Data Completion and Interpolation Using Graphical Model

Mo Shi

Jin Dapeng

Arthkumar Desai

1. Data Pre-processing

1.1 Manually check all the typos and make the data easy to process.

In this project we chose ML3 as our data set. Originally the dataset is in a mess. There are a bunch of typos when they write something down, especially, when we dealed with some features, such as row ‘attention check’ and row ‘major’. We have learned that there are many ways to write ‘I have read the instructions’ and their majors, such as ‘I have read the instuctions’, ‘I have not read the instuctions’, ‘phsycolgy’ and so on. So, we need to modify them manually.

1.2 Change all values in dataset to binary or categorical data.

We have right answers for some features, such as 'anagrams1', 'anagrams2', ‘attention check’, and from 'backcount1' to 'backcount10'. Therefore, we turn the values of this features to binary data. The reason why we chose categorical data is that we need to calculate P(X = x1), if x1 is a real value, then P(X = x1) will be 0, so that we cannot get any useful information. Therefore, instead of predicting precise value for each missing data, we predicted the category which the value belongs to. To decide the appropriate categories, we used Pyspark to calculate all statistical information, like mode or frequency to decide how many categories we need and what is the range of that categories.

1.3 Merge majors to bigger category to reduce the number of the features.

In the questionnaires there were too many different majors which is hard to predict. To solve this, we manually categorized them into 8 big areas instead of using the original 100 majors, 'Business', 'Science', 'Medical Science', 'Engineering', 'Humanities and Social Science', 'Sports', 'Behavioral Science' and 'Other', to reduce the number of the features.

1.4 Expand features which do not have numerical relations

If we have 3 categories (features) [‘CS’, ‘EE’, ‘Other’], if my major is CS, then these 3 rows will be [1, -1, -1]. And if Mo Shi’s major is EE, these 3 rows will be [-1, 1, -1]. In this way of encoding and feature expanding, we avoided some possible underlying mathematical relationship between features to obtain better performance.

1.5 To make sure that all values of features are either ‘NA’ or a number.

To sum over, we changed all answers to either an integer or ‘NA’ that represents a blank that the matrix will be recognized by our program, using the strategies above.

1.6 Return a list of all possible values for every feature asked by algorithm designer.

This is another information needed by the inference part. We noted down all possible values for each feature and stored them in another .csv file so that later we can use these values to do the prediction or infererce.

2. Building / Training the Model

2.1 Model Choice / Description

For this project, we chose graphical model to represent the correlation between different features. More specifically, we used the correlation matrix in which the value (i, j) represents "how much related" the feature i and feature j are, and positive/negative sign represents whether these two features are in a positive or negative correlation. One major advantage of graphical model is that it's very intuitive – by seeing the values directly we can know the relationship between features to some extent. Another advantage is graphical model does not require huge amount of samples like neural network. For our data set, we have nearly 2,500 samples with over 130 features, which is not very large. The use of graphical model helps us avoid the extra need of expanding the data set, though the trade-off is it may be more complexed to be trained.

To build the graphical model based on the data set provided, we chose Graphical Lasso as our training algorithm. The core idea of Graphical Lasso is forcing those small values in the correlation matrix to be 0, i.e., it provides a function of eliminating edges between features which are not so important, just like the usage of Lasso regression. Another famous algorithm – Chow Liu Algorithm also has the ability of edge elimination, but it will only return a tree, which means it can only look at two features each time and there will never be "a group of features" that are strongly related to each other. Comparing to Chow Liu, Graphical Lasso has two main advantages. First, it's a lot "milder" at eliminating edges that it will probably generate a better structure. Second, with the updated correlation matrix Graphical Lasso learns both the structure and the parameters of the graph, while Chow Liu just roughly used the probability estimated from the data set.

With the same idea of Lasso regression, Graphical Lasso also uses a penalty coefficient λ as the strength of eliminating parameters. To have a good performance we need to choose a proper λ value for training. We'll specify this and other details of implementation in the coming section.

2.2 Building / Training the Model

The theory and algorithm of Graphical Lasso is first specified in the paper *Sparse inverse covariance estimation with the graphical lasso* by Freidman et el. (2007) The core idea is: based on the data set first generate a matrix W (k*k, where k is the number of features), which is the estimated covariance (or correlation, in this case) matrix, and partition W in the following way:

$$W = \begin{pmatrix} W_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix}$$

where W11 is formed by the first k columns and first k rows of W, W12/W21 are the (k-1) vectors, and W22 is the scalar. The updating strategy for Graphical Lasso is that it will choose each column in W as the last column (explicitly), and compute a β value that:

$$\min_\beta \left\{ \frac{1}{2} ||W_{11}^{1/2}\beta - b||^2 + \rho||\beta||_1 \right\}$$

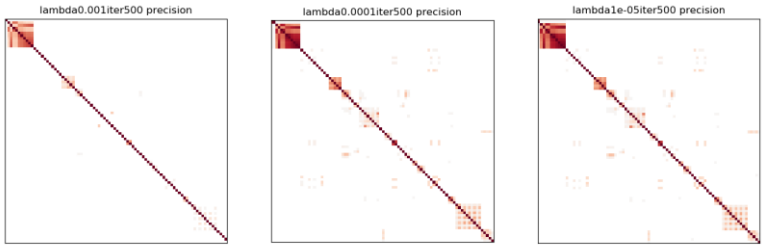
here ρ corresponds to the λ, which is the penalty coefficient of Lasso. And with this β value each part in W will be updated in some formula.

The core of this method is obviously the computation of β. To obtain the β value that solves the

formula above, there are several different strategies, including the general solution of Lasso regression based on coordinate descent. In our program we used another strategy called least angle regression (LARS), which also provides the function of feature selection.

As for the termination of training, we used two strategies together to guarantee that the process will terminate at a proper time. One is to test the convergence of the correlation matrix, and specifically we computed the "duality gap" between the previous W and the updated W – if the duality gap between them is less than a specific threshold, then we conclude that the convergence is reached. We also set a maximum iteration time in case that the convergence cannot be reached, though in practice we hardly ever run into situations like that.

There are several different things that will influence the performance. The most important one is of course the λ value. In practice we tested for several different λ values, from 0.01 to 0.000001, and generated different graphs for comparison. If λ is too large, then the final correlation matrix will be too sparse that only the elements on the diagonal are not zero. If λ is too small, the correlation matrix will soon reach the convergence since each time the change is small and the duality gap is soon beneath the threshold. We plotted our matrices under each λ value, and three of the results are as following:



Though not very clear we can still see some difference of sparsity between the matrices. At last we chose λ to be 0.0001 to be our final output, which reached the convergence after 78 iterations.

Another very important factor we found that influence the performance is the iteration time – not for the whole training process, but for the LARS. Originally we set the maximum number of iterations to be 300, and the program raised an error that the value in W is so small that caused overflow (e.g., e^{-128}), and the convergence could not be reached too. When we tested for smaller data set (in total 10-20 features) this would not happen. We found out the reason why this would happen is a flaw for the graphical lasso algorithm: when running the LARS or Lasso regression only W12 is supposed to be updated, but it turned out that the whole W matrix is changed (this flaw was pointed out in the paper of *The graphical lasso: New insights and alternatives* by Rahul Mazumder and Trevor Hastie, 2012). So, when running on our data set with over 130 features, each feature will be updated 300 times, which makes the whole matrix changed 300 times. We tried to print out the intermediate results and found all values were decreased in a very fast speed, and soon the overflow error raised. To solve this we changed our iterating time for LARS to 20, and the model could be generated successfully.

3 Analysis

After pre-processing we got a cleaned version of data set with nearly 150 features, and we chose 133 of them to build our model – some questionnaires seemed to appear alternately for different candidates. In other words, they answered either some questions or the others, which caused lots of 'NA' in the data set. To build our model we need a full matrix without any blank, and our method is to use the mode value to fill in those blanks first and build our model. If those alternative features are all included, then nearly half of all samples will have the answer 'NA', which makes our strategy of using the mode value totally rude, so each time we only chose on set of these kind of features and only use the half samples in which these questions are answered. From the graph we generated we had some interesting findings. We'll list these interesting feature pairs and try to infer why they showed correlation in between.

- 'Multiracial' – 'African American', 'Asian-American', 'Latino or Hispanic':

This is interesting because multiracial and other features listed above were originally one feature, which means if the candidate chose 'Multiracial' he wouldn't be able to choose 'African American' or 'Asian-American', so they should be negative correlated. But our graph shows they're positive correlated, which fits the common sense. Our inference for this is that the algorithm found some hidden feature(s) that didn't show up in the original data, and recognized African American and Asian-American candidates are also 'Multiracial'.

- 'Science' – 'NovaSoutheasternUniversity'

Both major and candidate's university are features we expanded, but surprisingly our graph shows a positive correlation between these two features, which probably suggests science is a considerable area in Nova Southeastern University, or the candidates from NSU are majorly selected from science area. We also found some similar results, such as 'Indian' and 'CarletonUniversity'.

- 'attention' / 'attentioncorrect' – 'backcount 1-10'

'attention' is a question that in a long description the candidates are asked to type in 'I read the instruction' in a blank below, but above the blank there are several choices that ask the candidate what they prefer to do in their free time. If the candidate is paying enough attention he'll notice what he/she is actually supposed to answer. 'backcount' is a series of questions that ask the candidate to count 10 numbers starting from 360, each time decreased by 3. We found out that for those candidates who didn't pass the attention check, they were also more likely to fail at back count questions, which totally makes sense since if they're not paying enough attention they won't count those numbers correctly.

- 'UniversityOfVirginia' – 'backcout 4 to 10'

This is an unexpectable but also kind of funny finding: the candidates from University of Virginia seemed to be not so good at back count questions, and generally they seemed start to fail from the forth number. Probably if we use more samples this edge will be eliminated (hopefully).

- ‘Engineering’ – ‘intrinsic_01 to 03’

‘intrinsic_01’ to ‘intrinsic_03’ are questions that basically ask candidates if they like complex or challenging questions, and our result shows that candidates studying in engineering area seemed to be more interested in those kinds of questions.

- ‘div3filler’ – ‘backcount 4 to 10’

‘div3filler’ is a question that gives a set of numbers and ask the candidate to find all numbers dividable by 3. An interesting point is that, there are also two ‘0’s in this set. When doing pre-processing we changed this feature from a string to scores that describes the number of correct answers the candidate could get, and the candidate who could find ‘0’s are considered to be the best at this question. Turns out that those candidates with high scores at this question were also more likely to get all back count questions correct, which makes sense since these questions are all requiring the sensitivity to numbers.

4 Estimation

4.1 Junction Tree

The inference in the tree is very easy to do, while in case of graph it becomes intractable. However, we try to change the graph to its most tree-like structure, and then run the message passing on this graph. At the very high-level, the junction tree algorithm partitions the graph into clusters of variables, there variables inside the cluster are technically strongly correlated to each other. These clusters are called cliques. However, the interaction between the clusters will have a tree like structure. Which means, clusters will be only directly influenced by its neighbors in the tree. This lead to tractable global solutions of the local(cluster- level) problems can be solved.

Given a correlation graph generated from graphical lasso, we initially start by creating the maximal cliques. These cliques represents that each feature in the clique is highly dependent on its neighbors in the same clique. After we have the cliques, we run the prim’s Algorithm to get the maximum spanning tree. The MST is generated from the graph that has edges between the cliques if they share the same features. These are weighted edges and we define the weight as “Number of feature they share”. After we run the junction tree algorithm, we get the order in which we start filling the data while predicting the values.

4.2 Finding a good Junction Tree

By hand: Typically, if we have a very small dataset, we can try to find the junction tree by hand. While on the other hand, if we have the dataset huge enough, then it is more convenient to develop the algorithm to build the Junction Tree.

Using Variable Elimination: This algorithm generates a junction tree over the variables. Thus, it is possible to use the heuristics to define the ordering of the features. Once, we have the ordering, we can start the prediction of the data.

4.3 Inference based on message passing

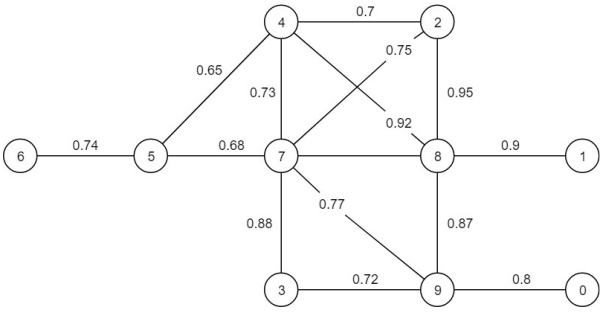
Once we've got the order of variable elimination, we can apply message passing or belief propagation to do the inference. An example of this is, if in the original graph we see a structure of "a-b-c", that shows 'b' have neighbors of 'a' and 'c'. But if 'a' is eliminated before 'b', then we can say 'a' is eliminated in terms of 'b'. Or in another word, we can see 'a' as the child of 'b' and 'b' as the parent of 'a'. For any vertex X in the graph, we have that:

$$P(X = x) = \sum_{parents} \frac{m_{parent}^{X}}{exp(A(\theta))} \prod_{children} m_{x}^{child}$$

where 'm' is called elimination function that computed based on the parameter ($\hat{\theta}$) of each clique that contains the current vertex, and the selected potential function. Based on a dynamic programming thought, for any vertex X, if all its children's elimination functions are already computed, the elimination function of X can be computed directly.

So, with the elimination order found by Junction Tree Algorithm, we can start with the leaf nodes, compute their elimination values and do the inference based on the formula above, and then pass the elimination value to each of their parents. This guarantees that given any feature, we don't need to worry about if its children's values are empty. But what if its parents are empty? To deal with this we first computed the mode value for each feature and store it, then if the parent feature is blank, we'll use the mode value to assign it and do the inference. (Moreover, after all blanks are filled in for the first time, we can do the whole procedures again and again till reach a convergence, which may improve the accuracy of prediction for a given query.)

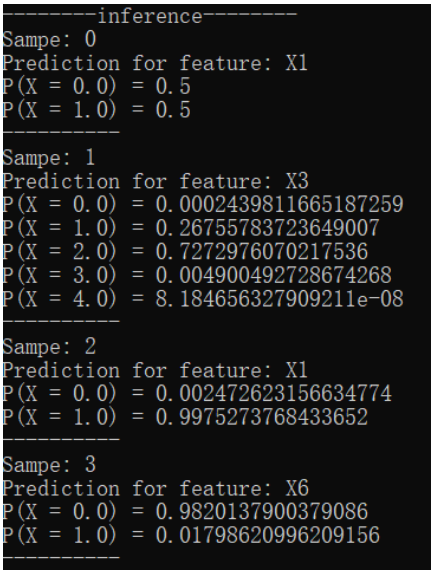
In our project we chose our potential function to be $\varphi(x_i, x_j) = e^{-|x_i - x_j|^2}$, and we only concerned the parameters of cliques within size 2 (i.e., only the nodes and edges in the graph). Our idea for the estimation is: given the original data, we randomly choose some samples and set some features to be blank, and then use our program to do the inference. After that, we compare the results of our program with the true answers to estimate the error. We first used a small testing graph to see if our program runs correctly, which looks like the following:



X0	X1	X2	X3	X4	X5	X6	X7	X8	X9
1	0	0	0	0	0	0	0	0	0.8
0	1	0	0	0	0	0	0	0.9	0
0	0	1	0	0.7	0	0	0.75	0.95	0
0	0	0	1	0	0	0	0.88	0	0.72
0	0	0.7	0	1	0.65	0	0.73	0.92	0
0	0	0	0	0.65	1	0.74	0.68	0	0
0	0	0	0	0	0.74	1	0	0	0
0	0	0.75	0.88	0.73	0.68	0	1	0.97	0.77
0	0.9	0.95	0	0.92	0	0	0.97	1	0.87
0.8	0	0	0.72	0	0	0	0.77	0.87	1

Based on this graph we randomly generated several samples and eliminate some values to be

blank, and try to apply the inference, which looks like the following:



So, our program is able to predict for each feature and output the probability following the order given by the Junction Tree Algorithm. However, the performance is not good. One reason may be about the choice of potential function, but a more probable reason is that the data we generated does not actually fit the distribution in the graph. We tried to do this for the graph we generated from the original data set, but unfortunately due to some error raised in the Junction Tree yet we haven't made it available on our original graph. For now we could just show a way of achieving this function.

(Note: Liya Guan was originally not a member of our group. Due to the mental health condition of her teammate we added her as a member halfway, but she submitted her own write-up, which is parallel to ours, so please notice that and also see her own write-up. Thank you!)