Midterm Exam

36-350, Data Mining

SOLUTIONS

1. *Wine recommendation*

   (a) Answer: This is the same as creating a data frame from bags-of-words. We have one feature for each distinct wine; it is 1 if the member owns the wine and 0 otherwise. The number of features is simply the total number of distinct wines.

   For creating the data frame, the strategy used in the homework code will work: use `table` to create a binary vector for each member, then take the union of the different wines owned by the different members, and, for each member, set the features for the wines they do not own to zero. (Glossing over issues like how `table` will handle the spaces in the names of wines is OK.)

   Alternately, start by taking the union of all the wines all the drinkers own, and then, for each drinker, go down the list of wines and put a 1 in the data frame if they own the wine and 0 if they do not.

   (b) (10 points) Answer: The most routine solution is to use the Euclidean distance between two drinkers' feature-vectors, normalizing by either the total number of wines owned or by the Euclidean length of the feature vectors, and weighting features (wines) by their IDFs.

   (c) (5 points) Answer: Writing n_i for the number of drinkers who own wine i, and n for the total number of drinkers, the IDF $w_i = \log n/n_i$. Thus the IDF weight of "L'Inferno 00" is $\log 5/3 = \log 5 - \log 3 = 0.51$. IDF should be useful because a shared enjoyment of unusual wines (with high IDF) should be a better sign of similar tastes than a shared enjoyment of common wines (which most people like).

   (d) (10 points) Answer: Take the k most similar drinkers to the customer, and take the union of all the wines they own. Remove the wines already owned by the customer. If there are more than m wines left, there are several reasonable selection strategies: pick the wines with the lowest IDF weights (most popular), the highest IDF weights (most distinctive), etc.

   (e) (Extra credit; 10 points) Answer: Before computing similarities, use principal components on the feature vectors, and take the q largest components. (Chosing the right q is hard here, and is really best

done by evaluating success.) Calculate similarities by the Euclidean distances between PCA vectors. Then, having identified the most similar drinkers to the customer, pick out the new wines as before.

2. *k-Means Clustering*

   (a) (15 points) ANSWER: (i) Start a set of n vectors $x_1, x_2, \ldots x_n$. (ii) Assign each vector at random to one of the k clusters. (iii) For each cluster, compute the mean of the vectors belong to that cluster. (iv) For each vector, assign it to the cluster whose mean is closest to it. (Do not recompute the means while these assignments are being made.) (v) If any vectors have changed their cluster assignments, go back to step (iii); if not, stop.

   (b) (5 points) ANSWER: No. It can never give more clusters, since at every stage every point is assigned to one of k clusters. To give fewer than k clusters, we would need there to be a cluster which got *no* points at one of the re-assignment stages. This means that its center would be farther from *every* point than one of the other cluster centers. But since the center lies in between the points currently assigned to the cluster, that's not possible.

   (c) (5 points) ANSWER: For each cluster, it is the sum of the squared distances of points in that cluster to their center, summed over clusters. Writing C_i for the points in cluster i, and m_i for the mean of cluster i,

   $$SS = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - m_i\|^2$$

   (d) (8 points) ANSWER: A reasonable guess here is 4; the sum of squares goes up dramatically after that, but adding more than 4 clusters does little to lower the sum of squares. Visually, $k = 4$ gives us four compact, well-separated clusters with fairly clear divisions between them, which is not true of either more or fewer clusters.

   — In fact, the data were generated as a mixture from four different Gaussians, centered at $(-1,-1), (-1,1), (1,-1), (1,1)$, all with covariance matrices $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

3. *What's That Got to Do with the Price of Houses in California?*

   (a) (6 points) ANSWER: `MedianIncome` is the obvious predictor, since it's most strongly correlated with the response variable `MedianHouseValue`. Following that, `MedianHouseAge`, `AveRooms` and `Latitude` are all possiblities, though `AveRooms` is reasonably correlated with `MedianIncome` so it may be redundant. `AveOccup` is only very weakly correlated with the response, and in fact very weakly correlated with everything else as well, so it seems like an obvious candidate for dropping.

(b) (8 points) ANSWER: Linear regression can be defended on the grounds that the bulk of the data seem to follow a roughly linear trend (admittedly with a big scatter), up to about the point where the diagonal line crosses 500,000, at which point something rather funny seems to happen to the data. The difference in fit between it and the kernel smoother only really takes off at this point, and the latter includes a very weird and implausible jog downward around `MedianIncome` ≈ 12.

The kernel regression can be defended on the grounds that (i) the fit really is better, (ii) the spread is so big that a specifically *linear* trend is hard to justify, and (iii) the data are the data, and we should be predicting them and not something else.

Both fits are not very good, and there is not only a lot of spread around the trend lines, but the magnitude of the spread is variable. This suggests that extra variables are worth trying — they should improve the precision.

(c) ANSWER: The first command does a principal components analysis of the predictor variables (i.e., everything other than `MedianHouseValue`). The function `r2.with.q` first regresses the response on the q largest principal components, and then extracts the r^2 of the regression. Finally, the last line calculates r^2 as a function of q for q from 1 to 8, and then plots it. Over-all, this is seeing how much of the variance in price can be predicted by regressing on the first q principal components of the independent variables.

This is not the same as a scree-plot, which shows how much of the variance *in the features themselves* is predicted by the first q principal components.

Since r^2 can only go up as the number of predictors increases, the plot produced by these commands has to be the first one. (The second one is, in fact, the scree plot for the PCA.)

(d) (6 points) ANSWER: It calculates summary statistics for each feature (column) of the data frame `CaliforniaHousing`. Notice that `AveRooms`, `AveBedrms`, `AveOccup` and `Houses` all have the funny trait of having a fairly small range from the 1st quartile to the 3rd quartile, but a maximum which is immensely larger. Also, notice that the average columns have minimums which are less than 1. This might make sense for the average number of occupants per house, if enough people own multiple houses, but there can't be less than one room or bed-room per house. So something's fishy.

(e) (6 points) ANSWER: There is a huge spike in the histogram for median house value around \$500,000. (Actually, looking at the summary table, \$500,001.) This is also evident from the scatter-plot with the regression curves. This is very suspicious, and suggests **top-coding**, where values above some cut-off are recorded as the maximum allowed value. Similar spikes for median income and median house age

suggest the same problem. The histograms for the average number of rooms, average number of bedrooms, average number of occupants and total number of houses are all widly lopsided, with most of the distirbution in a reasonably compact range, but a few outliers which are much, much larger.

Implications: the regression results are going to be screwy. A perfectly linear relationship could be seriously messed up by top-coding of the dependent variable (as well as the independent one). Also, this will tend to deflate correlations, as will the presence of outliers. (This may be why `AveOccup` has no correlation worth speaking of with anything.) We need to either clean the data, or to use methods which are more robust to this kind of ugliness.

— Incidentally, issues like top-coding, wildly implausible values, etc., are typical of large real-world data sets.