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' capacity across workouts as well as within each workout. We study this temporal evolution over 2 predictive tasks - prediction of duration of the next workout given previous workouts and prediction of average and instantaneous heart-rates. We show that accounting for temporal evolution improves results on these predictive tasks.

1. INTRODUCTION

People work out with the goal of becoming fitter. Regularly working out enables a person to take on more challenging work outs. For example, a runner training for long distance running can gradually attempt runs of longer durations. Thus, the running capacity or the fitness level of the person increases gradually with practice. Modeling this fitness level can be useful in order to be able to predict the duration of the next workout that the runner attempts. Such a prediction can be useful both as a goal and as a guide to the runner while planning the run.

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 own safety. The heart-rate at future instants in the run depends on how tired the runner is currently, 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 is more tired, while the heart-rate of an amateur runner might shoot up 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.

2. PREDICTIVE TASKS

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

1. Given the distance d_i and duration T_i for each workout i among first n workouts of a user 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 among first n workouts of a user 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 as follows:

1. Given the first n instantaneous heart-rates of a workout, predict the $(n+1)$'th instantaneous heart-rate.

2. Given the instantaneous heart-rates of the first f fraction of a workout, predict the heart-rates of the remaining $1-f$ fraction of the workout. This task is a generalization of the first.

3. RELATED WORK

[3] describes various models which account for temporal evolution of user expertise through online reviews on websites such as Amazon, BeerAdvocate, RateBeer, CellarTracker.

4. ENDOMONDO FITNESS DATA SET

In this section, we introduce the Endomondo fitness data set. [**?**].

5. BASELINE MODEL

This section describes a baseline model to predict the duration of a workout given the distance.

6. TEMPORAL EVOLUTION OF USERS

This section describes a model that attempts to account for temporal evolution of users across workouts. First, we describe the model and then describe the training algorithm.

6.1 Model Specification

In order to account for evolution in the fitness level or capability of a user over time, we associate a *experience level* or *fitness level* e with each workout w. This can be seen as a way of encoding how fit the user is at the time of the workout w. The value e is an integer in the interval $[0, E)$ where E is the number of experience levels.

Intuitively, we expect the experience level of a user 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 user, as given below [3]:

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

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

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

where e_{ui} is the experience level of the user u at the i'th workout. Thus, we have one parameter α_{ue} per user u per experience level e. Further, we have an intercept term α_e for every experience level e, common to all users. The terms θ_0 and θ_1 are global to all users and experience levels. Thus, given U users 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.

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 users.

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

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

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

$$(\hat{\Theta}, \hat{\beta}) = \arg\min_{\Theta,\beta} \frac{1}{N} \sum_{u,i} (T'_{ui} - T_{ui})^2 + \lambda_1 \Omega_1(\Theta) + \lambda_2 \Omega_2(\Theta, \theta_1) \quad (1)$$

$$s.t. \quad t_{ui} \geq t_{uj} \implies e_{ui} \geq e_{uj}$$

where D is the data set of all workouts for all users. This objective is similar to the one described in [3]. 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 [3], since in practice similar experience levels should have similar parameters [3]. 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 trade-off the importance of regularization versus prediction accuracy at training time [3]. We select λ_1 and λ_2 through a grid-search over values in the set $\{0.0, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$ and select the values which yield the highest prediction accuracy on a validation set.

6.3 Training the model

We optimize the parameters Θ and β using a co-ordinate descent [1] i.e. we alternately optimize equation 1 for Θ given β and β given Θ. We optimize the model parameters Θ given β using L-BFGS [4] available in the SciPy library [2]. 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 [3]. Since the experience levels are discrete, this problem can be solved efficiently using dynamic programming, as is done in [3].

We alternately optimize Θ given β and β given Θ until β does not change.

7. TEMPORAL EVOLUTION OF WORKOUTS
8. EVALUATION
9. CONCLUSION AND FUTURE WORK
10. REFERENCES

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

	# Examples	Variance	Linear Predictor	Collaborative Filtering	Latent Factor	Mahout ALS
Training	640135	0.049970	0.029748 **(0.404688)**	0.023681 **(0.526092)**	0.020354 **(0.592680)**	**(0.586866)**
Validation	160033	0.049826	0.033429 **(0.329085)**	0.033488 **(0.327886)**	0.030545 **(0.386954)**	**(0.392062)**
Test	200041	0.049818	0.033776 **(0.322010)**	0.033779 **(0.321948)**	0.030765 **(0.382451)**	**(0.388799)**

Table 1: MSE and R^2 obtained using the 3 predictors discussed in this work and Mahout's ALS recommender on the MovieLens dataset. Values in boldface/brackets are R^2 values.

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

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

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