mentioned earlier.
The Steps to be followed while learning question toughness could be :

1. Start with a default toughness coefficient.
2. For each response recorded, reward or penalise the question toughness by a term which would depend on the prediction of the user answering it correctly.
3. If the outcome is a failed attempt, the magnitude of reward would also depend on current toughness as a tough question would have a smaller reward for an incorrect outcome as compared to an easy question for the same user attempting it.

To encode the relationship described in **3.**, we use a function exponential with base equal to the difference with the largest toughness i.e. 10. On the other hand, for a correct outcome, the easier questions would be less penalised than a harder question. So the base should now be the difference of current toughness with the lowest toughness i.e. 1. While adding this term to the toughness coefficient, we should use a factor w of proportion to provide an appropriate weightage to this change. The updation term described above can be summarised in the below formulae :

$$QT = \frac{QT + w \times term}{w + 1}$$

$$\text{where } \mathbf{term} \ = \begin{cases} (10 - QT)^{P(QT,CG)} & \text{for} \quad O_t = 0 \\ (QT - 1)^{1 - P(QT,CG)} & \text{for} \quad O_t = 1 \end{cases}$$

w: weight factor (Could be number of questions answered or may be any constant. Performance results on varying **w** could be seen in **Table-2**)

O_t : The actual outcome at this time

P : the probability function for outcome providing the probability of correctness for a given question toughness and a given user grade.

An Insight into Convergence of the Heuristic :

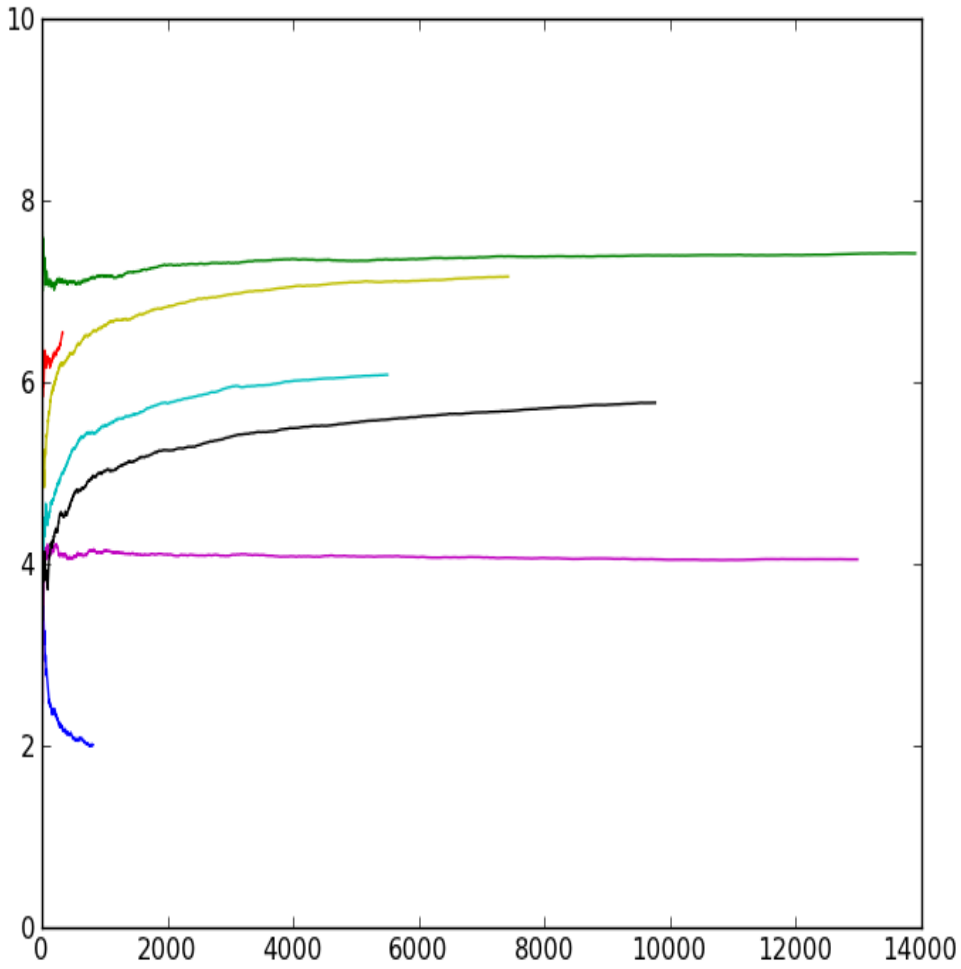

Fig. QT(**X**) v/s No. Of Instances Encountered (**Y**)

The above graph shows how question toughness varies for a particular question (Each distinct color represents the question Id for a question) as the number of instances encountered in the data set increases. We can see that, question toughness does converge fairly enough for the question which have adequate number of encounters. This greatly helps as we can always provide substantial exposure to our question set so that the Question toughness values converge over time. Also using these convergence values in recurrent user grading improves the prediction performance.
Note that though the formula provides convergence to coefficients of question toughness but it cant be applied analogously to User Grades in all scenarios. It demands for substantial exposure of the Users to the system for being successful in that condition.

Table 2: Weight factor v/s Performance (* -Best)

w	Binomial Capped Deviance
$\frac{1}{50}$	0.26838
$\frac{1}{30}$	0.26790
$\frac{1}{10}$	0.2690
$\{Averaging\} \rightarrow \frac{1}{\lvert answered \rvert}$	0.2675*

3.5 Need Of Topicwise Evaluation

Till now we have not dealt with the term topicwise evaluation. Now, we are in a shape to inroduce it. Each question was assigned a toughness coefficient and each user, a grade based on his answers. There shouldnt be one single grade assigned to a user. Every student has his own strengths and weaknesses which can be wide varying in nature for the same student. We need to evaluate the user as far as possible in different paradigms.
Tag strings is the attribute which captures the topics contained in a question. If a user answers a question correctly, it should reflect his quality in the topics contained in the question and perhaps those related to these topics. The term related topics implies "can you infer about the grade in some other topics given you have knowledge about these topics".
Given the data set, we can infer about the relation in topics if they occur together in questions quite frequently. This is because a question if contains multiple topics inside it, then it denotes a way how you could combine these topics to form a question. If multiple number of questions often contain the tagstrings together, then the tagstrings must be related to each other.
Degree of this relationship can be determined by the frequency of simultaneous occurence. Hierarchial clustering can capture this notion through its distance metric. It would club two topics together if they are close enough to each other. Here distance would mean the inverse of frequency of occurrence. The greater the frequency, the lesser the distance and hence closer in space they are.

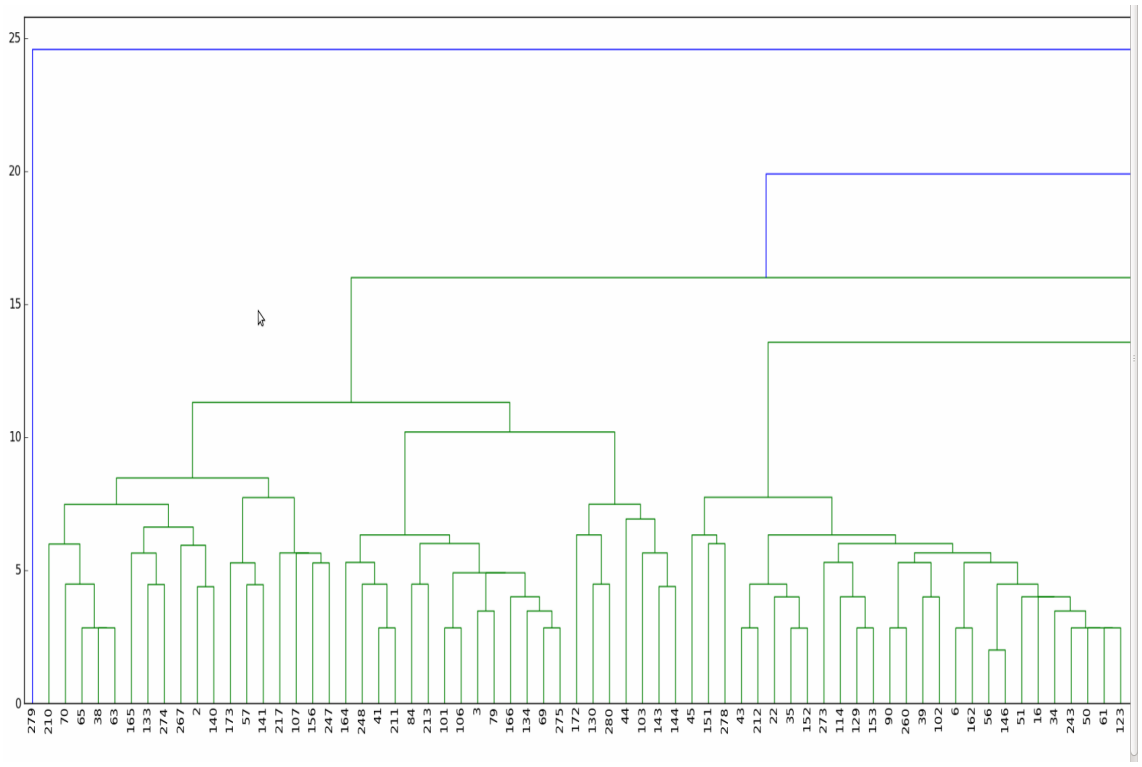

Fig. A snapshot of results provided by Hierarchial Clustering.
Each number on the **X** axis denotes a tag string (i.e. a topic). **Y** axis denotes the varying levels of threshhold associated. The higher the threshhold, the larger the cluster and the weaker the relationships.

Why hierarchial clustering :

Why we chose hierarchial clustering and not agglomerative clustering could be clear by reviewing our distance metric. Two points are related by how close they are to each other and there does not hold any transitive relation in this relationship. If two topics occur together and a third topic occurs with one of them doesnt necessarily mean the the other one has to be related to the third.

Dealing with clusters :

We start by fixing a threshhold value for clustering which measures how close two points be to be considered under the same cluster. Each user is assigned a seperate grade for each cluster that is formed. Now whenever a question is answered by a user, the cluster to which each of topic belongs gets affected. If he gives a correct answer, he needs to be rewarded for the clusters corresponding to each one of the topics. Same holds for demotion on an incorrect answer. Question toughness Learning in such a scenario would consider the cluster of each topic contained inside it and the same question instance would now be treated as multiple instances each corresponding to the clusters of the topics contained. In course of time, obviously a question comes into mind to decide how many clusters should one choose to maximize the performance in an online situation. We present here a result of our studies denoting the behaviour of Performance metric with the number of clusters created.

Number of clusters ? :

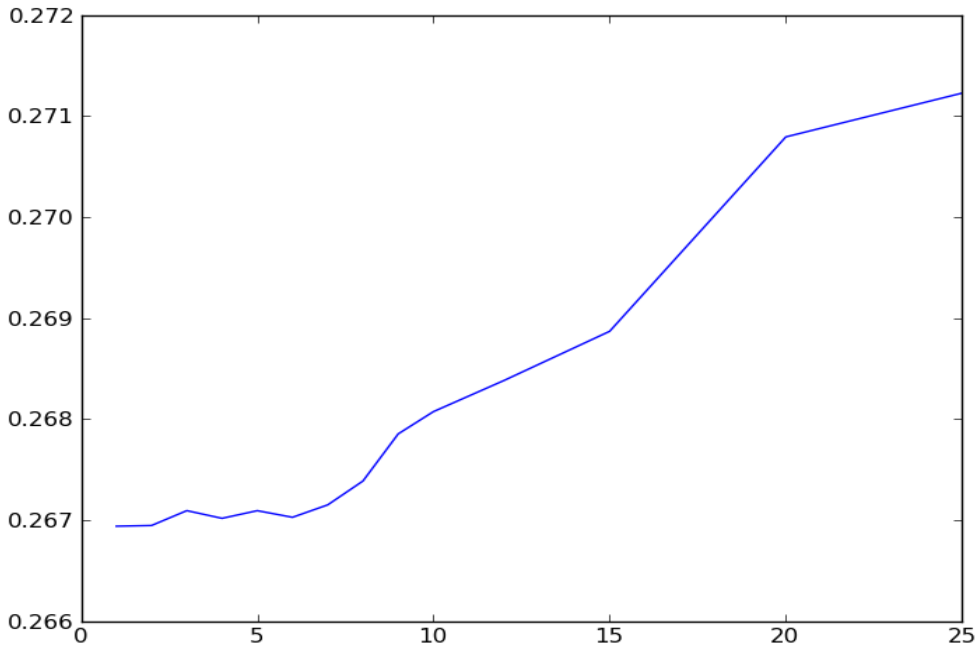

Fig. Performance Measure **Y** axis v/s Threshhold for heirarchial clustering **X** axis $(div = 5), a = c = 40$

This indicates to us that clustering would affect your prediction negatively. But we cant disregard clustering as though we may hamper our prediction quality but clustering would greatly help in deciding the grades for topics even before they were offered to the student. This may be particularly useful in a situation where in just limited number of questions, you can give a complete estimate of strengths and weakness of a student in the various topics.
So a balance is required between the performance and clustering threshhold. We require greater clustering to gain quick decisions on a user which may be quite relevant in an exam like scenario. But it factually comes at the cost of performance. At the end of discussion what remains is putting together the approach in an algorithmic form. That is what we are to do next.

4 THE ONLINE UNSUPERVISED LEARNING ALGORITHM

" An algorithm to evaluate students in an online situation"

1. Choose a clustering threshhold to obtain the number of clusters.
2. Initialize the grade (CG_0) of students for each cluster.
3. Make a prediction on the question he faces based on the probability function(P) for outcome.
   Pass it to update grade and update toughness functions
4. Observe the outcome (O_t) of the user.
5. For each cluster contained in the question
   Update the grade (CG_{t+1}) of the student
   Update the toughness of the question (QT_{t+1}) using the above grade
6. Repeat steps 3-6 till the student is available or evaluation time expires.

5 Result

On various adjustments that were made to the functions, parameters and the clustering threshholds, the best result was obtained for **binomial capped deviance = 0.2659** with worth a mention that the clustering threshhold was set to zero. The other parameters were adjusted on small scales to get this overall best result.

6 Concerns and Future Research

After having gone through the analysis of the heuristics, its incomplete to end without discussing the concerns in the above algorithm. In the initial phase when it goes about learning question toughness, its really hard to say on a response whether the user is underestimated/overestimated or is the question. What we do here is first update the grades of the user and treat this updated grade as actual grade and learn question toughness on it. It is actually designed to perform when the question toughness values have been saturated. What one intuition could say is not immediately update the question toughness on each response of its instance but wait till a particular number of instances of the question have been responded. But that gives a worse performance when attempted so. We still are not sure what the optimal and the best defining heuristic functions and the parameters could be. We would wish to test the algorithm on a real time exam so that we could further achieve some information on what path to be pursued considering the behaviour of the data. This needs to be done as our Probability functions were based on some heuristics and real world understanding that were later adjusted according to the performance metric. We also need to stabilise the algorithm. Though it could be made to converge trivially but we need to avoid a misleading convergence especially when something like a user grade holds such an importance.

7 Applications of the Heuristic Approach

As mentioned before there are two critical environments where this algorithm could be vital for the application purpose. One of them is evaluating students for the purpose of exams. The students just like our approach could be fed a series of questions and for the response generated on each instance posed, the prediction functions start working and try to gather some local region for the grades of student in each topic cluster. Conventional evaluations consider only the statistical measures for grading a student, but this algorithm could extract out farther knowledge than that about a student and hence make a better grade if led appropriately.
Another vital use of such a technique could be in a tutoring environment. With growing intent towards education amongst the people, its becoming increasingly difficult to manage a large mass of students and still take care of individual performance through the course of study. If a machine learning algorithm like this comes up with a method of discovering the weaknesses of the students, it could be a relief to everyone involved in the education system. Courses of study can be then planned using an appropriate scheme. A scheme would here refer to what series of questions to pass to the student to tutor him/ her.

8 Acknowledgements

I would like to dearly acknowledge the enormous contributions of Mr. Kapil Garg (Junior UnderGraduate , Computer Science and Engineering Department) towards the analysis and the ideas towards the advancements of this research. Also, I express my sincere regards to suggestions, advice and recommendations provided by Dr. Harish Karnick.

9 References

[1] http://www.cs.uic.edu/ liub/teach/cs583-fall-05/CS583-supervised- learning.ppt, last accessed on April 15, 2012.

[2] "Correlation-based Feature Selection for Machine Learning", Mark A. Hall, 1999, Pg. 7-19

[3] "Predicting Good Probabilities With Supervised Learning", Alexandru Niculescu-Mizil and Rich Caruana

[4] "Inductive Logic Programming – Techniques and Applications", Nada Lavrac, Saso Dzeroski

[5] "An Efficient Sequential Covering Algorithm for Explaining Subsets" Matthew Michelson and Sofus Macskassy

[6] "Bayes Nets for representing and reasoning about uncertainty", Andrew W. Moore

[7]"Calibrated Lazy Associative Classification", Adriano Veloso , Wagner Meira Jr , Mohammed Zaki, 23rd Brazilian Symposium on Databases