Kwyk Following
Startup company in Education
Aug 7 · 6 min read

Multi-state LSTMs for categorical features

Context

Neural Networks are now widely used in many ways. From image caption generation to breast cancer prediction, this great diversity of applications is a natural consequence of the important variety of **neural architectures** (**Feed Forward** Neural Networks, **Convolutional** Neural Networks, etc…). Among all these architectures, **Long Short Term Memory** (**LSTM**)—a particular case of Recurrent Neural Networks—have proven very successful on tasks such as machine translation, time series prediction or generally anything where the data is **sequential**. This is mainly due to their ability to memorize relatively **long term dependencies**, which is achieved by taking into account previous information for further predictions.

> *But LSTMs alone aren't always enough. Sometimes we need to tweak these layers and adapt them to the task at hand.*

At **Kwyk**, we provide **online math exercises**. And every time an exercise is answered, we collect some data which is then used to tailor homework for each student using the website. In order to determine which exercises are the most likely to make a student progress, we need to know how likely he/she is to succeed on each exercise at any given point in time. And by being able to correctly predict these **probabilities of success** we can choose the homework that **maximizes the overall progress**.

This post aims to introduce our modified LSTM cell, the "**Multi-state LSTM**", which we based our model on to try and solve this problem.

A quick data overview

When an exercise is answered, we save information about both the student (or "user") and the exercise, along with an additional **score**

value that is either (0) or (1) depending the user's success. The information we collect is in the form of **categorical features** that tell us : "**which exercise is it ?**", "**what chapter is it ?**", "**what student is it ?**", "**from which grade is he/she ?**" … This results in features that have from tens to thousands of modalities, which we use to predict the "score" variable and get **success probabilities**.

user_id	grade	exercise_id	chapter	...	score
"1000"	"6th"	"401"	"001"	...	1
"1001"	"10th"	"1507"	"211"	...	1
"1001"	"10th"	"1511"	"211"	...	0
"1000"	"6th"	"452"	"003"	...	1
"1002"	"7th"	"603"	"061"	...	0
...	...	...	...	...	...

A sample of our data

LSTM-based model

There are two main motivations behind the choice of an LSTM-based model.

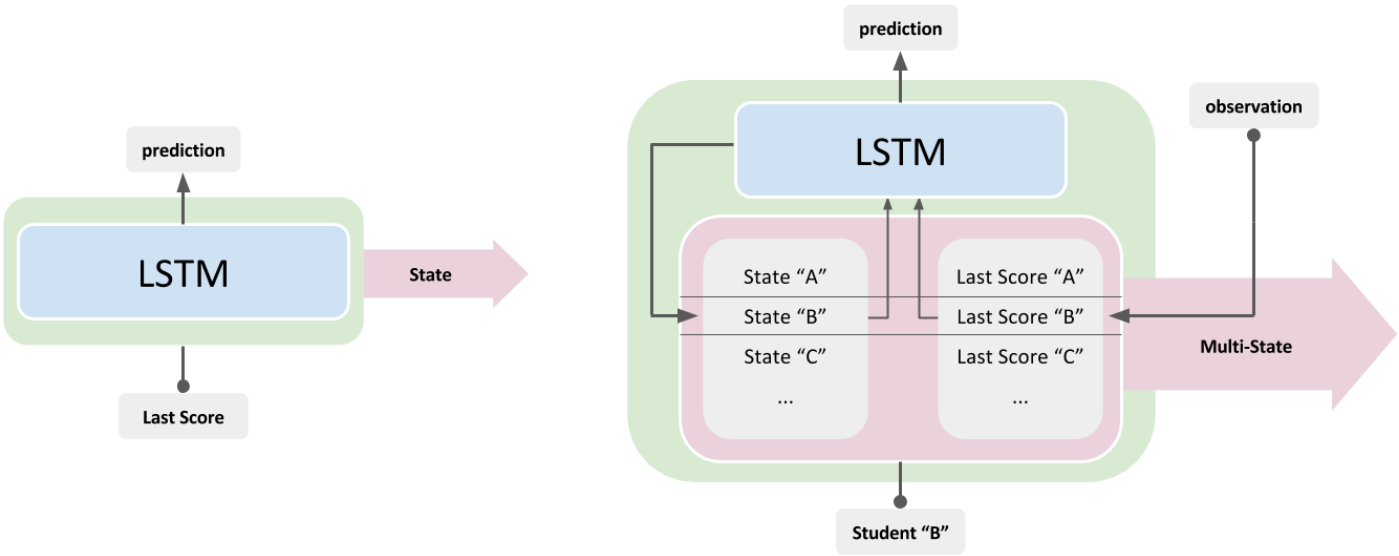
The first one comes from the fact that all the features we're using are categorical. That's the **real source of information** we want to learn from. And by manually cooking features such as moving averages of previous scores, we introduce **biases** due to the subjective choices we make. Therefore, we chose to rely on **neural networks,** directly feed our data and let the features form **automatically** and "**objectively**".

The second motivation concerns the choice of **LSTM**s among all available architectures. This is simply due to our belief that for each student the **probability of success** on each exercise *depends on all previous results in time* and therefore is **sequential**.

> *With this in mind, we thought of an architecture where for each categorical feature, a shared LSTM would keep a history of previous results for each modality.*

Let's take the "**user_id**" feature for example. In this case, we need to adapt an LSTM cell to memorize **on the fly but independently for each student** a history of how he/she was successful in the past. This

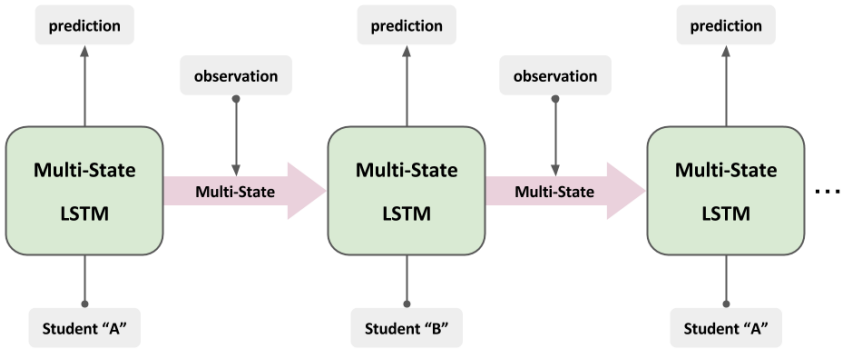
is done by adding what we call a "**Multi-state**" to the basic LSTM cell :



Left : Basic LSTM cell / **Right** : Multi-state LSTM cell. Note : If you are not familiar with LSTMs you can refer to this great post by Christopher Olah.

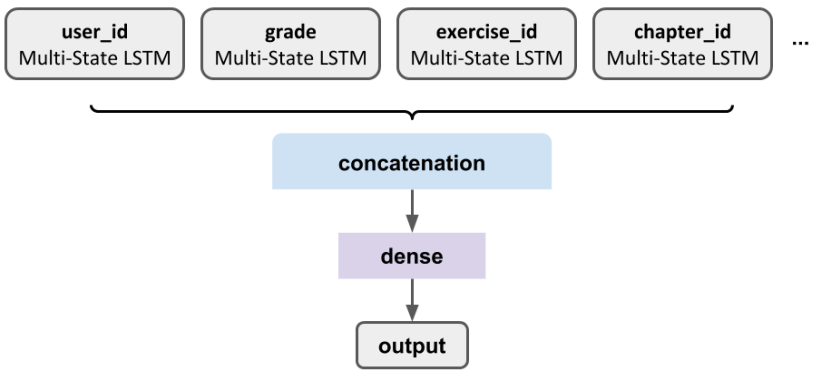
Every time a student's label is fed to the Multi-state LSTM cell, it starts by looking up the corresponding **state** and **previous score**. Then, the student's state is sent to **update** a **shared LSTM cell**. In meantime, the previous score is fed to this LSTM cell which produces an **output**. Next, the resulting **state** goes back to **update** the student's data in the Multi-state. Finally, once we observe the actual **score**, this value is in turn sent to **update** the student's previous score in the Multi-state.

This modified version of the basic LSTM cell works similarly but has an additional advantage : it can now directly take in a sequence of student labels as inputs :



An unrolled Multi-state LSTM.

For the sake of our previous example we considered the "user_id" categorical feature, but this can actually be applied to all of our categorical features. In fact, using a Multi-state LSTM on the "**exercise_id**" feature for example would result in it learning **historical success rates** for each exercise. From here, the complete network we used is pretty straightforward :



The complete graph we used to solve our problem.

All in all, this architecture tries to learn separate histories for each modality of each categorical feature then uses the LSTM's outputs to predict a success probability for a given student on a given exercise.

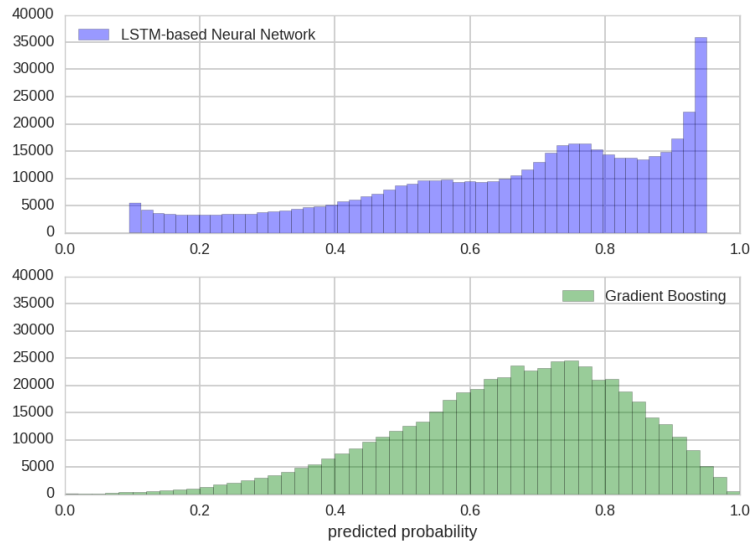
Results

Baseline

To assess our LSTM-based model we need a **baseline**. Before trying any Deep Learning methods, we used to have a boosting algorithm (**Gradient Boosting**) that used some handmade features. From all the categorical features, we cooked up some **fast and slow moving averages** of previous scores per each modality of each feature. Then, we fed these features to a Gradient Boosting classifier and let the magic happen.

Comparison

Both classifiers have the same approach. In fact, the baseline model uses handmade moving averages as features and the LSTM-based method automatically learns these histories to predict the scores. Sure enough, both algorithms got similar Accuracy Scores (Baseline : 74.61%, LSTM-based network : **75.34%**). But the predicted distributions are very different :



We can see that the LSTM-based model is much better at segregating students.

In fact, the LSTM-based model seems to *separate 3 to 4 different populations* :

- One for students who shouldn't be able to answer correctly (probability<10%)
- One for those who have a moderate to good chance to succeed (50<probability<80%). These can be students for whom we don't have enough data, or simply good-average students.
- One for those who have a very important chance of succeeding (probability>90%).

Conversely, the baseline only predicts a skewed normal distribution around the overall average success rate with no particular discrimination.

But to further investigate the difference between the two models, we need to have a more detailed look at the MSE. In fact, judging by the MSE alone, the two models have similar performance :

- Baseline : 0.17
- LSTM-based network : **0.16**

But as we discussed in a previous post, when real randomness is associated with the targets (here, predicting human behavior) we get a better assessment of the performance of a classifier by looking at its **reliability measure (REL)**.

Reliability is computed by sorting the data into bins of similar predicted probabilities. Then, we compute an MSE for each bin against the average target in the bin. Finally, we take an overall average to get our REL value.

The metrics obtained for each model are :

- Baseline : 3.86 e-3
- LSTM-based network : **2.86 e-4**

Indeed, we see that the LSTM-based network is more than **10 times more precise** than the gradient boosting baseline, which should explain the **better segregation** we observed when comparing both distributions.

Conclusion

To sum up, we've seen on a real life example how **LSTM**s can be effective for **sequential data** classification. But we also saw how our **modified version** of the LSTM—the "**Multi-state LSTM**"—was able to reach a greater performance *without any preprocessing or feature engineering at all*. In fact, this new LSTM cell can directly take in a sequence of labels as inputs, which means that it **can be used categorical features only** and still produce good results.

To **further improve** on this Multi-state LSTM, a next step would be to take into account the **correlations between multiple labels**. In fact,

when predicting the performance of a student on a given pair of similar exercises, the predicted probabilities should be very similar. One way to try and ensure this behavior could be to integrate an exercise embedding into the network and let it **learn the dependencies**.

Author : Hicham EL BOUKKOURI