Comparing a temporal algorithm with non temporal algorithms for predicting the mood of patients

Alexander van Someren, Casper Thuis, and Wolf Vos

Universiteit van Amsterdam,
Vrije Universiteit
{alexander.vansomeren, casper.thuis,wolf.vos}@student.uva.nl

1 Introduction

Temporal learning algorithms are commonly used in various fields such as finance and health care. This research looks at the difference in performance of a temporal algorithm and a non temporal algorithm for predicting the mood of a patient (or subject) "on the next day", based on data of the past five days.

The data set has the following form:

Time	**Patient**	**Variable**	**Value**
2014-04-27 10:44:59	AS14.28	mood	7
2014-04-28 12:31:04	AS14.28	appCat.communication	15.076
⋮	⋮	⋮	⋮

Table 1: Example of data set

From this data set we want to predict the mood on the next day given the measurements of the previous 5 days. Section 2.1 will discuss the data set as it is given. Section 2.3 discusses an approach to temporal learning considering only the mood variable. Subsequently, section 2.4 discusses a non temporal approach where different variables are used without explicitly using the temporal dimension.

2 Methods

2.1 Data

In the data set, 27 patients have been monitored. There exist about 200 observations of the mood for every patient. The amount of other variables differ significantly per patient. The patient with the minimal number of observations for all variables has 2,848 observations where the one with the most observations has 21,999. Table 2 shows the meaning of the variables. For the purpose of comparability, all patients with missing data are omitted, this results in 6 patients that are eligible for training and testing.

The "mood" variable has a mean of 7.0 and a standard deviation of 1.0, both the first quartile and the median are 7 (the third quartile is 8). This indicates that the distribution of the mood is very narrowly centered around 7. Only 2% of the values lie below 5 and 4% lie above 8. This gives a general idea about the domain and will be used for the discussion of the results.

2.2 Preprocessing

Before any algorithms can be applied to the data some preprocessing has to be done. To predict the mood "on the next day" based on the past five days, we need features that describe those five days.

To obtain these features, the raw data as seen in table 1 is first pivoted such that the "variables" column turn into separate columns with the corresponding "values" as values (note that this generates a sparse data set with many `NaN` values).

Next, for each patient, the data is "resampled" using `pandas`'s `resample`[1] function, where a resample rule is specified for each variable. These rules are shown in table 2 and can be summarized as followed:

- Variables that have a measured value in the form of a score, are resampled using the mean.
- Variables that have a measured value in the form of an event (that either did or did not occur) or in the form of a duration are resampled using a summation.

The resampling interval is set to one day, such that columns are now the averages and summations of variables during one day.

The last step is letting a sliding window move over this newly created matrix that gathers the information of the last five days and stacks these observations together (these are the features used for training) and finally adding the mood variable "on the next day" as a target variable.

2.3 Temporal learning

ARIMA During this project an attempt was made to predict the mood of the next day by looking at the moods of the past days. Performing Auto-Regressive Integrated Moving Average (ARIMA) with parameters p, d and q set to 1, 0 and 0 respectively yields relatively good results while training. But the model is very limited since we use the moods of the past days only, and not the other inputs. For the ARIMA implementation a package from Statsmodels was used, which can be found here [2].

[1] See http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.resample.html

Variable	**Description**	**Method**
mood	The mood scored by the user on a scale of 1-10	mean
circumplex.arousal	The arousal scored by the user, on a scale between -2 to 2	mean
circumplex.valence	The valence scored by the user, on a scale between -2 to 2	mean
activity	Activity score of the user (number between 0 and 1)	mean
screen	Duration of screen activity (time)	sum
call	Call made (indicated by a 1)	sum
sms	SMS sent (indicated by a 1)	sum
appCat.builtin	Duration of usage of builtin apps (time)	sum
appCat.communication	Duration of usage of communication apps (time)	sum
appCat.entertainment	Duration of usage of entertainment apps (time)	sum
appCat.finance	Duration of usage of finance apps (time)	sum
appCat.game	Duration of usage of game apps (time)	sum
appCat.office	Duration of usage of office apps (time)	sum
appCat.other	Duration of usage of other apps (time)	sum
appCat.social	Duration of usage of social apps (time)	sum
appCat.travel	Duration of usage of travel apps (time)	sum
appCat.unknown	Duration of usage of unknown apps (time)	sum
appCat.utilities	Duration of usage of utilities apps (time)	sum
appCat.weather	Duration of usage of weather apps (time)	sum

Table 2: Different variables in the data set with corresponding resampling methods.

Recurrent Neural Networks Recurrent Neural networks (RNNs) are a great way to model complex temporal data but since the data has limited data points RNNs are not implemented. The reason for this is that RNNs need tremendous amounts of training data to converge to an optimum. Thus to our best knowledge, RNNs are not suited for this problem.

2.4 Non temporal learning

SVM One of the classifiers used is a SVM classifier. The advantages of using a SVM classifier is that it prevents overfitting and can easily be adapted by use of a different kernel [1]. The change of kernel influences the decisions boundaries in the SVM and can therefore model the data differently. In this research three different kernels (linear , radial basis function and Sigmoid) are compared for the data. The SVM implementation used is from the sklearn package [2], available here [3].

Random Forest The random forest algorithm is a boosting method, that combines multiple random decision trees. The random forest algorithm has the advantage that it has no hyper parameters to tune and that the information gain

[2] http://statsmodels.sourceforge.net/0.6.0/dev/index.html
[3] http://scikit-learn.org/stable/

decision rule can be used to determine interesting aspects of that discriminative structure of the features, [1]. The random forest implementation that was used is also from the `scikit-learn` toolbox [2].

Neural Network Neural networks have proved be very useful for many classification and regression tasks. In this project an attempt is made to predict the mood score of a person using the aggregated features of the past five days. The data is aggregated as described in table 2. The neural network is randomly instantiated with 20 hidden layer nodes, one output node (which represents the mood score) and a changing amount of input nodes. The amount of input nodes is equal to the amount of features used to train the model. The model is trained using the PyBrain [3] library [4]. The training is using back-propagation with momentum set to 0.1 and weight decay to 0.01.

In an attempt to better grasp the nature of the data, i.e. deviations from the mean, it is also investigated if standardizing the data improves performance. The intuition behind the is that deviations are interesting and not the regular value of an input. Thus all the inputs and the output are standardized in the experiments. This was also done to be able to train the neural network on multiple persons at the same time, since the mean mood of a person will most likely be different from the mean mood of another person and by standardizing it is easier to generalize over the different patients.

2.5 Baseline

To evaluate the possible algorithms, two baselines were constructed. The first baseline simply predicts the mean. As mentioned in section 2.1, the standard deviation of the *mood* is low, thus predicting the mean is a good base line. Another logical baseline is to predict that the mood is the same as yesterday.

3 Results

3.1 ARIMA

The ARIMA algorithm has a very low MSE but since some implementation issues arose, the testing is also done on the training data, thus these results should not be taken into account.

3.2 SVM

The SVM preformed similar around all kernels as can be seen in table 4. Therefore, as choice was made to select the rbf kernel for the data. However, other kernels would be consider viable as well.

[4] http://www.pybrain.org/

Algorithm	**1 Patient**	**6 Patients**
Baseline (7)	0.72	0.98
Baseline (previous day)	0.32	0.47
SVM `mood`	0.93	0.50
SVM `¬mood`	1.33	1.01
SVM `all`	0.41	1.03
Random Forest `mood`	0.77	0.53
Random Forest `¬mood`	0.51	0.45
Random Forest `all`	0.39	0.53
Neural Network `mood`	**0.22**	0.35
Neural Network `¬mood`	0.35	0.51
Neural Network `all`	0.40	**0.33**

Table 3: Mean Squared Error for different algorithms on 1 and 6 patients

Kernel	**MSE**
rbf	1.01
sigmoid	1.02
linear	1.20

Table 4: Kernel method mean squared error

3.3 Random Forest

The random forest algorithm did not perform better than "previous day" baseline as can be seen in table 3. One could expect that it would just base its decisions on the mood on the previous day. However, when inspecting the information gain, this did not appear to be the case. The screen activity on the previous day as well as the usage of social apps the day before that both contributed more than the mood.

3.4 Neural Network

The Neural Network performed rather good as is visible in table 3. The network has the best performance for the 1-patient experiments and the 6-patients experiments.

For the 1-patient experiment no standardization is used and using only the mood scores of the past five days yields the best results (MSE = 0.22). The only downside is that the model is overfitted on one patient and will perform worse on the complete data set. For the 6-patient experiment standardization is used because that yielded the best results. The network performed best when using all the input variables (MSE = 0.33).

Standardization Another interesting discovery is that standardization is not always increasing performance. For the grouped experiment with 6 patients the algorithm benefits from standardization but for the 1 patient experiment the algorithm does not. This is in line with our intuition; that personal averages should not be taken into account. If you look at table 5 (columns 1, 3 and 5) it is clearly visible that for the inter-patient experiments standardization is non-beneficiary. If you look at the other columns (2, 4 and 6) you can see that the MSE is lower with the standardization. This could be taken into account when designing a real-life application that uses this data.

	Mood-1	**Mood-6**	**¬Mood-1**	**¬Mood-6**	**All-1**	**All-6**
Standardization	0.340	**0.352**	0.901	**0.352**	0.488	**0.327**
No Standardization	**0.216**	0.633	**0.514**	0.405	**0.401**	0.545

Table 5: Standardization influence on 1 and 6 patients experiments

4 Discussion

The task at hand appeared to be a rather difficult one. The relatively good performance of the baselines are consistent with the observation that the mood of a patient is narrowly centered around the mean. Also, since we are predicting the average mood the next day (most patients have multiple mood observations per day), the predicted variable will have an even narrower distribution than the mood as given in the original data set.

Therefore, it could be argued that the task at hand is too coarse in a temporal dimension. It seems likely that specific usage of your mobile phone correlates with certain patterns of behaviour and thus, it is possible that data about this usage could be used to predict the mood of a patient. However these (latent) patterns of behaviour could be very subtle and thus invisible when looking at daily averages.

We encourage future researchers of this data set to look at the data at a higher temporal resolution. Furthermore, when studying the mood of a patient, predicting strong deviations from the mean could provide a more interesting research domain than the complete prediction.

Bibliography

[1] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[3] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 2010.