Job Change Prediction

Description

Using TensorFlow, I have analyzed input parameters that would, hopefully, lead to predicting whether or not a candidate could be open to changing their jobs. Finding viable candidates early, and also eliminating nonviable ones, would quickly hasten talent acquisition. I have managed to attain only a 78% accuracy score, repeatable, on my testing dataset.

Data Preparation

Out of the columns in the CSV, I stripped the unique candidate ID and the city code. It's possible that the city code could play a role (certain city cultures could be relevant), but there wouldn't be enough data for each city in order for it to be relevant. It would also be incredible hectic to one-hot encode so many different cities.

I categorically labeled (one-hot encoded) the following columns: gender, relevent [sic] experience, enrolled university, education level, major discipline, company size, company type, and last new job.

I normalized the training hours and the years of experience fields on the interval [0, 1]. It's likely that one-hot would help this more, but that would be a lot of additional input neurons to handle considering there are dozens of choices for both columns.

Lastly, I shuffled the rows in the dataset. Then, I split the data into 85% training/15% testing. When fitting the model, 17.65% of the training data (therefore 15% overall) is split off for validation.

Network & Training Configuration

I have found the best results by using a network on the smaller side. My final network has two hidden layers: 128 neurons and 64 neurons. The output layer is a single neuron. The hidden layers use tanh as their activation, and the final layer is sigmoid.

Other attempts I have tried include: using relu for hidden layer activation, softmax on the output layer with two neurons, larger number of layers and fewer number of layers. Two neurons in the output layer relays the same sort of information as one binary neuron; however, results tend to be worse by about 2% when evaluating the model.

It's possible to reach 95% training accuracy with 400 epochs; however, this overfits the model and actually reduces the testing accuracy. The accuracy starts to drop around 50 epochs, which is probably where the model starts to become overfit.

Conclusion & Future Scope

After typing hundreds of lines of Python, but only keeping 60 of it, I believe that there isn't enough data to accurately train a model to predict whether or not someone would be willing to switch jobs. The dataset just isn't very large, and the number of people looking to change a job is 25%. The network could always output a 0 and be 75% accurate. Instead of 15000 or so items, I would be interested in seeing what the results would be like with millions.

There's also a possibility that there is no real correlation between our features and the label. It seems like it would be obvious that there would be, but at this point in time, I'm not entirely convinced without more data. There are also a few columns that have too many null values or minority values (such as gender).

There are other toolkits, such as sklearn, that allow you to modify your network to account for minority representation. I attempted to only stick with the assignment requirements of TensorFlow, but I would like to step out of this requirement to try to increase my accuracy.