Predicting mood using time-series analysis in mental health

Ioannis Gatopoulos[2654227] Filip Knyszewski[2656880]
Philipp Ollendorff[2655580]

September 1, 2019

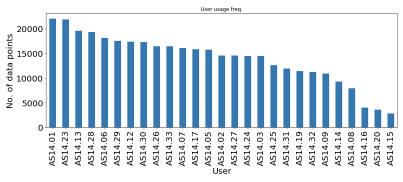
1 Introduction

The increase in computing power and availability of data lead to the recent success of predictive modelling. In this report, we will showcase the data mining workflow using a challenging temporal data set from the field of mental health. The task is to predict a patient's mood on any given day. In general, we can summarize our approach into three steps: First, we will pre-process the data. Second, the data is fed to various models. Last, we analyse the results and conclude with the best predictive model.

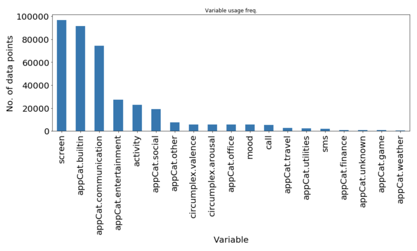
2 Pre-process the data set

2.1 Inspecting the Data

The raw data file given contains five columns and 376,912 rows and can therefore initially be categorized as medium-sized. We will shortly explain each of the columns: First we have a simple row **index**, indicating which observation we are currently looking at. The second column is our main focus, as it represents the unique **ID** of each patient. Third, we get a combined **Timestamp** of the day of the year and time of day. The next column is an indicator which **Variable** this observation is concerned with. At last, the fifth column is the actual **Value** of this observation, which is the main feature of this data set. The three biggest features in terms of the amount of observations are: screen (96,578), appCat.builtin (91,288) and activity (22,965). The aim of this report is to predict the value for mood, for which we have 5,641 data points. An overview is shown in Figure 1



(a) User usage frequency



(b) Variable usage frequency

Fig. 1: Visualization of data points for each of the 27 patients (Left) and 19 features (Right)

2.2 Cleaning the Data

A data analysis showed that 202 NaN values exist in the data set. There are various ways of dealing with NaN values, the simplest being deletion of the respective rows. Other possibilities include setting those cells to zero or to the mean of this patient. Because the number of NaN values is very small compared to the size of the entire data set (about 0.05%) we chose to simply delete them. We also chose to split the timestamp column into two columns: date and time. This will help in aggregating the data for each day.

2.3 Aggregation of Data

We transformed the data set to more accurately describe individual patients at a given day, by including all features as individual columns, therefore widening the table to 22 columns in total. This introduces many cells with value zero, as we do not have observations for every single feature at every single time point. In contrast, it also eliminates a lot of redundancy, as we only save each pair of patient and time once. By transforming the data set this way we can effectively use all features at the same time. In addition, we aggregated all data points for each patient for each day by summing them, assuming that the sum over all features retains more information than e.g. the mean. This last step effectively eliminates the time-id column.

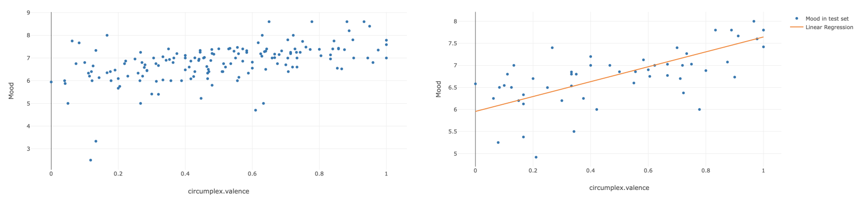
2.4 Correlation and further inspection

Further investigation of the data revealed the correlations between features and mood. An excerpt of the highest correlations is shown in Table 1.

We can use this information to later focus on specific features that give us more information on the mood of that patient. Using the highest correlation *circumplex.valence* feature we can already make quite accurate predictions. They are visualized in Figure 2.

	call	**weekdays**	**circumplex.arousal**	**activity**	**circumplex.valence**
mood	0.07	0.09	0.13	0.14	0.41

Table 1: Most important correlations of the data set



(a) Data of train set of `circumplex.valence` feature

(b) Linear Regression and projection of test data. **MSE**: 0.25

Fig. 2: Fitting of linear regression for variables `circumplex.valence` (x-axis) and `mood` (y-axis).

Additionally we computed a new feature by converting date to weekdays. The relationship between weekdays and mood is shown in Figure 3. The graph confirms our expectations: Mood is lowest on Mondays, gradually increases as the weekend approaches with a peak on Saturday followed by a sudden decrease.

2.5 Outlier detection

Data sets usually have an underlying distribution, but may also have some random noise due to technical failures or human errors. It is therefore a good idea to get rid of outliers before fitting to any model. We did this using the **Tukey rule** (also known as Tukey's fences [1]). It relies on the interquartile range (IQR), which is the distance between the values of the first quartile and the third quartile. The Tukey rule finds appropriate boundaries per feature per patient to determine outliers in a data set. The boundaries are as follows:

$$[Q1_{(f,p)} - 1.5 \cdot IQR_{(f,p)}, Q3_{(f,p)} + 1.5 \cdot IQR_{(f,p)}]$$

Anything outside these values is considered not representative of the underlying distribution

Now there are two ways of coping with outliers: Deleting the respective row or capping the value to fit into the boundaries. Any deletion of rows not

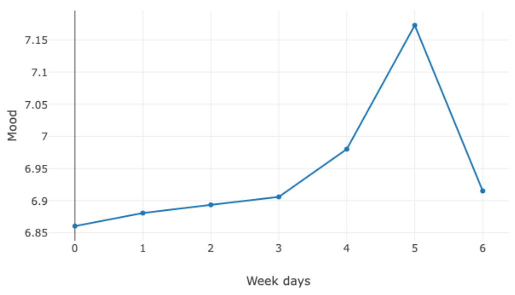


Fig. 3: Relationship of weekdays with mood on average over all patients. From 0:Monday to 6:Sunday

only removes the outlier but potentially valuable data. Therefore, we chose to replace all outliers with their maximum or minimum value respectively.

2.6 Normalization

As the last step we normalize the data. The terms standardization and normalization are often used interchangeably when referring to feature scaling. However, these are slightly different operations: Standardization refers to scaling a set of values so that they have a mean of zero and a standard deviation of one. Normalization refers to scaling a set of values so that the range is between zero and one. We chose the latter option.

2.7 Definition of attributes and targets

To train a supervised learning algorithm for traditional machine learning methods we will need to use specific attributes to learn from and target values to predict. For this assignment the targets were pre-defined as the average mood per person over each day. The machine learning method uses both the most correlated feature and all features combined, while the temporal learning algorithm does not have any attributes, it only looks at the target values. Now we are all set to actually feed this data to a model and make predictions.

3 Learn using the data set

3.1 Performance metrics

Our goal is to predict the average mood for a certain day, which can be any real number between 0 and 10. Ideally our models output such continuous values. It is for this reason that accuracy is not necessarily a good measure. Accuracy is more appropriate in discrete cases, such as a binary or multi-class classifier. For regression the most standard measure is the Root Mean Squared Error (RMSE), which we will be using as our performance metric.

3.2 Benchmark model

As a naive but intuitive baseline we implement a simple benchmark model to compare against. It predicts any patients mood to be exactly the same as the day before, thereby disregarding any features.

3.3 Traditional Machine Learning approaches

Considering a regression problem, given a training set of data, we wish to make predictions for new inputs x that we have not seen in the training set. Thus it is clear that the problem at hand is inductive; we need to move from the finite training data D to a function f that makes predictions for all possible input values. To do this, we must make assumptions about the characteristics of the underlying function, as otherwise any function which is consistent with the training data would be equally valid. In general, we can describe two common approaches. The first is to restrict the class of functions that we consider, for example by only considering linear functions of the input (e.g. $f(x) = mx + c$). We used this approach to calculate the mood wrt `circumplex.valance` feature, illustrated on figure 2. The second approach is to give a prior probability to **every** possible function, where higher probabilities are given to functions that we consider to be more likely, for example because they are smoother than other functions.

Gaussian Processes (GPs) belong to the second family of approaches and provide an alternative approach to regression problems [2]. It is a 'non-parametric' model as it finds a distribution over the possible functions $f(x)$ that are consistent with the observed data. As with all Bayesian methods, it begins with a prior distribution and updates this as data points are observed, producing the posterior distribution over functions.

To define a GP, we only need to be able to define a distribution over the functions values at a finite, but arbitrary, set of points, say $x_1, x_2, \ldots, x_n$. A GP assumes that $p(f(x_1), , f(x_N))$ is jointly Gaussian, with some mean $\mu(x)$ and covariance given by $k(x_i, x_j)$, where k is a positive definite kernel function. The key idea is that if x_i and x_j are deemed to be similar, then we expect the output of the function at those points to be similar, too. For example, if we know the value at one point x_i and the kernel proposes that x_j is close to x_i, then the distribution between them should be smoother than other points that are further away.

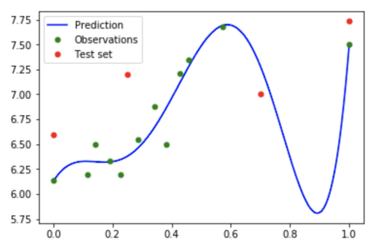
Besides their solid mathematical background, the motivation to use GPs to our problem is to provide us with a measure of **uncertainty** of our predictions. This and the fact that they are non-parametric are the elements that make GPs attractive.

First, we are going to build a separate model for every patient using only the training data, followed by measuring the MSE for the test data on our distribution, to get an intuition of how well our model generalizes. Then, we are going to do the same, but with all of the users (we will treat them as one) and hopefully come up with a more general model. Our experiments were implemented with an RBF kernel of variance 1.5 and length-scale 0.5.

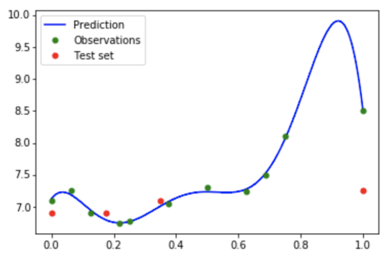
Models for individual patients

As we saw, the variable `circumplex.valence` correlates more than any other with `mood`. At this stage, we will fit a GP for every patient taking into account only this variable. This will give us an opportunity to visualize our results (the problem is now two-dimensional) but also, if with only one variable, we can achieve satisfactory results.

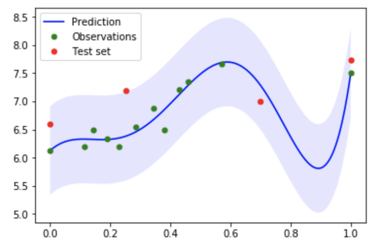
Figure 4 illustrates the resulting distribution of two patients, namely `"AS14.01"` and `"AS14.03"`. As we can see, the GP does a very good job fitting a "reasonable" function to the training data (green points). Also, we can see that at some regions where we do not have much data (like on 4b at $0.8 - 1.0$), it predicts smooth curves that connect the two points. The reason that it does not just connect them with a straight line, is that it takes into account how the underlying training data progresses and the kernels hyperparameters are optimized during fitting. Thus, it results in a more complex assumption. Moreover, both models perform also very good on the test set, as the former has a mean squared error of 0.24 and the other



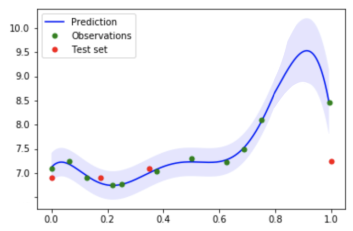
(a) Patient `"AS14.01"`. **MSE**: 0.24



(b) Patient `"AS14.03"`. **MSE**: 0.40



(c) `"AS14.01"` w/ uncertainty $\pm 1.96 * \text{std}$



(d) `"AS14.03"` w/ uncertainty $\pm 1.96 * \text{std}$

Fig. 4: Fitting of Gaussian Process for variables `circumplex.valence` (x-axis) and `mood` (y-axis).

of 0.4. The performance of all models for every patient has a mean MSE of 0.38, with a minimum value at 0.01 and maximum at 1.57. This gap of performance between the max and min, is due to the correlation between the mood of the person and the feature `circumplex.valence`; bigger correlation yields lower MSE. To illustrate how important the element of uncertainty is in our predictions, we also provide plots with and without the visualization of standard deviation (std). We can see that most of the test samples fall into the $\pm 1.96 * \text{std}$ (std of `"AS14.01"` 0.3 and `"AS14.03"` 0.1)

Going forward, we will try to use all 19 features of the patients to fit a high-dimensional GP. In our surprise, the models perform, on average, worse than before; MSE avg 0.89, min 0.09, max 2.76. Our intuition is that most of the features have nearly no correlation with the mood, causing the model to not generalize well.

A Model for all patients

At this stage, we treat all patients as one and try to fit a model to these observations, making it more generalizable. Although we expect this model to not perform as well as the individual models, our motivation is that we will receive a way of predicting new patients that the algorithm has never seen before.

Following the previous process, we will build two models; one using only the `circumplex.valence` feature, and another one that takes advantage of all of them. For the former, the MSE on the test set is 0.48 and the latter for 0.57. These performances are lower than the individual models with `circumplex.valence` feature (as expected). However, they outperform the individual models that take into account all of the features. This can be explained by the amount of available data and the high-dimensional regression problem: every model has only little data to fit a 19-dimensional curve.

3.4 Learning with temporal Data

As we have seen before, our data is of temporal nature and so intuitively a time series forecasting model should perform well on it. One such very popular model is the Autoregressive Integrated Moving Average Model (ARIMA), a generalization of the Autoregressive Moving Average model. This model uses past values of a certain feature to make predictions about their future values In essence this model tries to:

(AR) **Autoregression**: Make use of the dependent relationships lagged observations.

(I) **Integrated**: Use differencing of raw observations with the goal of making the data stationary.

(MA) **Moving Average**: Use dependencies between the observations and the residual error from the moving average of the lagged observations.

The ARIMA model is further defined by three parameters:

p: The number of lag observations that are part of the model (lag order). This parameter can be compared to the number of time series data points we think are relevant to predict the next.

d: The amount of time the raw data points are differenced (degree of differencing). Intuitively this parameter controls the degree to which it is likely that the mood will change today, if it has not changed for the last days.

q: The size of the window used to compute the moving average (the order).

Once again, one model was created for every patient. Naturally, prediction error varied largely between patients due to differences in data magnitude and mood variance. In figure 5 we can see the predictions of the best and worst performing ARIMA models. We used a short grid search to determine the optimal parameter settings. The values used for all models were: $p = 3$, $d = 1$ and $q = 0$. The final ARIMA model achieved a mean RMSE over all patients of 0.68.

4 Conclusion

In Table 2 we present the final results of each model and the benchmark.

	Benchmark	**ARIMA**	**GP**
RMSE	0.86	0.68	0.38

Table 2: RMSE results averaged for all patients

Gaussian Processes provide a well-founded background that derive their power more from the relationship (distance) of the data and less from prior

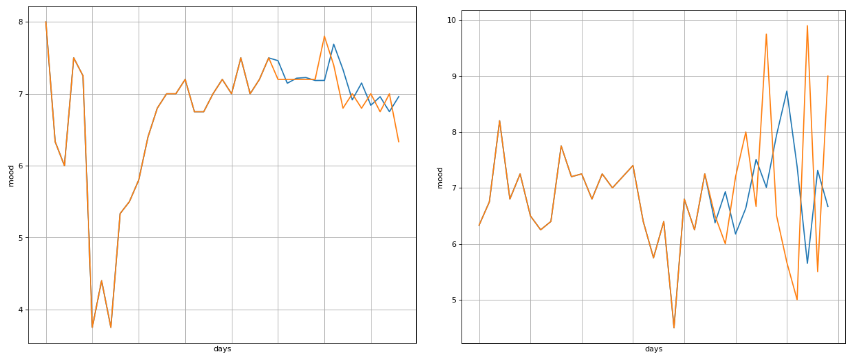


Fig. 5: Mood predictions (blue) and ground truth (orange) of the ARIMA model. On the left, the best predicted patient (0.1 RMSE on test), on the right the worst predicted patient (4.2 RMSE on test)

knowledge. We managed to find a distribution for every patient that describes his mood given either one feature (`circumplex.valence`) or all of them. It was a surprise that the model actually performed better using this one feature than all the other combined. After observing these results, it is therefore reasonable to consider collecting more data on this feature and related features to build even better GP models in the future.

ARIMA is a straightforward way of using a feature to predict future values for itself. In this case it outperformed the benchmark, but stays behind the more powerful gaussian processes. The main benefits of using ARIMA include the simplicity of implementation, intuitive understanding and reasonably good results.

4.1 Future Work

During the pre-processing stage we argued qualitatively in favour of some assumptions about the data, e.g. that aggregation is more meaningful as a sum rather than a mean or that outliers should be capped at a minimum/maximum value according to the Tukey rule. A deeper analysis is needed to confirm these assumptions quantitatively as well. Other conceivable approaches include replacing outliers by randomly sampling from the underlying distribution of the patient or setting them to the mean.

We have also examined the temporal data using both a time-series learning method as well as a supervised machine learning algorithm. There are plenty of more promising techniques that might be worth testing to achieve an even lower RMSE, such as RNNs or Random Forests.

References

1. Tukey, John W.: Exploratory Data Analysis. AddisonWesley Publishing Company Reading (1977)
2. C. E. Rasmussen C. K. I. Williams: Gaussian Processes for Machine Learning. the MIT Press, (2006)