CS 536: Final Project - Data Completion and Interpolation

Yunhao Shi **NetID: ys536**

Overview

The dataset I choose is ML1 dataset which contain total 386 columns but only part of these features is correlated to each other and useful to interpolate. It contains 12 effects in a single experimental package across numerous labs of psychological study. There is a list of names of these different sections and their columns number in the *Appendix* part of the report.

Data representation:

These features contain a number of diverse data type. For example, the answer for *Anchoring* questions are real values, answer for *Allowed and Forbidden* are binary value and the *Art and Math* are ordered discrete values.

To deal with different types data, I first build two different model to handle with ordered discrete value and categorical value. For categorical data I use one hot encoding to embed the data to vectors with 0 and 1s.

Model Selection:

I select Variational Autoencoder (VAE) to learn and interpolate the data. It is a powerful neural network of unsupervised learning to compress the input data into latent feature space and then reconstruct the data from that latent feature space. The assumption is that the features are correlated in some degrees and the Autoencoder can fill the missing from the known features.

The prime objective is to reconstruct the data as precise as it could be. So there will be a loss function to measure how good the Autoencoder can regenerate the data, assume input is x and output is $\hat{x}$, we have

$$\text{Reconstruction Loss} = \text{diff}(x - \hat{x})$$

One more thing is that we predefine the distribution of latent feature z is assembling to the gaussian distribution as p(z), the encoder generates the distribution from the input x is $q\phi(z|x)$, we can get the formula $KL(q\phi(z|x) \parallel p(z))$ to represent the difference between those two-distribution using KL divergence metric.

So, the final loss function the Autoencoder try to minimize will be

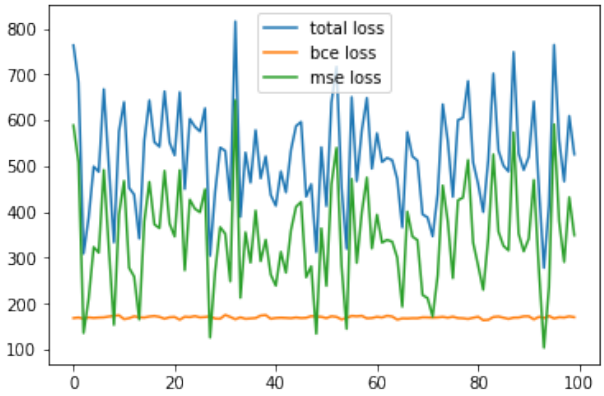
$$\text{Loss} = \text{Reconstruction Loss} + \text{KL Divergence loss}$$

In terms of the reconstruction loss, the two models have different loss function. The first model to fill the ordered discrete variables use minimum square loss (MSE) and the other use binary cross entropy loss (BNE) to measure the difference.

Training Algorithm:

To fit the data into Autoencoder, the most common way is using the gradient decent to update the weights of our neural network. In this project, I choose the small batch gradient decent (SGD) method to train the network. The optimization target is to minimize the reconstruction loss value and Kullback–Leibler (KL) divergence of the latent variable distribution with normal distribution.

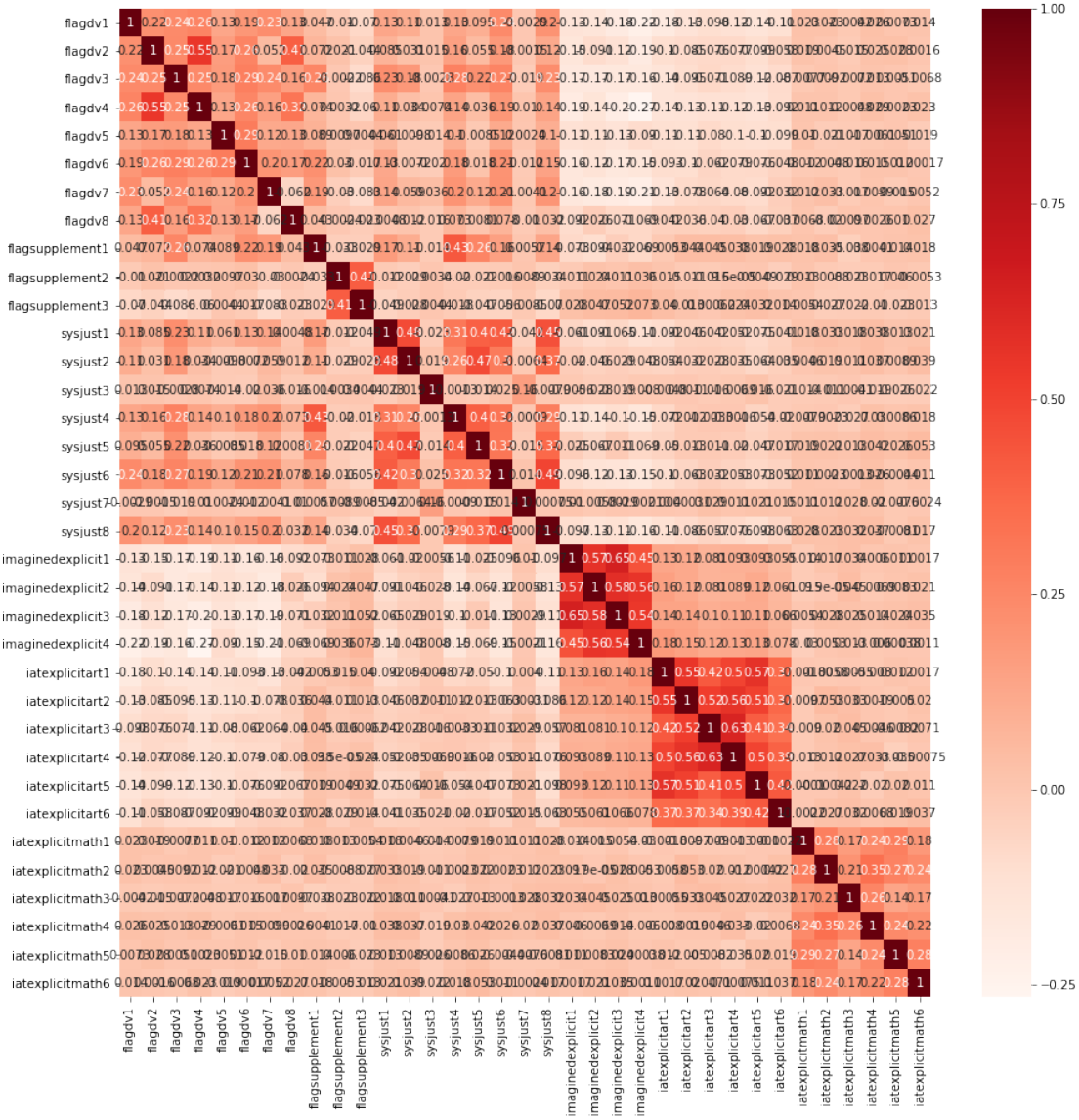
Using the SGD method can train features of same type. But when we use the SGD to train the mixture model, whose loss function is the sum up of MSE loss and BNE loss.



We can see that the value of MSE loss is much larger than the BNE loss. So the we can see that the gradient is mainly calculated by the continuous value and the ordinal values. The BNE loss which measure the categorical prediction loss remain unchanged.

Ordinal Feature Analysis

The ordinal variables are variable whose value exists on an arbitrary scale where only the relative ordering between different values is significant. For example, the score from 1-9 can reflect the level of preference over art or math. First we analysis the correlationship



Here we will first plot the Pearson correlation heatmap and see the correlation of independent variables with the output variable MEDV. We will only select features which has correlation of above 0.5 (taking absolute value) with the output variable.

The correlation coefficient has values between -1 to 1

—A value closer to 0 implies weaker correlation (exact 0 implying no correlation)

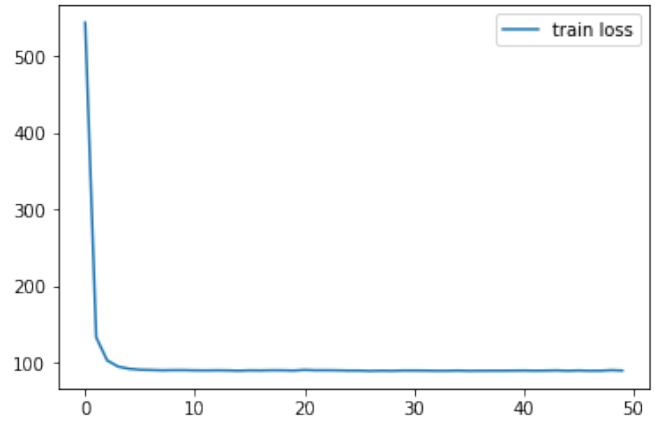
—A value closer to 1 implies stronger positive correlation

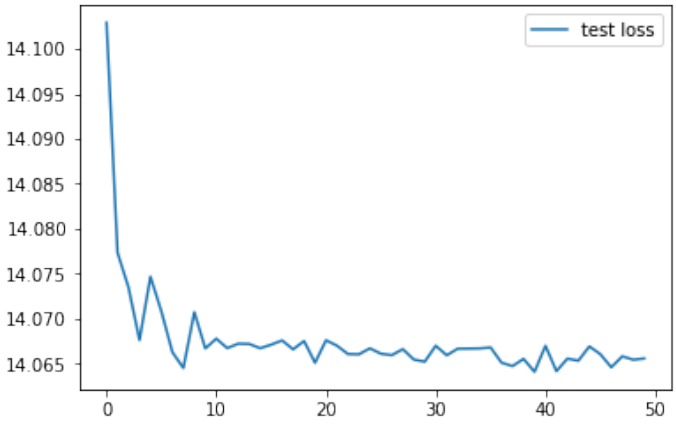
—A value closer to -1 implies stronger negative correlation

To train first model, I select 35 features which are all ordinary discrete variables, total 5698 samples. We can see that the correlation values of some entries are really small. These features can be recognized as irrelevant features. Since the neural network is for multivariate regression and one of the assumptions of linear regression is that the independent variables need to be uncorrelated with each other.

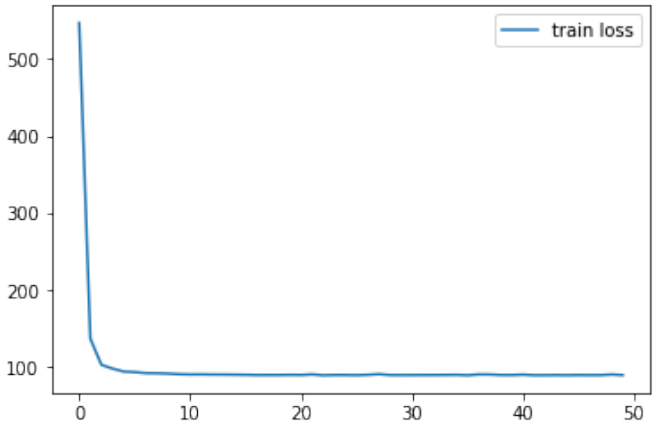
Training and Evaluation Process

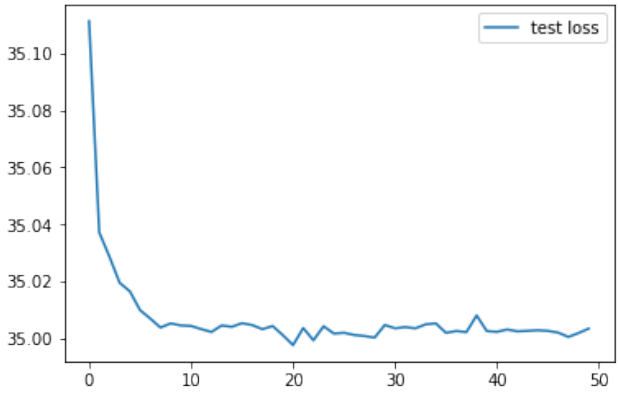
First, we just throw all the data into the Autoencoder and see the result. Although some features are irrelevant to other, but the model can still be trained to minimize the loss function overall. The missing method is random dropping, just randomly drop the entry in the dataset. I set the drop rate is 0.2.





We can see that as the training loss decrease, the test loss also decreases.
Then we set the drop method as uniform dropping, which will drop continuous part of the dataset. We can see the value of test error increase a lot comparing with the one using random drop. But the model can still work on it. It indicates that the model will perform better when it can get some information from every section of data than it get all information of several sections but without the rest of sections.





Categorical Feature Model

The second model is primly focus on interpolating categorical value, whoser input will binary or multi-categorical values. To represent the data, I use the *pd.get_dummies* function to encode the features into vectors. The original data has 413 rows with 8 features and the encoded data is represented by a zero/one matrix with 413 rows and 20 columns.

```
reciprocityothera_No                    413 non-null uint8
reciprocityothera_Yes                   413 non-null uint8
reciprocityusa_No                       413 non-null uint8
reciprocityusa_Yes                      413 non-null uint8
allowedforbiddena_No                    413 non-null uint8
allowedforbiddena_Yes                   413 non-null uint8
flagtimeestimate1_Afternoon             413 non-null uint8
flagtimeestimate1_Evening               413 non-null uint8
flagtimeestimate1_Morning               413 non-null uint8
flagtimeestimate2_Afternoon             413 non-null uint8
flagtimeestimate2_Evening               413 non-null uint8
flagtimeestimate2_Morning               413 non-null uint8
flagtimeestimate3_Afternoon             413 non-null uint8
flagtimeestimate3_Evening               413 non-null uint8
flagtimeestimate3_Morning               413 non-null uint8
flagtimeestimate4_Afternoon             413 non-null uint8
flagtimeestimate4_Evening               413 non-null uint8
flagtimeestimate4_Morning               413 non-null uint8
dtypes: uint8(20)
```

```
(413, 20)
[[0. 1. 0. ... 0. 1. 0.]
 [1. 0. 1. ... 0. 1. 0.]
 [0. 1. 0. ... 1. 0. 0.]
 ...
 [0. 1. 1. ... 0. 0. 1.]
 [1. 0. 0. ... 0. 1. 0.]
 [0. 1. 0. ... 0. 1. 0.]]
```

The drop method is set as random drop with drop rate at 0.1.

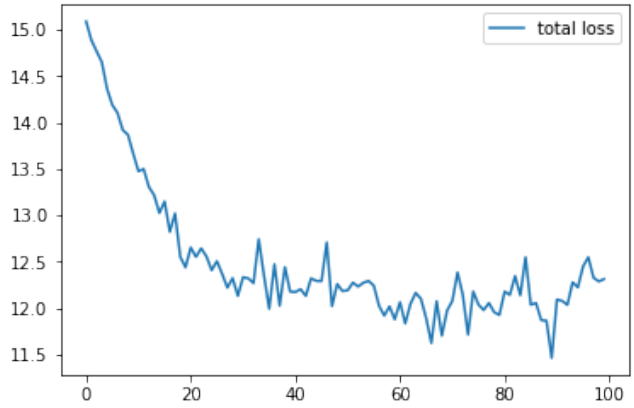
```
(413, 20)
[[nan nan  0. ...  0.  1.  0.]
 [ 1.  0.  1. ...  0.  1.  0.]
 [ 0.  1.  0. ...  1.  0.  0.]
 ...
 [ 0.  1.  1. ...  0.  0.  1.]
 [ 1.  0.  0. ...  0.  1.  0.]
 [ 0.  1.  0. ...  0.  1.  0.]]
```

Analyzing the Data

Since we use one hot encoding to turns the 8 features into 20, meaning that each feature has about 2 encoded values. So reduce any of them may lose some useful information. So unlike what we have done to analyze the ordinal variables. We just keep them unchanged.

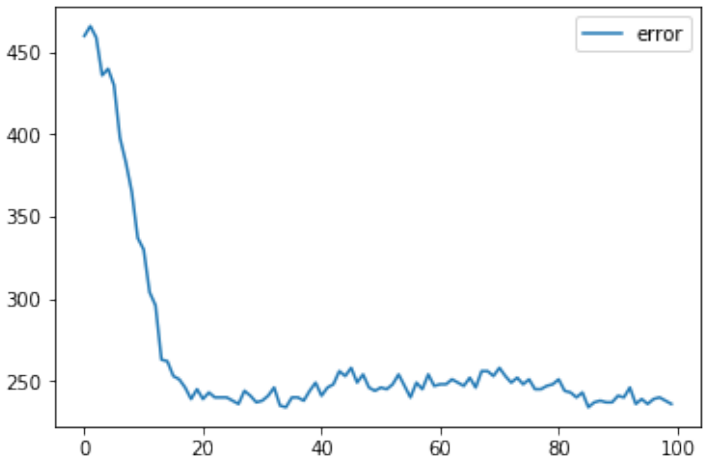
Training and Evaluation Process

Since all the values are in the range (0,1). A sigmoid function is implemented at the output layer to squeeze the value. In the same time, the reconstruction loss function is changed to binary cross entropy loss (BNE).



We can clearly see that as the training number increase, the total loss decreases from 15.08473113200855 to 12.314599579818024.

To measure the effect of the model, we still use the mask to hide some entries in the input data and compare these entries with reconstruct ones. The number of incorrect predictions decrease as the number of training epoch increase. At the beginning, the incorrect prediction is more than 450 and the number decrease to less than 250 after 20 training iterations.



Mixture Model

In this last model, I want to combine several features together including categorical value, ordinary discrete value and real value. The dataset contains 68 feature and 42 samples.

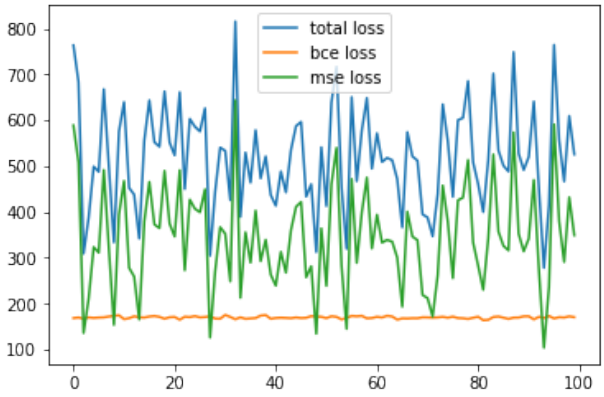
The gradient decent update method cannot work well when combine minimum square loss (MSE) and binary cross entropy loss (BNE) together. If just simply add up two loss values and Backpropagation, the weights cannot be optimized.

```
Epoch [928/1000], lter [1/5], Loss: 1308580.125000
Epoch [929/1000], lter [1/5], Loss: 3675745724522869847617506482585600.000000

---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-68-49cc0000e922> in <module>
    165         recon_batch, mu, logvar = model(batch_data)
    166         #print(actual[:,38:])
--> 167         loss = loss_function(recon_batch, actual, mu, logvar)
    168
```

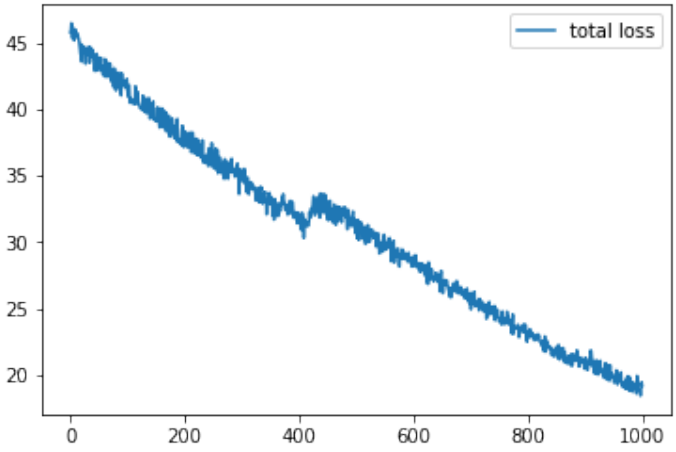
To avoid the gradient increase exponentially, I choose Huber loss as a better alternative to get rid of the problem when the error becomes too large. The loss function acts like L2 near the origin, and like absolute L1 when move away from the origin.

However, the result shows that the function cannot converge since the Huber loss still fluctuate.



To keep the value of loss in the same range, I just use the sigmoid function to squeeze all output prediction and use sigmoid function to squeeze the original data too. BNE loss function is used to measure the overall reconstruction loss. Finally, the training algorithm minimize the loss.

The following picture shows that the BNE loss decrease as the training number increase.


total loss
45
40
35
30
25
20
0
200
400
600
800
1000

Bonus

This project has three potential bonuses available.

- Bonus 1: Build your system to include processing and analysis of natural language answers, to try to use them to inform prediction of other features. How can you represent natural language answers in a useful way? What language processing is necessary to be able to use them? Do they actually provide useful information for the prediction problem?

In ML1 dataset, we have the text feedback from the participants. In the problem "imagineddescribe", the participants are required to describe what they just imagined for 1 minute. Most contents are totally irrelevant to our features. But some of feedback still can provide useful information to inform prediction of some features

For example, one item of the feeback is that

```
You learn about their customs and culture and are able to compare them to
your own customs. You are able to do away with certain stereotypes that yo
u have heard about Muslim people and make up your own mind about them and
their beliefs. You realize that not everyone thinks and looks like you do
but that does not mean that they should be treated any different than you.
```

The answer give exactly the information of the altitude toward Muslim groups. We can use the NLPK Python library to extract the useful information.

If we extract all nouns and all adjective words and then to extract the key word and compare them with the questions asked. We can use the information to change the score if some prediction values.

- Bonus 2: Build your system to include prediction about the natural language answers. Based on available features, what can you predict about a person's answers to natural language questions (if not the actual answers themselves)? How can you assess the quality of these predictions?

- Bonus 3: Build your system to include generation of natural language answers to questions, based on available features. How can you model this generation problem, and how can you evaluate the quality of the answers?

Appendix

1. Sunk costs (Oppenheimer et al., 2009).

sunkcosta
sunkcostb

2. Gain versus loss framing for combating disease (Tversky & Kahneman, 1981).

33-34

diseaseframinga
diseaseframingb

3. Anchoring

22-29

anchoring1a
anchoring1b
anchoring2a
anchoring2b
anchoring3a
anchoring3b
anchoring4a
anchoring4b

4.Retrospective gambler's fallacy (Oppenheimer & Monin, 2009).

52-53

gamblerfallacya
gamblerfallacyb

5. Low-vs.-high category scales

scalesa
scalesb

6. Norm of reciprocity (Hyman and Sheatsley, 1950).

93-96
reciprocityothera
reciprocityotherb
reciprocityusa
reciprocityusb

7. Allowed/Forbidden (Rugg, 1941).
20-21

allowedforbiddena
allowedforbiddenb

8. Quote Attribution (Lorge & Curtis, 1936).
90-91
quotea
quoteb

9. Flag Priming (Carter et al., 2011; Study 2).
37-51
flagdv1
flagdv2
flagdv3
flagdv4
flagdv5
flagdv6
flagdv7
flagdv8
flagsupplement1
flagsupplement2
flagsupplement3
flagtimeestimate1
flagtimeestimate2
flagtimeestimate3
flagtimeestimate4

82-85
noflagtimeestimate1
noflagtimeestimate2
noflagtimeestimate3
noflagtimeestimate4

10. Currency priming and system justification (Caruso et al., 2012).

sysjust1
sysjust2
sysjust3
sysjust4
sysjust5
sysjust6
sysjust7
sysjust8

11. Imagined contact (Husnu & Crisp, 2010, Study 1).

imaginedexplicit1
imaginedexplicit2
imaginedexplicit3
imaginedexplicit4

12. Sex differences in implicit math attitudes and relations with self-reported attitudes
54-70
iatexplicitart1
iatexplicitart2
iatexplicitart3
iatexplicitart4
iatexplicitart5
iatexplicitart6
iatexplicitmath1
iatexplicitmath2
iatexplicitmath3
iatexplicitmath4
iatexplicitmath5
iatexplicitmath6