Predicting Alcohol Abuse Using Convergence of Geometric Harmonics

Justin Kosslyn
Advisor: Professor Zucker
May 2008

Abstract
Our goal is to predict which individuals suffer from alcohol abuse or dependency. A successful model could help policymakers allocate alcoholism prevention resources. Our technique is a similarity classifier that uses kernel extension methods to overcome noise in the training data. Our data is drawn from a large government questionnaire-based survey. We employ correlative weights on each feature and information on which questions each subject was asked to build a useful similarity kernel function. The resulting model performs above benchmarks.

Introduction
Alcohol abuse causes over 100,000 deaths in the United States every year. Yet according to a report prepared by the Schneider Institute for the Robert Wood Johnson Foundation, less than one-fourth of individuals needing treatment for alcohol abuse receive it, often due to “structural barriers, such as lack of available space or funding.” If we could accurately predict at-risk individuals, we would be able to better allocate scarce resources. It would even be useful to identify a subset of likely alcohol abusers that could be targeted for treatment, especially if there were few resource-wasting false positives.

Our solution begins with a classic approach: similarity classifiers. Given training data and new people to be classified, we examine whether the new people are similar to the alcohol abusers or the non-abusers in our training set. This similarity kernel allows fast classification of new individuals. However, such approaches suffer when the training data includes eccentric individuals, occasional errors, and overall noise. To combat this problem, we turn to kernel extension methods.

Kernel extension methods allow us to effectively smooth the training data by using eigenvectors of the similarity matrix. We use the eigenvectors in two ways. First, we project the training classifications onto the eigenvectors to find the weighted sum of eigenvectors most approximating the training classifications, and we save those weights for later. Second, we extend the eigenvectors to the testing data by assuming that $Kf = \hat{Af}$ still holds true when we add new rows of data corresponding to new people, where $K$ is the similarity kernel and $f$ is the classifications vector. And finally, we use the weights from the training classifications approximation to calculate a weighted sum of the kernel extensions. The resulting values are our predicted classifications for the new people.

NIAAA Data

We rely on a large dataset from the National Institute of Alcohol Abuse and Alcoholism. Called NESARC WAVE 1, the dataset contains answers to 3,008 questions for 43,093 Americans. Typical questions include:

- EVER GAMBLED 5+ TIMES IN ANY ONE YEAR
- HAVE TROUBLE THROWING OUT WORN-OUT/WORTHLESS THINGS EVEN IF HAVE NO SENTIMENTAL VALUE
- HAVE OTHERS TOLD YOU THAT YOU ARE STUBBORN OR RIGID
- LOST APPETITE NEARLY EVERY DAY FOR 2+ WEEKS
- HAD HIGH BLOOD PRESSURE OR HYPERTENSION IN LAST 12 MONTHS
- BUSINESS OR INDUSTRY: CURRENT OR MOST RECENT JOB
- TOTAL FAMILY INCOME IN LAST 12 MONTHS
- NATURAL GRANDFATHER ON YOUR FATHER'S SIDE EVER HAD PROBLEMS WITH DRUGS
- HOW OFTEN DRANK ANY ALCOHOL IN LAST 12 MONTHS

The questionnaire has a branching and complex structure. To further complicate matters, the questionnaire cannot even be represented as a graph because some branches refer back to earlier questions. See Figure 1 for an example.

---

Figure 1: Page four of the questionnaire. Note that multiple boxes may be checked in 7c and that Check Item 1.1D refers back to 7a.

---

2 Available online at http://niaaa.census.gov/. The data was collected in 2001 – 2002.
An entire section of the questionnaire covers alcohol consumption patterns, from which a DSM-IV diagnosis of alcohol abuse or dependence in the last year is determined for each subject. This variable is our classification target.

Method

We go through several steps to predict classifications: preprocessing, building the similarity matrix, generating the smoothed training classifications, and finally calculating predictions.

Preprocessing

Before we can begin classification, we must perform preprocessing on the data to get it into usable form. Each possible answer to each question becomes a distinct feature. In the case of Figure 1, for example, question 7b becomes three features (7b_1, 7b_2, and 7b_3), each of which will be valued 0 or 1 in our new data matrix. Some questions, such as “Age?” allow answers in a range. In those cases, the answer is scaled to between 0 and 1, where 0 is the bottom of the range and 1 is the top. The alcohol abuse or dependency feature goes into a column vector, f, in which 1 means abuse and/or dependency on alcohol and 0 means otherwise.

Due to the branching nature of the questionnaire, many questions are never asked to many subjects. For example, “How many cigarettes do you smoke each week?” is irrelevant if the subject has already been identified as a non-smoker. Unasked questions are labeled as blanks in the dataset, but our one-feature-per-possible-answer conversion loses that information. So we create a separate matrix, called answered, containing information on which questions each subject actually answered. So if the first question has two possible answers, “yes” and “no,” and subject 1 answered “yes”, then data(1,1)=1, data(1,2)=0, and answered(1,1)=1.

During preprocessing, we also remove all questions about alcohol, drinking, and alcohol-related incidents. Using them would be tantamount to cheating. We also remove any data fields that signify imputation flags and administrative checkpoints, since they are irrelevant for our purposes. Next, due to memory constraints, we randomly select 10,000 subjects from the dataset and throw away the remainder.

Finally, we divide the data into training and testing sets of equal size. Within the training data, we find the largest possible subset that contains equally large sets of alcohol abusers and non-abusers. This is our usable training data; the unbalanced remainder of the training data is ignored. We are now ready to build our classification model.

Building the Similarity Matrix

Our first step is to build the similarity matrix for the training data, K. We use a heuristic kernel function:

\[
K = e^{\frac{\text{data}_1 \times \text{data}_2}{\text{answered}_1 \times \text{answered}_2}}
\]  

[Formula 1]

3 http://niaaa.census.gov/pdfs/datanotes.pdf
4 Using 1-indexing. We follow matlab conventions throughout this project.
with some normalization\(^5\), where \textbf{data} is the matrix whose rows are subjects and whose columns are features, and \textbf{answered} is the matrix whose rows are subjects and whose columns are whether questions were answered. \(K(i,j)\) is the similarity between subjects \(i\) and \(j\). Note that this kernel computes the number of specific answers each pair of subjects shared divided by the number of questions each pair of subjects shared. So the similarity of two subjects is based on the percent of same answers to questions both subjects were asked. This approach prevents the branching questionnaire from yielding bizarre results, since similarity is based only on the overlapping segments of two subjects’ branches.

However, not all questions are created equal. There is an entire questionnaire section about the subjects’ family history of gambling, but it is clearly less relevant to inter-subject similarity than the section on income, ethnicity, and industry. When every question is weighed equally, our similarity metric may yield useless results. And there are far too many questions to decide their relevancies on a case-by-case basis.

The resolution of this problem is simple. Using the training data, we calculate the correlation between each feature and the training classifications, \(f\). We then weigh each column of \textbf{data} by that feature’s correlation with \(f\). Thus, irrelevant features are effectively marginalized without the need for manual identification.

#### Generating the Smoothed Training Classifications

We now have a similarity matrix \(K\) and are ready to work with it. This process has several steps. First, we singular value decompose \(K\) to get its orthogonal eigenvectors and eigenvalues. We take the thirty eigenvectors with the largest corresponding eigenvalues and put them in a matrix \(U\).\(^6\) This is a basis for the more relevant part of the eigenspace.

We now compute the projection of \(f\) onto \(U\), which we call \(\alpha\).\(^7\) This is just \(<f, U>\); ie, \(\alpha\) is the inner product of \(f\) and \(U\). Note that by definition \(\alpha\) contains weights for each eigenvector such that

\[
f' = \sum_{i=1}^{30} \alpha_i U_i = U\alpha \quad \text{[Formula 2]}
\]

where \(f' = f\) (if the projection onto \(U\) did not lose too much of \(f\)’s magnitude. See the Rayleigh Quotient analysis in the Results section for more on this). Furthermore, \(Kf' = f'\) due to the eigen property, since \(f'\) is simply the weighted sum of orthogonal eigenvectors of \(K\).

\(^5\) In Matlab notation, \(K = \exp(-(1-K0)^2./((1-sigma)*(1-sigma'))), \) where \(K0 = (\text{data}^*\text{data'})/(\text{answered}^*\text{answered'})\) with the value of each entry capped at 1, and \(\text{sigma} = \text{Ksort(10, : )}, \) where \(\text{Ksort} = \text{sort(K0, ‘descend’). And then use the Graph Laplacian normalization, which preserves K’s eigenvectors: p = sum(K), p = p(:), K = K./(sqrt(p*p'))}\).

\(^6\) Thirty is a completely arbitrary number, but this fact seems to have little bearing on our results – switching to only fifteen eigenvectors, for example, changes the AUROC by less than 0.02.

\(^7\) We also employ a useful heuristic: first replace \(f\) with \((f - \text{mean}(f))\) so that the vector sums to zero, and later add \(\text{mean}(f)\) back onto our predictions as a final pre-evaluation step.
\( f' \) is our smoothed training classifications, the closest we can get to \( f \) without our eigenvectors. However, \( \alpha \), the eigenvector weights, will actually prove more relevant to the final step in our analysis.

Calculating Predictions

We now temporarily turn our attention back to \( U \). Because \( U \) is a set of eigenvectors, \( KU_k = \lambda_k U_k \), where \( U_k \) is one of the eigenvectors and \( \lambda_k \) is its eigenvalue. Thus, by definition,

\[
\sum_j K(i,j)U_k(j) = \lambda_k U_k(i)
\]

where \( U_k(i) \) is the \( i \)th entry in eigenvector \( U_k \). This fact is very useful to us. When we are given a new row of \( K \) corresponding to the similarities between a to-be-classified individual and our training subjects, we can use this formula to extend \( U_k \) to include an entry for our new individual. The equation is simply

\[
\frac{1}{\lambda_k} \sum_j K(\text{new person}, j)U_k(j) = U_k(\text{new person}) \quad \text{[Formula 3]}
\]

This technique assumes that the new person’s similarity data is close to the eigenspace derived from the training similarity data, and so the eigen equation still holds. As our results will show, this is a reasonable assumption to make.

Using Formula 2, we can extend all the eigenvectors in \( U \) to cover our new subjects. And once we have that we are almost done: predictions = \( \text{test}U \times \alpha \). This is simply Formula 2 (calculating \( f' \), the smoothed predictions) using the extended eigenvectors. And it yields us predicted classifications, our goal throughout this process.

One minor step was glossed over, though – calculating the new parts of \( K \). This is simply achieved. We start with a new matrix \( \text{testdata} \) of features for our to-be-classified subjects, as well as a new matrix \( \text{testanswered} \) of which questions each test subject actually answered. We use them to build a new similarity matrix:

\[
\text{test}K = e^{\left(\frac{\text{testdata} \ast \text{data}}{\text{testanswered} \ast \text{answered}}\right)}
\]

So \( \text{test}K(i,j) \) is the similarity between subject \( i \) and subject \( j \), where \( i \) is a testing subject and \( j \) is a training subject. Thus \( \text{test}U \), the extension of \( U \) (which is the only part of \( U \) we actually care about at this point), is simply

\[
\text{test}U = \text{test}K \times U \times \frac{1}{\text{diag}(\lambda)}
\]

---

8 Similar normalization to Formula 1’s normalization. \( \text{test}K = \exp(-(1-\text{test}K0)^2/((1-\text{sigma})*(1-\text{testsigma}))) \), where \( \text{test}K0 = (\text{testdata} \ast \text{data})/(\text{testanswered} \ast \text{answered}) \) with the value of each entry capped at 1, and \( \text{testsigma} = \text{testKsrt}(; , 10) \), where \( \text{testKsrt} = \text{sort}(\text{test}K0, \text{‘descend’}) \). And then once again use the Graph Laplacian normalization: \( \text{testp} = \text{sum}(\text{test}K, 2) \), \( \text{testp} = \text{testp}(;) \), \( \text{test}K = \text{test}K/\text{testp} \ast \text{ones(size(\text{test}K, 2), 1)} \).
where $\text{diag}(\lambda)$ is a diagonal matrix of the eigenvalues corresponding to each eigenvector in $\mathbf{U}$. This follows directly from Formula 3.

So we now see how to get from raw data to similarity matrices to eigenvectors to projection weights and kernel extensions to, finally, predictions. This is our method.

**Results**

In short, our model performs well. The AUROC is .777, yielding 93% accuracy if we want to correctly classify 99% of nonabusers and 16% of alcohol abusers and dependents. This means we can find a subset of the at-risk population with very few false positives.

We benchmark this performance against a KNN classifier and an SVM classifier. The KNN model’s AUROC is .695, but with no useful spot near a corner. We can get 76% accuracy if we want to correctly classify 78% of nonabusers and 62% of alcohol abusers and dependents. Due to memory constraints, this model is run on only 5,000 subjects – half as many as our experimental model. It also takes considerably longer to evaluate. Note that $N=1$. 

![Figure 2: Our model’s ROC curve](image)
Figure 3: The KNN model’s ROC curve, which shows reasonable performance that our model exceeds.

The SVM model performs significantly worse, with an AUROC of .504 – only marginally better than a coin flip. Once again, the model's memory demands limit us to 5,000 subjects. An RBF kernel is used.

Figure 4: The SVM model’s ROC curve, which shows terrible performance.
It is also interesting to examine what happens under our model if we do not weigh each question based on its correlation with the alcohol abuse vector. Under this condition, our model performs as poorly as the SVM model, with AUROC = .504. A good kernel function representing actual similarities seems essential to our approach.

![Image of ROC curve](image)

Figure 5: Our ROC curve without weighting data columns based on their correlation with the classifications vector.

With that in mind, here is what our similarity matrix actually looks like:
Figure 6: Our model's similarity matrix under the exponential kernel function. Note that most subjects are particularly similar to a relatively small number of other subjects.

There are a lot of similar pairs of subjects in this kernel, and one wonders whether they are all equally relevant to our calculations. To answer this question, we observe that the fact that \( \text{predictions} = \text{testU} \times \alpha \) combined with Formula Three yields

\[
\text{predictions} = \text{testK} \times \text{U} \times \frac{1}{\text{diag(\lambda)}} \times \alpha
\]

which means that our predictions are just linear weights on the similarity data in \( \text{testK} \). That is, after we have extended our similarity kernel to the test subjects, we are effectively computing a weighted sum across each row to predict their classifications.

We calculate \( \text{U} \times \frac{1}{\text{diag(\lambda)}} \times \alpha \) in the absence of any specific testing data to look at the distribution of weights, and find that not all training subjects are equally relevant:

![Weights applied to testK when computing predictions](image)

Figure 7: The weights applied to the columns of \( \text{testK} \) to compute predicted classifications, sorted in ascending order. Recall that the columns of \( \text{testK} \) are the similarities with each training subject.

The weights for most training subjects are close to zero, meaning that the similarity between the test subjects and those people has little bearing on our alcoholism predictions. However, there are a relatively small number of training subjects for whom the weights are significant, so the similarity between the test subjects and that subset of the training subjects will disproportionately determine the classifications. This result is hardly surprising; our model set out to cut through the noise in our training data, and has found a small number of archetypes whose similarity to the testing data is particularly pertinent.

Other Features and Rayleigh Quotients
How does our model fare for features beyond alcohol abuse? For each feature \( l_i \) across all subjects, we compute
\[ R_l = \frac{\|U^T f_l\|_2}{\|f_l\|_2} \]

Where \( U \) is our thirty eigenvectors from the training similarity matrix, \( f_l \) is a vector of all training subjects’ values of feature \( l \), and \( R_l \) is the Rayleigh Quotient of feature \( l \). In effect, we are calculating what percent of that feature survives projection into our eigenspace. Though this quotient will presumably vary from feature to feature, higher is generally better because it means our eigenvectors are capturing most of what is going on in the dataset.

Figure 8: The Rayleigh Quotients of every feature that is nonzero for at least 2.5% of subjects, sorted in descending order. We see that our quotients are generally quite high, so our eigenvectors are covering most of the feature landscape.

We can confirm our model’s multi-feature applicability by classifying smoking behavior rather than alcoholism. Our AUROC is .710, and the model yields 75% prediction accuracy if we want to correctly classify 91% of nonsmokers and 30% of smokers – a potentially useful tool for allocating resources for anti-smoking publicity campaigns.
Conclusion

Our goal was to predict alcohol abusers and dependents. To achieve this goal, we implemented a complex similarity classifier using kernel extension techniques. Our primary obstacles were the complex structure of the questionnaire and noise in our training data. The first problem – hundreds of unasked or irrelevant questions for each subject – was tackled in two ways. First, we used each feature’s correlation with alcoholism as weights on the data. This cut through the irrelevant features in an automated and unbiased way. Second, for our similarity matrix, we implemented a kernel function that, when computing pair-wise similarity, only counts questions that both subjects were asked. This handles the difficulty posed by the questionnaire’s branching nature.

Our second obstacle – noise in the training data – was tackled with kernel extension techniques. We found eigenvectors of the similarity matrix and extended them to the testing data. We projected the training classifications into the eigenspace to find the appropriate weights for each eigenvector, and used those weights on the eigenvector extensions to calculate our predicted classifications for the testing data.

We expected this approach to yield accurate predictions, and we were vindicated. Our model performs above benchmarks and could effectively be used to assist policymakers allocate resources to combat alcoholism. Our model can also be used to predict other features such as smoking behavior, with similar accuracy.

Figure 9: Our Model’s ROC when predicting smoking behavior, which is about as predictable as alcohol abuse and dependency. Note that all questions related to nicotine were removed from the dataset for this task, while those about alcohol were put back in.
Appendix: The Structure of the Dataset

One frustration of our approach is that it is quite detached from the actual data. We are curious what the members of each class actually look like – what characteristics tend to separate those who abuse alcohol from those who do not?

To answer this question, we implement a biclustering algorithm. We create a block of subjects and features that initially contains the entire data matrix and classes vector, and normalize it so that the matrix sums to zero. We will iteratively add or remove a subject or a feature until a coherent block is reached. This can be achieved through a greedy algorithm in which the value of a given block is simply the average value of the features in that block for the subjects in that block. If we require that the classes vector remain in the block, then our algorithm will essentially select for the subset of alcohol abusers with the most consistent answers to other questions (yielding consistent features that the algorithm will keep in the block).

At each stage of iteration, we need only calculate the block’s value under four possible options:

1. If we add a subject, we would add the highest-valued subject not in our group. So calculate the block’s hypothetical value with that subject.
2. If we remove a subject, we would remove the lowest-valued subject in our group. So calculate the block’s hypothetical value without that subject.
3. If we add a feature, we would add the highest-valued feature not in our group. So calculate the block’s hypothetical value with that feature.
4. If we remove a feature, we would remove the lowest-valued feature in our group. So calculate the block’s hypothetical value without that feature.

We then simply pick the option with the highest value, or stop if our current value exceeds all options.

This simple algorithm yields surprisingly rich results. When run with 10,000 subjects and our standard feature set, the resulting block contains 509 subjects, all of whom are alcohol abusers or dependents. This is the majority of the 737 alcoholics among our 10,000 subjects. Our block also contains 914 features, which represent uninteresting facts like “present situation is not a paid vacation,” “never used inhalants,” and “no pathological gambling in last twelve months.”

Our interesting results come about when we look at which features have the largest and smallest ratios between the average value among subjects in our block and the average value among subjects outside our block. We find, for example, that subjects in our block of alcoholics are male 162% as often as subjects outside our block. Subjects in our block are also 192% as likely to be smokers, 215% as likely to have sought help for drug or medicine use, and 205% as likely to be former drug users who have been clean for at least a year.

Furthermore, subjects in our alcohol-abusing block are on average only 79% as old as subjects outside our block – 37 years versus 47 years, on average. Thus it is hardly surprising
that subjects in our block are 160% as likely to have had no children and only 30% as likely to be currently retired.\(^9\)

Finally, subjects in our block report making 151% as much money in personal income, on average, as subjects outside our block. Average income for our block of alcoholics is $42,600 per year, while average income outside our block is only $28,200 per year. Intriguingly, alcoholics outside our block make, on average, $25,500 per year. That is, personal income is a major division between the in-block alcoholics and the out-of-block alcoholics.

What have we learned from this exercise? That our dataset contains a significant block of alcohol abusing subjects who fit a mold: young, well off, employed, and with tendencies towards nicotine and drugs. There are other blocks, like the poorer alcoholics, and other divisions are likely discernable. It is such structural characteristics that our similarity classifier exploits to yield our predictions.

References


\(^9\) One may be justly concerned about small sample sizes skewing these ratios. But the features are generally frequent enough to have some significance. In the case of having children, for example, 212 subjects in our block have no children, compared with 2461 subjects outside our block.