2017-01-13

Tom Rochette <tom.rochette@coreteks.org>

August 30, 2025 — 861fb9d0

0.1 Context

0.2 Learned in this study

- Use of Keras
- Use of basic numpy functions (`np.asarray`, `np.zeroes`, `np.argmax`)
- Use of pandas `pandas.period_range`
- Use of peewee to read from MySQL
- How to create a on-hot vector from a list/set of labels

0.3 Things to explore

1 Problems faced

- Read data from a MySQL database
- Create a list of datetime spaced by a given period
- Initial terrible training/validation accuracy/loss (it was about 10-15% accurate)

2 Overview

Today I worked on what I consider a simple beginner's problem. The goal is to predict what a person/employee would do in a day by predicting what he will be doing if sampled every 15 minutes. I have about 40k training samples to learn from.

I formatted my input data in the following way:

such that, for an entry for 2017-01-13 12:30, I would have the following input vector

My desired output is a label such as "Project X". However, since I deal with numbers here, the labels have been converted to numerical identifiers such as "6". Then, since I want to train my neural network to learn about categories because the labels are not representing linear values, I convert these labels into `one hot` vectors.

The process looks as follow:

```
Project X -> 6 -> [0,0,0,0,0,0,1]
```

Since my data comes from a MySQL database, I do not convert the text into a numerical label myself, I simply use the primary key instead. You'll also notice that the index 0 will likely to always be 0 since MySQL does not allocate 0 as an auto increment key (it starts at 1). I do not mind losing a single memory space for the simplicity of dealing with the index right number.

I then built an extremely simple neural network using Keras.

where the input_dimension is 6 and the output dimension is based on the maximum project id retrieved from the database.

To train it, I decided to use the `nadam` optimizer with `categorical_crossentropy` as the loss function (what I'm trying to optimize). What `categorical_crossentropy` does here is attempt to reduce the overall difference between the network predicted output and a given sample, but where the output is generally a one-hot vector (it could also have been a multi category vector as well if I'm not mistaken).

After training, the accuracy was extremely low (<15%), which was unexpected. What I found particularly strange was that during training, the training/validation accuracy/loss would always end up being approximately the same, which was an indication that it either had learned it on first epoch, or that it wasn't learning.

In order to improve my results, I tried various things:

- Deeper networks
- Larger (breadth) networks
- Changed the activation function used at different layers
- Introduced dropout

The best I was able to get on that day was about 44.7% validation accuracy, which was at least twice better than what I had originally started with.

3 See also

4 References