Modeling Temporal Evolution in the Endomondo Fitness Data Set

Yashodhan Hemant Karandikar

Advisor : Prof. Julian McAuley

ABSTRACT

The goal of this project is to model temporal evolution of runners' physical capacities *across* workouts as well as *within* each workout. We study this temporal evolution over 2 predictive tasks - *a*) prediction of duration and average-heart rate given the distance of the next workout and given data of previous workouts of the runner; and *b*) prediction of the next instantaneous heart-rate values given the next instantaneous distance values and given previous instantaneous heart-rate and distance values within the workout. We evaluate our models on the Endomondo fitness data set [5]. Results indicate that accounting for temporal evolution within each workout significantly improves accuracy of predictions of instantaneous heart-rate values.

1. INTRODUCTION

People work out with the goal of becoming fitter. Regularly working out enables a person to take on more challenging workouts. For example, a runner training for long distance running can gradually attempt runs of longer and longer durations. Thus, the running capacity or the fitness level of a runner increases gradually with practice. Modeling this fitness level can be useful in predicting the duration of the next workout that the runner attempts.

Alternatively, runners often monitor their heart-rate while running and set goals of keeping the heart-rate below or above a certain value, either for improving their fitness level or for their safety. The heart-rate at future instants in the run depends on how tired the runner currently is, among other factors. Further, this relationship between the heart-rate and how tired the runner is, varies from one runner to another. For example, an experienced runner might experience only a slight increase in heart-rate as he or she feels more tired, while the heart-rate of an amateur runner might increase significantly soon after the run starts.

The examples above highlight the temporal evolution of a runner's capacity, both across workouts and within each workout. We attempt to model both these forms of temporal evolution in this work. Modeling this temporal evolution has several potential applications. Predictions of how long the next run will take or the average and maximum heart-rates in the next run will be useful both as a goal and as a guide to the runner while planning the run. Similarly, predicting how the heart-rate will change in the remaining part of a run will be useful for a runner who is monitoring heart-rate during the run. This can potentially be extended to suggest the runner a route for a run, given some parameters such as the desired average or maximum heart-rate, desired total duration etc.

The rest of the report is organized as follows. Section 2 describes the predictive tasks we use to evaluate our models. Section 3 describes some of the work in the literature which is relevant to temporal evolution of user expertise and to these tasks. Section 4 describes the Endomondo data set [5]. Sections 5, 6 and 7 describe our models and algorithms for training and making predictions. Section 8 discusses some implementation details. Section 9 discusses results of our models on the predictive tasks described in section 2. Finally, section 10 concludes.

2. PREDICTIVE TASKS

In order to study temporal evolution of a runner *across* several workouts, we choose the following 2 predictive tasks:

1. Given the distance d_i and duration T_i for each workout i for the first n workouts of a runner u and given the distance d_{n+1} for the $(n+1)$'th workout, predict the duration T_{n+1} of the $(n+1)$'th workout.
2. Given the distance d_i and average heart-rate H_i for each workout i for the first n workouts of a runner u and given the distance d_{n+1} for the $(n+1)$'th workout, predict the average heart-rate H_{n+1} of the $(n+1)$'th workout.

In order to study temporal evolution of a runner *within each* workout, we choose the following 2 predictive tasks:

1. Given the first n instantaneous distance and heart-rate values in a run and given the $(n+1)$'th instantaneous distance value, predict the $(n+1)$'th instantaneous heart-rate value within the same run.
2. Given the instantaneous distance and heart-rate values for the first f fraction of a run and given the instantaneous distance values for the remaining $1-f$ fraction of the run, predict the instantaneous heart-rate values for the remaining part of the run. This task is a generalization of the first.

3. RELATED WORK

[14] describes 4 different models accounting for temporal evolution of user expertise through online reviews on websites such as Amazon, BeerAdvocate, RateBeer, CellarTracker. [14] describes these models along 2 orthogonal directions: individual / community evolution and evolution at uniform / learned intervals, for a total of 4 models. Of these, the model for individual evolution at learned intervals is the most general. Runners improve their physical capacities for running at very different rates and independently of other runners, unlike general evolving trends in entire communities of users. Hence, the model for individual evolution at learned intervals makes most sense in the context of temporal evolution of runners' physical capacities for running. The work in this project is based on this model.

[12, 16] describe temporal collaborative filtering models for recommendation systems which account for deviations in parameters with time. These models can be used to characterize how runners' fitness levels evolve over time. Models for time-series analysis and prediction, studied extensively in the literature [8, 13, 1], are relevant to the task of predicting the next instantaneous heart-rate values in the remaining part of a run given a fixed number of previous instantaneous heart-rate values up to the current time instant.

4. ENDOMONDO FITNESS DATA SET

In this section, we introduce the Endomondo fitness data set [5]. Endomondo [5] is a free app and website that allows users to keep track of their workouts. The data extracted from the Endomondo website [5] is available in the form of a SQL dump of size 200 GB approximately after compression. It contains data for around 10 million workouts. Data for each workout is in the form of a HTML document string. This HTML document contains information about the workout such as the sport (such as running, walking, cycling etc.), workout id, user id, total distance, duration, wind, weather, hydration, date & time etc. The HTML document also contains embedded JSON data which includes GPS co-ordinates (latitude, longitude and altitude), distance, duration, heart-rate, speed / pace sampled throughout the duration of the workout. The 'duration' trace indicates the time instants at which the remaining attributes were sampled. We developed a parser in Python to extract the relevant data from each HTML document, details of which are discussed in section 8.

4.1 Data set statistics

We list some basic statistics of the data set. Table 1 lists the total number of workouts, the total number of users in the data set and the average number of workouts per user.

Total number of workouts	10320194
Total number of users	2083006
Avg. number of workouts per user	4.95

Table 1: Number of workouts and users in the data set

Although the average number of workouts per user is only 4.95, the distribution of workouts among users is skewed, so that there are many users with more than 10 workouts. Table 2 lists number of users with more than 10, 40, 70, 100 workouts.

n	**Number of users with more than n workouts**
10	271004
40	11679
70	2191
100	768

Table 2: Number of users having more than 10, 40, 70, 100 workouts

Table 3 lists the number of workouts for some of the activities (such as running, walking, cycling etc). The number of workouts for the *running* activity is the highest and it is significantly more than that for other activities. Hence, in this work we use data from workouts for the *running* activity only.

Activity	**Number of Workouts**	**Number of Users**
Running	4413262	1187925
Walking	2154563	558436
Cycling, sport	938108	260828
Cycling, transport	816808	216750
Mountain biking	429906	130603
Step counter	265403	13353
Fitness walking	172440	47775
Hiking	147061	57235

Table 3: Number of workouts and users by activity

5. BASELINE MODEL

This section describes a baseline model to predict the duration of a workout given the distance. Given the total distance d_{ui} for the i'th workout of runner u, the predicted duration T'_{ui} of the workout is given by:

$$T'_{ui} = (\alpha + \alpha_u)(\theta_0 + \theta_1 d_{ui}) \tag{1}$$

We have one parameter α_u per runner u. The parameters $\alpha, \theta_0, \theta_1$ are global to all runners. Thus, we have a total of $U + 3$ parameters, where U is the number of distinct runners in the data set. These parameters are learned during training.

Let $\Theta = (\alpha, \alpha_1, ..., \alpha_U, \theta_0, \theta_1)$. We need to find Θ which satisfies the following objective:

$$\hat{\Theta} = \arg\min_{\Theta} \frac{1}{|D|} \sum_{T_{ui} \in D} (T'_{ui} - T_{ui})^2 + \lambda\Omega(\Theta)$$

where D is the data set of all workouts for all runners, T_{ui} is the true distance of workout i of runner u. The term $\lambda\Omega(\Theta)$ is a regularizer, where $\Omega(\Theta)$ is given by

$$\Omega(\Theta) = \alpha^2 + \theta_1^2 + \sum_u \alpha_u^2$$

λ is the regularization hyper-parameter, which we select by performing a line search over values in the set $\{0, 10^{-5}, 10^{-4}, ..., 10^{3}, 10^{4}\}$ and choosing the λ which minimizes the error on a validation set. We find the Θ which satisfies the objective using the L-BFGS [15] implementation available in SciPy [10].

If the task is to predict the average heart-rate instead of the duration, we use the same model specification as in equation 1 by replacing T'_{ui} by H'_{ui}. Thus, the predicted average heart-rate is given by

$$H'_{ui} = (\alpha + \alpha_u)(\theta_0 + \theta_1 d_{ui})$$

6. TEMPORAL EVOLUTION ACROSS WORKOUTS

This section generalizes the baseline model described in section 5 and attempts to account for temporal evolution of runners across workouts.

6.1 Model Specification

In order to account for evolution in the fitness level or capability of a runner over time, we associate an *experience level* or *fitness level* e_{uw} with each workout w of runner u. This can be seen as a way of encoding how experienced the runner u is at the time of the workout w. The value e_{uw} is an integer in the interval $[1, E]$ where E is the number of experience levels.

Intuitively, we expect the experience level of a runner to either stay the same or increase with each workout. We encode this intuition in the form of a monotonicity constraint on the experience levels of workouts for each runner, as given below [14]:

$$\forall u, i, j \quad t_{ui} \geq t_{uj} \implies e_{ui} \geq e_{uj}$$

where t_{ui} is the time at which workout i of runner u occurs.

Then, given the total distance d_{ui} for the i'th workout of runner u, the predicted duration T'_{ui} of the workout is given by:

$$T'_{ui} = (\alpha_{e_{ui}} + \alpha_{ue_{ui}})(\theta_0 + \theta_1 d_{ui}) \tag{2}$$

where e_{ui} is the experience level of the runner u at the i'th workout. Thus, we have one parameter α_{ue} per runner u per experience level e. Further, we have an intercept term α_e for every experience level e, common to all runners. The terms θ_0 and θ_1 are global to all runners and experience levels. Thus, given U runners and E experience levels, we have a total of $UE + E + 2$ parameters.

Note that setting $E = 1$ reduces the model to the baseline model described in section 5.

We can specify a similar model for the predicted average heart rate H'_{ui} of the runner u during the workout i:

$$H'_{ui} = (\alpha_{e_{ui}} + \alpha_{ue_{ui}})(\theta_0 + \theta_1 d_{ui}) \tag{3}$$

6.2 Objective Function

As explained above, we have a different model for each of the E experience levels. Each of the E models has the following parameters:

$$\Theta_e = (\alpha_e; \alpha_{1e} ... \alpha_{Ue})$$

where $1 \leq e \leq E$ is the experience level and U is the number of distinct runners.

Thus, all the parameters in all of the E models together can be written as:

$$\Theta = (\Theta_1, \Theta_2, ..., \Theta_E, \theta_0, \theta_1)$$

Let β denote the set of all experience levels for all workouts of all runners. Then, we need to choose the optimal parameters $(\hat{\Theta}, \hat{\beta})$ according to the objective

$$(\hat{\Theta}, \hat{\beta}) = \arg\min_{\Theta, \beta} \frac{1}{|D|} \sum_{T_{ui} \in D} (T'_{ui} - T_{ui})^2 + \lambda_1 \Omega_1(\Theta) + \lambda_2 \Omega_2(\Theta, \theta_1) \tag{4}$$

$$s.t. \ \forall u, i, j \quad t_{ui} \geq t_{uj} \implies e_{ui} \geq e_{uj}$$

where D is the data set of all workouts for all runners. This objective is similar to the one described in [14]. Ω_1 and Ω_2 are regularizers defined as follows:

$$\Omega_1(\Theta) = \sum_{e=1}^{E-1} \|\Theta_e - \Theta_{e+1}\|_2^2$$

and

$$\Omega_2(\Theta, \theta_1) = \theta_1^2 + \sum_{e=1}^{E} \|\Theta_e\|_2^2$$

Ω_1 is a smoothness function which penalizes abrupt changes between successive experience levels [14], since in practice similar experience levels should have similar parameters [14]. Ω_2 is another regularizer which penalizes complex models i.e. it penalizes models which have higher magnitudes of parameters. These regularizers are necessary to avoid overfitting, since we have a large number $(UE + E + 2)$ of parameters. λ_1 and λ_2 are regularization hyperparameters, which tradeoff the importance of regularization versus prediction accuracy at training time [14]. We select λ_1 and λ_2 through a

grid-search over values in the set $\{0.0, 10^{-3}, ..., 10^4, 10^5\}$ and select the values which yield the highest prediction accuracy on a validation set.

For the task of predicting the average heart rate instead of the duration of the workout, the objective function remains the same as in equation 4, except that the term $T'_{ui} - T_{ui}$ is replaced by $H'_{ui} - H_{ui}$.

6.3 Training the model

We optimize the parameters Θ and β using a co-ordinate descent [3] i.e. we alternately optimize equation 4 for Θ given β and β given Θ. We optimize the model parameters Θ given β using L-BFGS [15] available in the LibLBFGS library [6]. Optimizing β given Θ means assigning a experience level to each workout so that the mean-squared-error is minimized, subject to the monotonicity constraint [14]. Since the experience levels are discrete, this problem is related to the *Longest Common Subsequence* problem [9] and can be solved efficiently using dynamic programming. The algorithm is identical to the one described in [14].

We alternately optimize Θ given β and β given Θ until β does not change or a maximum number of iterations has been reached.

6.4 Making Predictions

Once the model parameters and experience levels have been learned, we predict the duration or average heart rate for the next workout w for each runner u using equation 2 or 3. Note that we need to estimate the experience level for workout w in order to be able to use equation 2 or 3 to predict the duration or average heart-rate. We estimate this experience level for workout w and make predictions in 2 modes, namely *final* and *random*, as is done in [14].

In the *final* mode, we use the first $N-2$ workouts of each runner u as the training set, the $N-1$'th workout as the validation set and the N'th workout as the test set, where N is the total number of workouts for the runner u in the data set. We assume $e_{u(N)} = e_{u(N-1)} = e_{u(N-2)}$ and use these experience levels to predict the duration (using equation 2) or average heart rate (using equation 3) for the examples in the validation and test sets. The *final* mode is the setting that would be used in practice, since such a system would be used in predicting the duration or average heart-rate of the next workout of a runner, instead of making predictions about past workouts [14].

However, this mode of selecting workouts makes our validation and test sets biased towards workouts of more experienced runners [14]. To avoid this, we evaluate our models in the *random* mode as well. In the *random* mode, we randomly select 1 workout as the validation set and 1 workout as the test set, for each runner. The experience levels for these workouts is assumed to be equal to those of the workouts closest in time in the training set for that runner.

We present results for both modes in section 9.

7. TEMPORAL EVOLUTION WITHIN WORKOUTS

We now consider temporal evolution of a runner's capacity *during* a run and use it to predict the runner's instantaneous heart rate values in the remaining part of the run.

7.1 Model Specification

Given the heart-rate at various instants in a workout, we associate a *tiredness level* e_{wi} with heart-rate sample i in the workout w. As before, the value e_{wi} is an integer in the interval $[1, E]$, where E is the number of tiredness levels. We expect the tiredness level of the user to either stay the same or increase with each sample in the workout. We encode this in the form of a monotonicity constraint as before:

$$\forall w, i, j \quad t_{wi} \geq t_{wj} \implies e_{wi} \geq e_{wj}$$

where t_{wi} is the time at which sample i in workout w was sampled.

Then, given the distance d_{wt} covered up to sample t in the workout w, the predicted instantaneous heart-rate h'_{wt} at sample t in workout w is given by:

$$h'_{wt} = (\alpha_{e_{wt}} + \alpha_{we_{wt}})(\theta_0 + \theta_1 d_{wt}) \tag{5}$$

where e_{wt} is the tiredness level at the t'th sample in workout w. Thus, as before, we have one parameter α_{we} per workout w per tiredness level e and an intercept term α_e for every tiredness level e, common to all workouts. The terms θ_0 and θ_1 are global to all workouts and tiredness levels. Thus, given W workouts and E tiredness levels, we have a total of $WE + E + 2$ parameters.

7.2 Objective Function

As before, we have a different model for each of the E tiredness levels. Each of the E models has the following parameters:

$$\Theta_e = (\alpha_e; \alpha_{1e} ... \alpha_{We})$$

where $1 \leq e \leq E$ is the tiredness level and W is the number of workouts in the data set.

Thus, all the parameters in all of the E models together can be written as:

$$\Theta = (\Theta_1, \Theta_2, ..., \Theta_E, \theta_0, \theta_1)$$

Let β denote the set of tiredness levels for all samples in all workouts. Then, we need to choose the optimal parameters $(\hat{\Theta}, \hat{\beta})$ according to the objective

$$(\hat{\Theta}, \hat{\beta}) = \arg\min_{\Theta, \beta} \frac{1}{|D|} \sum_{h_{wt} \in D} (h'_{wt} - h_{wt})^2 + \lambda_1 \Omega_1(\Theta) + \lambda_2 \Omega_2(\Theta, \theta_1) \tag{6}$$

$$s.t.\ \forall u, i, j \quad t_{wi} \geq t_{wj} \implies e_{wi} \geq e_{wj}$$

where D is the data set of all samples in all workouts. This objective is similar to the one described in [14].

As before, Ω_1 and Ω_2 are regularizers defined as follows:

$$\Omega_1(\Theta) = \sum_{e=1}^{E-1} \|\Theta_e - \Theta_{e+1}\|_2^2$$

$$\Omega_2(\Theta, \theta_1) = \theta_1^2 + \sum_{e=1}^{E} \|\Theta_e\|_2^2$$

Ω_1 is a smoothness function which penalizes abrupt changes between successive tiredness levels [14], since in practice similar tiredness levels should have similar parameters [14]. Ω_2 is another regularizer which penalizes complex models i.e. it penalizes models which have higher magnitudes of parameters. These regularizers are necessary to avoid overfitting, since we have a large number ($WE+E+2$) of parameters. λ_1 and λ_2 are regularization hyperparameters, which trade-off the importance of regularization versus prediction accuracy at training time [14]. We select λ_1 and λ_2 through a grid-search over values in the set $\{0.0, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ and select the values which yield the highest prediction accuracy on a validation set.

7.3 Training the model

The training algorithm is identical to the training algorithm described in section 6.

7.4 Making Predictions

Once the model parameters and tiredness levels have been learned, we predict the instantaneous heart rate at 1 sample in each workout w using equation 5. As before, we need to estimate the tiredness level at that sample in order to be able to use equation 5 to predict the instantaneous heart-rate.

As in section 6, we make predictions in 2 modes, namely *final* and *random*. In the *final* mode, we use the first $N-2$ samples from each workout as the training set, the $N-1$'th sample from each workout as the validation set and the N'th sample as the test set, where N is the total number of samples in a workout w. We assume $e_{wN} = e_{w(N-1)} = e_{w(N-2)}$ and use these tiredness levels to predict the instantaneous heart-rate using equation 5.

In the *random* mode, we randomly select 1 sample from each workout as the validation set and 1 sample from each workout as the test set. The tiredness levels for these samples is assumed to be equal to those of the samples closest in time in the training set.

We present results for both modes in section 9.

7.5 Prediction of several instantaneous heart rate values

We now generalize the problem of predicting 1 instantaneous heart-rate value to the problem of predicting the heart-rate values for the last f fraction of the workout. We use the same model as described earlier in this section. Let e_{wn} be the tiredness level of the last sample in workout w in the training set. In order to predict the instantaneous heart-rate values in the remaining part of the workout w, we assume $e_{wn} = e_{w(n+1)} = e_{w(n+2)} = ... = e_{w(N)}$ where N is the total number of samples in workout w. In other words, we assume that the last tiredness level of the workout fitted during training stays the same for the remaining part of the workout. In practice, this is not a valid assumption, but this provides us with a conservative estimate of how well the model performs on the remaining part of the workout i.e. on the validation and test sets.

Note that the difficulty here is not only that of estimating the tiredness levels for the remaining part of the workout accurately, but also not having learned model parameters for tiredness levels beyond the last tiredness level fitted during training. Thus, even if the true tiredness levels for the remaining part of the workout are known, these cannot be used for prediction of heart-rate values.

8. IMPLEMENTATION DETAILS

8.1 Extracting relevant data

As mentioned in section 4, the data extracted from the Endomondo website [5] is available in the form of a SQL dump of size 200 GB approximately after compression. It contains data for around 10 million workouts. Data for each workout is in the form of a HTML document string. We developed a parser in Python to extract the relevant data from each HTML document. We use parallelization to speed-up the extraction of the data. Each line of the SQL dump contains data for multiple workouts in the form of HTML documents. A 'master' process reads each line, extracts the HTML documents and hands them over to 'worker' processes. Each 'worker' process takes 1 HTML document at a time, parses it to extract the relevant information about the workout and encodes it into a dictionary in JSON format. The worker then queues this workout data into a global queue. A separate 'writer' process dequeues workout information from the global queue and writes it to a single file in the form of 1 line per workout. Having a separate writer process to write the workouts extracted by worker processes eliminates disk contention among worker processes. It also allows data for all workouts to be written to a single large file instead of millions of small files, without any need for synchronization between processes since only a single process is performing the writes. All of the extracted data is written to a file of size around 30 GB after compression.

8.2 Performance optimizations

We use Python for most of the data processing tasks, with NumPy [7] and SciPy [10] libraries. We observe that training is computationally the most intensive part of the execution. We first speed-up the training using Cython [4]. This gives considerable speed-ups and scales well for the tasks of predicting duration and average heart-rate for the next workout of a user. However, even with this optimization, training still remained prohibitively expensive for the large data sets involved in the task of predicting instantaneous heart-rate

values (see section 9 for details of the data sets). Hence, we port the training part of these tasks to C++. This gives much higher speed-ups and enables a single run to complete in less than 25 minutes on a personal laptop machine.

8.3 Verifying Correctness of Derivatives

We verify that our implementations of derivative functions are correct by comparing their outputs to those obtained by computing numerical derivatives. We use functions provided in SciPy[10] and Dlib[11] to compute numerical derivatives.

9. EVALUATION

This section describes the results of the models described in sections 6 and 7 for the predictive tasks described in section 2. For all the tasks, we consider only workouts of the sport 'running' in the dataset.

9.1 Temporal evolution across workouts

This section discusses results of the model described in section 6 for the tasks of predicting the average heart-rate and total duration of a runner's next workout, given information about the runner's past workouts and the distance of the runner's next workout.

9.1.1 Prediction of average heart-rate

For this task, we consider only workouts of runners having more than 10 workouts. Further, we discard workouts with invalid date/time (i.e. a timestamp of 0), extremely long workouts (greater than 100 mi), extremely short workouts (lesser than 0.1 mi), workouts with extremely high average heart-rate (greater than 500 bpm) and workouts with extremely low average heart-rate (lesser than 10 bpm). We also discard workouts which do not have either heart-rate information or distance information.

Tables 4 and 5 show results for prediction of average heart-rate, using the *final* and *random* modes of evaluation, as discussed in section 6. As discussed in section 6, we have 1 workout in the validation set and 1 workout in the test set for each runner in the data set, for both modes of evaluation. The results are shown in terms of the coefficient of determination, R^2 [2], which is a measure of the fraction of variance in the data explained by the model. Table 6 shows the optimal values of hyper-parameters found using grid search.

As seen from the results, the model which accounts for temporal evolution of runners does not perform better than the baseline model ($E = 1$). A closer look at the hyper-parameters resulting from the grid search, shown in table 6, gives us some insight into this result. As is seen from table 6, grid search yields high optimal values of λ_1. Recall from section 6 that λ_1 is a hyper-parameter which penalizes drastic changes in parameters of similar experience levels. A high value of λ_1 means that the experience levels are very similar to each other and this essentially reduces the models with $E = 2$ and $E = 3$ to the baseline model $E = 1$.

9.1.2 Prediction of duration

For this task, we consider only workouts of runners having more than 10 workouts. Further, we discard workouts with invalid date/time (i.e. a timestamp of 0), extremely long workouts (greater than 100 mi and/or 72 hours), extremely short workouts (lesser than 0.1 mi and/or 36 seconds). We also discard workouts which do not have either duration information or distance information.

	Training R^2	Validation R^2	Test R^2
# Examples	39545	2384	2384
Baseline	0.520913	0.422149	0.423354
$E = 2$	0.520928	0.422081	0.423318
$E = 3$	0.520927	0.422085	0.423312

Table 4: Prediction of avg. heart-rate - *Final* mode of evaluation

	Training R^2	Validation R^2	Test R^2
# Examples	39545	2384	2384
Baseline	0.517636	0.480985	0.430890
$E = 2$	0.517645	0.480964	0.430880
$E = 3$	0.517692	0.480968	0.430886

Table 5: Prediction of avg. heart-rate - *Random* mode of evaluation

Tables 7 and 8 show results for the prediction of the duration of the workout, using the *final* and *random* modes of evaluation, as discussed in section 6. As discussed in section 6, we have 1 workout in the validation set and 1 workout in the test set for each runner in the data set, for both modes of evaluation. The results are shown in terms of the coefficient of determination, R^2 [2]. Table 9 shows the optimal values of hyper-parameters found using grid search.

The model which accounts for temporal evolution of runners does not perform better than the baseline model ($E = 1$). Further, as is seen from table 9, grid search yields relatively high optimal values of λ_1, indicating that the experience levels are very similar to each other and this essentially reduces the models with $E = 2$ and $E = 3$ to the baseline model $E = 1$.

	Training R^2	Validation R^2	Test R^2
# Examples	743987	52109	52109
Baseline	0.780689	0.678433	0.706519
$E = 2$	0.775247	0.677833	0.714295
$E = 3$	0.782736	0.677098	0.718808

Table 7: Prediction of duration - *Final* mode of evaluation

	Training R^2	Validation R^2	Test R^2
# Examples	743987	52109	52109
Baseline	0.728423	0.626545	0.589249
$E = 2$	0.691232	0.627357	0.578873
$E = 3$	0.705088	0.634563	0.581045

Table 8: Prediction of duration - *Random* mode of evaluation

	λ_1	λ_2
$E = 2$ (final)	10^4	0.0
$E = 3$ (final)	10^5	0.0
$E = 2$ (random)	10^5	0.0
$E = 3$ (random)	10^4	0.0

Table 6: Optimal values of hyper-parameters λ_1 and λ_2 for prediction of average heart-rate

	λ_1	λ_2
$E = 2$ (final)	10^4	0.001
$E = 3$ (final)	10^2	0.001
$E = 2$ (random)	10	0.01
$E = 3$ (random)	10	0.01

Table 9: Optimal values of hyper-parameters λ_1 and λ_2 for prediction of duration of the workout

The lack of improvement in the prediction accuracy after accounting for temporal evolution across workouts seem to suggest that we need data for many more workouts per runner and over much longer periods of time in order to be able to notice a reasonably strong evolution of the runner's fitness level across workouts. Further, an increase in the runner's physical capacity might also be exhibited in the data through the fact that the runner was able to complete a workout of longer duration and not necessarily through a change in the relationship between the duration (or average heart-rate) and the distance. This kind of temporal evolution would not be visible as an increase in the R^2 score in the prediction of the duration of average heart-rate.

9.2 Temporal evolution within workouts

We now discuss results for temporal evolution *within* workouts. First, we discuss results for the task of predicting a single instantaneous heart-rate value in each workout, followed by prediction of several instantaneous heart-rate values in each workout.

9.2.1 *Prediction of a single heart-rate value*

For this task, we consider only workouts with at least 200 samples of instantaneous heart-rate values and distance values. Tables 10 and 11 show results for the task of predicting a single instantaneous heart-rate value in each workout, using the *final* and *random* modes of evaluation respectively. The results show that accounting for temporal evolution within each workout significantly improves the prediction accuracy over the baseline ($E = 1$), for both modes of evaluation.

	Training R^2	Validation R^2	Test R^2
# Examples	24347765	83423	83423
Baseline ($E = 1$)	0.608874	0.585329	0.560480
$E = 10$	0.887163	0.839256	0.799058
$E = 20$	0.944594	0.899215	0.853857

Table 10: Prediction of instantaneous heart-rate - *Final* mode of evaluation

	Training R^2	Validation R^2	Test R^2
# Examples	24347765	83423	83423
Baseline ($E = 1$)	0.613856	0.608207	0.615789
$E = 10$	0.907340	0.894736	0.897766
$E = 20$	0.944524	0.928297	0.928000

Table 11: Prediction of instantaneous heart-rate - *Random* mode of evaluation

We plot the average absolute error in the prediction of instantaneous heart-rate against the sample number within the part of the workout in the training set, averaged over all workouts. Figure 1 shows this plot for both $E = 20$ and the baseline $E = 1$. We observe that the average errors for $E = 20$ are significantly lesser than those for the baseline model ($E = 1$), as expected. Note that this plot uses only the training part of each workout.

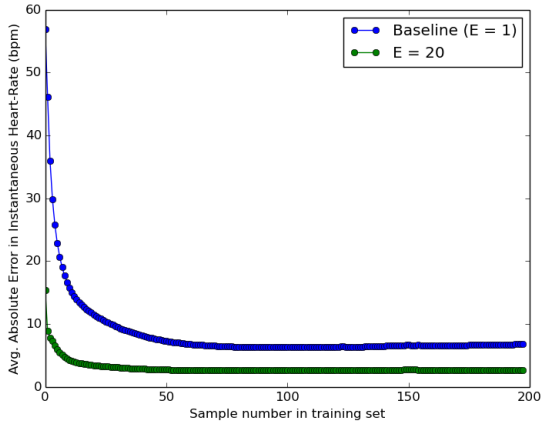

Figure 1: Average error in prediction of the instantaneous heart-rate for each sample number, averaged over all workouts in the training set.

Figure 2 shows how the prediction accuracy on the training and validation sets increases as we increase the number of tiredness levels E.

9.2.2 *Prediction of last several heart-rate values*

Runners often monitor their heart-rate while running and set goals of keeping the heart-rate below or above a certain value. Accordingly, a prediction of next several instantaneous heart-rate values would be more useful for the runner instead of a prediction of just the next 1 heart-rate value. This is one of the potential applications of this work. Hence, in this section, we evaluate how well our model predicts the heart-rate values in the remaining part of the workout, given samples of instantaneous heart-rate values and distance covered up to the current instant and given the samples of instantaneous distance values that will be covered in the remaining part of the workout.

For this task, we consider only workouts with at least 200 samples of instantaneous heart-rate values and distance values. Given a workout of $N \geq 200$ samples of instantaneous

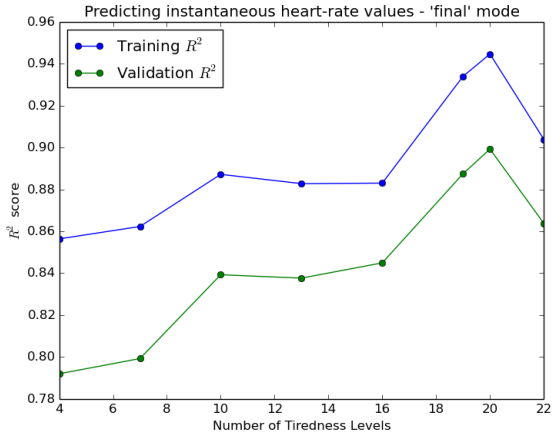

Figure 2: R^2 score on training and validation sets with varying number of tiredness levels.

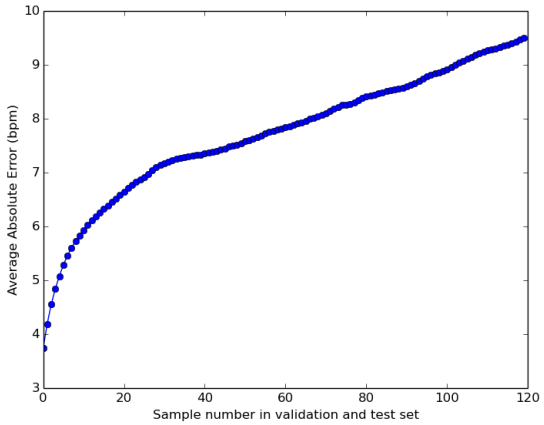

Figure 3: Average absolute error versus the sample number in validation and test part of the workouts, averaged over all workouts, for $E = 20$

heart-rate and distance values, we split N (distance, heart-rate) samples into training, validation and test sets of sizes $0.4N$, $0.3N$, $0.3N$ respectively. Table 12 shows results for the task of predicting last several instantaneous heart-rate values in each workout. The results indicate that accounting for temporal evolution significantly improves results compared to the baseline.

	Training R^2	Validation R^2	Test R^2
# Examples	9841317	7352229	7321065
Baseline ($E = 1$)	0.567424	0.451540	0.356603
$E = 10$	0.911437	0.632749	0.464435
$E = 20$	0.926065	0.633388	0.458077

Table 12: Prediction of several instantaneous heart-rates in each workout

We observe that the R^2 scores on the training set are much higher than those for the validation set, which in turn are much higher than those for the test set. This is expected due to the fact that the samples in the test set are further in time than the samples in the validation set, which in turn are further in time than the samples in the training set. Recall from section 7 that we assume $e_{wn} = e_{w(n+1)} = e_{w(n+2)} = ... = e_{w(N)}$ where N is the total number of samples in workout w and e_{wn} is the last sample in workout w in the training set. This assumption is lesser likely to hold for samples further and further in time. Thus, this assumption is less likely to hold for the test set than for the validation set. Hence, the R^2 scores on the test set are lower than on the validation set, which in turn are lower than on the training set. This can be seen from the plot shown in figure 3. The X axis is the sample number since the last sample in the training set. The Y axis is the absolute error averaged over all workouts. As expected, the average absolute error increases as the sample is further and further in time.

10. CONCLUSION AND FUTURE WORK

We describe and evaluate 2 different ways of accounting for temporal evolution of runners' physical capacities for running workouts - one *across* workouts and the other *within* workouts. Results indicate that accounting for temporal evolution across workouts does not improve prediction accuracy, perhaps due to the relatively small number of workouts per runner in the data set. On the other hand, accounting for temporal evolution *within* each workout significantly improves results over the baseline model.

Future work includes evaluating these models for other types of workouts, such as cycling, walking etc. The results on these types of workouts might give some interesting insights about the nature of these activities and about how users evolve with time in each of these activities. It might be interesting to repeat this work for a much larger data set with many workouts per user, since temporal evolution across workouts might be more prominent in such a data set. Models for time series prediction and analysis, as described in [1, 8, 13] can also be used to predict the instantaneous heart-rate values in the remaining workout.

11. REFERENCES

[1] Autoregressive Model. `http://en.wikipedia.org/wiki/Autoregressive_model`.

[2] Coefficient of Determination. `http://en.wikipedia.org/wiki/Coefficient_of_determination`.

[3] Coordinate Descent. `http://en.wikipedia.org/wiki/Coordinate_descent`.

[4] Cython - Working with Numpy. `http://docs.cython.org/src/tutorial/numpy.html`.

[5] Endomondo. `https://www.endomondo.com/`.

[6] libLBFGS - A library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). `http://www.chokkan.org/software/liblbfgs/`.

[7] Numpy. `http://www.numpy.org/`.

[8] R. Adhikari and R. Agrawal. *An Introductory Study on Time Series Modeling and Forecasting*. LAP Lambert Academic Publishing, Germany, 2013.

[9] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *SPIRE 2000 Proceedings of the Seventh International Symposium on String Processing Information Retrieval*, 2000.

[10] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[11] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[12] Y. Koren. Collaborative filtering with temporal dynamics. In *Communications of the ACM*, 2010.

[13] S. G. Makridakis. *A Survey of Time Series Issue 131 of INSEAD research papers series*. European Institute of Business Administration, 1975.

[14] J. J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *World Wide Web*, 2013.

[15] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35:773–782, 1980.

[16] L. Xiong, X. Chen, T. K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SIAM Data Mining*, 2010.