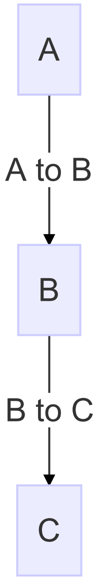
Alcoholic Detector

In this project we try to find alcohol consumption and health status of a students from their social characteristics.



Pre-processing The Dataset

The two dataset files those have been used in the project have been obtained from this page from Kaggle.

The dataset contains the following columns:

1. `school` : Student's school
2. `sex` : Student's sex
3. `age` : Student's age
4. `address` : Student's home address type
5. `famsize` : Family size
6. `Pstatus` : Parent's cohabitation status
7. `Medu` : Mother's education
8. `Fedu` : Father's education
9. `Mjob` : Mother's job
10. `Fjob` : Father's job

11. `reason` : Reason to choose this school
12. `guardian` : Student's guardian
13. `traveltime` : Home to school travel time
14. `studytime` : Weekly study time
15. `failures` : Number of past class failures
16. `schoolsup` : Extra educational support
17. `famsup` : family educational support
18. `paid` : Extra paid classes within the course subject
19. `activities` : Extra-curricular activities
20. `nursery` : Attended nursery school
21. `higher` : Wants to take higher education
22. `internet` : Internet access at home
23. `romantic` : With a romantic relationship
24. `famrel` : Quality of family relationships
25. `freetime` : Free time after school
26. `goout` : Going out with friends
27. `Dalc` : Workday alcohol consumption
28. `Walc` : Weekend alcohol consumption
29. `health` : Current health status
30. `absences` : Number of school absences
31. `G1` : First period grade
32. `G2` : Second period grade
33. `G3` : Final grade

The school is not important because we will try to find the generalized alcohol consumption. Unnecessary columns like `school` , `guardian` and `paid` have been removed from dataset.

The grade columns which indicate school success, have been simplified. School success calculated via the formula down below:

```
success = (G1 + G2 + G3) / 3
```

`age` and `absences` are the only columns that have numeric absolute value. So those columns remain as original.

`health` , `Dalc` , `Walc` , `goout` , `freetime` , `failures` , `traveltime` , `studytime` and `famrel` columns have the data that parsed as categorical despite they have numerical quantity values. So their data have been become between 0 and 1.

`sex` , `address` , `famsize` , `Pstatus` , `schoolsup` , `famsup` , `activities` , `nursery` , `higher` , `internet` and `romantic` columns encoded as binary because they have two values each.

The remaining columns are categorically processed with the method called *One Hot Encoding*

One Hot Encoding Problem

Tensors are multi-dimensional vectors like arrays and matrixes etc. In machine learning, data must be provided as a tensor. Categorical columns like *dog or cat* encoded with this method.

Pandas is a python library that allows us to process the data has rows and columns. I made a mistake when categorical data were processed with this method. I encoded the data and write them all to their own columns as array. Naturally TensorFlow raised an exception when I feed the placeholders with my wrong data. After I noticed what I did, I encoded the columns in seperate columns for every categorical value in each column.

For example we can encode `label` column in the following dataset:

index	value	label
1	45	'cat'
2	13	'dog'
3	42	'cat'
4	98	'rat'

Wrong encoding:

index	value	label
1	45	[1, 0, 0]
2	13	[0, 1, 0]
3	42	[1, 0, 0]
4	98	[0, 0, 1]

Correct encoding:

index	value	labelcat	labeldog	labelrat
1	45	1	0	0
2	13	0	1	0

index	value	labelcat	labeldog	labelrat
3	42	1	0	0
4	98	0	0	1

First, I used `pandas.get_dummies()` the result was enough for train but I faced another problem at test phase. When I created a dataframe for the test input, I noticed the column order of dataframe was changed after the dataframe to matrix conversion. I decided to change the conversion method and do it manually row by row.

Learning Phase

Simple, feed-forward neural network has been used in the project. The architecture of the neural network is:

```
Input[45] > Hidden_1[100] > Hidden_2[100] > Output[2]
```

Despite its simplicity, there was something wrong with the first teaching experiments. The error was decreasing a little and it was increasing continiously after it. I noticed that I was checking the error for test dataset. Then I added training error check. Voila! The error is decreasing for training set. I realized that the network was exposed to overfitting. Overfitting problem can be solved with dropout layers. I added a dropout layer after each hidden layer with probability of 0.5. After the dropout integration the network started to learn properly.

Usage of TensorFlow Sessions

After playing with TensorFlow a little bit, I learned a few things about TensorFlow sessions.

In TensorFlow, the defined placeholders must have been fed with data via the `feed_dict` argument in `Session.run()` function. The data that requested from the running session must be specified in the `Session.run()` function.

For example:

```
# What we give for feeding placeholders
feed = {
    input_placeholder: training_inputs,
    output_placeholder: training_outputs
}

#                         *Feed with this data*
#                                   v
training_cost = sess.run(cost, feed_dict=feed)
#                         ^
#           Evaluate and return this data
```

Cost and test values were fetched with this method in this project.

Am I Alcoholic?

The program has no UI so the result is displayed in the console. Now we have properly trained neural network. Unfortunately, I faced another problem. The test results are not stable. They gives different result for each evaluation. So I realized that I forget to deactivate dropout layers for testing. At last I disabled the dropout layers and neural network is good to go!

The result formatted as:

```
[
    [Workday alcohol consumption (0-1)],
    [Weekend alcohol consumption (0-1)]
]
```

Can it detect some alcoholics? Let's see...

My input:

```
# Orcan
input_row = ds.create_input_row(
                sex='M',
                age=22,
                address='U',
                famsize='GT3',
                medu=2,
                fedu=4,
                mjob='at_home',
                fjob='services',
                reason='reputation',
                traveltime=0.7,
                studytime=1,
                pstatus='A',
                success=0.4,
                failures=1,
                schoolsup=True,
                famsup=True,
                activities=False,
                nursery=True,
                higher=True,
                internet=True,
                romantic=True,
                famrel=0.9,
                freetime=0.3,
                goout=0.1,
                health=0.75,
                absences=10
            )
```

Result:

```
[[ 0.02392704  0.14131819]]
```

Not bad but I can say I consume more alcohol than this result. Lets give another shot with my friend.

```
# Ali
input_row = ds.create_input_row(
                sex='M',
                age=22,
                address='U',
                famsize='GT3',
                medu=3,
                fedu=3,
                mjob='at_home',
                fjob='other',
                reason='preference',
                traveltime=0.7,
                studytime=0.75,
                pstatus='A',
                success=0.6,
                failures=0.8,
                schoolsup=True,
                famsup=True,
                activities=False,
                nursery=False,
                higher=True,
                internet=True,
                romantic=False,
                famrel=0.65,
                freetime=0.5,
                goout=0.5,
                health=0.7,
                absences=10
            )
```

Result:

```
[[ 0.24936736  0.60401809]]
```

I think this time neural network predicted more accurately.

Conclusion

Neural networks can easily be exposed to overfitting. Dropout is a good solution as long as it is not left open in the testing process.

An UI is always required for good presentation of a project. Actually, good presentation is required for every project.

Resources

- Kaggle
- TensorFlow Examples
- Multi Layer Perceptron MNIST
- Dropout layers