PREDICTING WEIGHT LOSS USING THE MYFITNESSPAL DATASET

Nick Comly
Department of Electrical Engineering
Stanford University
ncomly@stanford.edu

Marissa Lee
Department of Mechanical Engineering
Stanford University
mrlee1@stanford.edu

William Locke
Department of Computer Science
Stanford University
wlocke@stanford.edu

1 Motivation

39% of adults worldwide and over 65% of US adults are clinically overweight or diagnosed with obesity[1][2]. Overweight and obesity increase the risk for health issues including diabetes, hypertension, and osteoarthritis [3]. Half of US adults try to lose weight each year. Most often, they attempt to do so by exercising more and eating less [4], since negative net calorie intake is associated with weight loss. To aid in this process, many individuals use calorie tracking apps. MyFitnessPal, for example, is an online calorie counter used to track and work toward weight loss goals. Users can track calories through diet and workout logs, and they can provide weight information over time. In this project, we develop a model to predict percent weight loss over a month using user data from MyFitnessPal. Understanding the behaviors over time that lead to weight change is important in predicting how successful a new user might be in reaching healthier weights.

2 Related Work

Previous work with the MyFitnessPal dataset has been conducted by Gordon, Althoff, and Leskovec. Using a limited set of features from the first seven days of a user's activity, the authors predicted whether a user would ultimately achieve their goal with an area under the receiver operating characteristic of 79% [5].

Aside from the above work, research that applies machine learning to user nutrition logs is sparse due to the lack of availability of such datasets. However, machine learning has been widely applied to the domain of electronic health records (EHRs) [6][7]. Choi et al. apply neural networks to EHRs to predict heart failure [8], Christopoulou et al. use natural language processing with deep learning to report adverse drug events [9], and Miotto et al. built a system they dub "Deep Patient" that they trained on EHRs for around 700,000 patients to classify among 78 diseases [10].

Although nutrition log data is not strictly in the realm of EHR, data challenges we faced such as heterogeneity, sparsity, noise, time dependency, incompleteness, irregularity, and systematic biases match those commonly cited in the EHR literature [11][12][13][14]. Therefore, in dealing with these challenges we drew heavily from this domain. Decisions we made regarding outlier detection techniques [15], recurrent neural networks application to time-series data [16][17], and model architectures for combining static and temporal data [18] are all informed by the cited EHR work.

3 Dataset and Features

3.1 Overview

MyFitnessPal records a wide range of information describing user demographics, goals, weights, food consumption, and workouts. Each user has three static goals: 1) a goal weight, 2) a weekly weight loss goal, and 3) a daily net calorie goal. Weights can be recorded throughout the day so a user can track their progress toward their overall weight goal.

Food logs include detailed nutritional information including amounts of calories, macronutrients, vitamins, and minerals. Logging a food involves searching and selecting from a database of around 10 million foods or generating a custom food log. 86% of food logs are entered via search and select, with the remaining 14% entered manually. Workout logs record the type of workout along with metrics such as workout duration and an estimate for the number of calories burned. Metrics taken from fitness tracking devices, such as step count and average pace, can also be included.

Our full dataset includes MyFitnessPal user data from 2011 to 2017. This consists of 1.7M users, 99 million weight measurements, 169 million workouts, and 3.2 billion food logs.

3.2 Inclusion Criteria

To enhance the credibility of our data, we included only users with weights likely recorded from Bluetooth digital scales (rather than recorded manually). Although not explicitly distinguished in the dataset, we found a clear cluster separation between weights with one decimal point or less of precision, and those with long trailing decimal patterns (i.e. >6 decimal places). From this, we could infer and filter out manually-entered weighs. This resulted in a 20% error reduction in our baseline model.

Users with any two weight logs made 28 days apart were included. For example, if a user logged weights on January 1, January 29 (4 weeks later), they were included. A single user could produce multiple examples if they presented multiple (non-overlapping) instances of this pattern. 465,265 instances of this occurred with 222,115 unique users.

3.3 Weight Fluctuation

Taking the difference of weights entered on consecutive days, we found a 0.6% (or 1.0 lb) mean average absolute difference existed, supportive of existing literature that reports marked random and cyclical daily weight fluctuations[19]. We use this as a marker for the lower bound of our mean absolute prediction error.

3.4 Data Engineering

Due to the self-reported nature of this dataset and inconsistency in human behavior, challenges with the dataset include sparsity, credibility, heterogeneity, irregularities, and systematic biases. Consequently, a significant amount of data engineering was required in order to produce data of sufficient quality for our neural networks (4.3).

3.4.1 Outliers

Basic thresholding removed impossible examples (e.g. examples with negative nutritional features). In order to further de-noise our training data we fit a linear regression model using our baseline feature set and extracted the 5% least likely datapoints [15].

3.5 Features

Our static feature set included 53 features, while daily nutrition log aggregations, 16 for each of the 28 days, are provided to our RNN based architecture (totaling 448). Static features included user demographics, goals, and summary information (e.g. averages and totals) describing weights, food consumption, and workouts between Day 0 and the end of Week 4.

All features were normalized to have zero mean and unit variance. Features which were related (e.g. percent carbohydrates, percent protein, and percent fat in the month's diet) were normalized together to ensure an accurate representation of each user.

4 Methods

4.1 Linear Regression

Linear Regression was used for baselines and basic models. Linear Regression approximates true values, y, by using a linear combination of the inputs features, x: $\hat{y} = \theta^T x$. An iterative process is used to minimize the loss function $||y - \theta^T x||_2^2$ for all training examples, thus achieving closer approximate percent weight loss. Both traditional and Elastic Net Linear Regression was used. Elastic Net has a combination of L_1 and L_2 regularization, adding $\alpha\ell_1||\theta||_1^2 + \alpha(1-\ell_1)||\theta||_2^2$ to the loss function. This limits the size of θ, which decreases the likelihood of over-fitting.

4.2 Gradient Boosted Decision Trees

Decision Tree (DT) Learning uses a decision tree to make regression predictions on a dataset. DT is a weak learning algorithm, i.e. it can perform better than random, however, it is not anticipated to perform perfectly. A Gradient Boosting Machine (GBM) starts with a greedily-generated decision tree. Then it continues to create more trees with modified gradients in the loss function for the incorrect examples from the previous trees. This is an attempt to correctly predict the examples previous trees could not. Prediction is then conducted by using a weighted sum of all trees. GBM can perfectly predict a training set because it can continue to make more trees until all examples are correct. Of course, this does not generalize well, so tuning must be conducted carefully to avoid over-fitting.

4.3 Neural Networks

For our neural network architectures, we adopt two main variants; the first taking static (including aggregated) features, and the second taking static along with dynamic features in the form of daily food logs fed through a recurrent neural network (RNN). The static feature model is a four-layer neural network (including input and output layers) with two hidden layers of size 400 and rectified linear units [20] as the activation functions. The second model feeds static features through the above architecture while dynamic features are fed through a long short-term memory (LSTM) [21] component detailed in Section 4.4. The outputs of these two components are then combined and fed through a linear layer to produce the final output. A diagram of this model is shown in Figure 1.

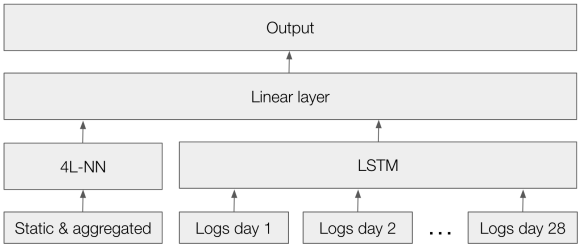

Figure 1: LSTM/4L-NN Model

4.4 RNNs

In their 2019 review of health informatics papers, Kwak et al. find RNNs (51%) to be the most common deep learning method used on EHRs, as compared to convolutional neural networks (20%), deep neural networks (12%), autoencoders (9%), restricted Boltzmann machines (4%), and deep belief networks (3%) [7]. The structure of RNNs makes them capable of ingesting time-series data in a way that can capture directional dependence. This makes them a good fit for EHR data, which is often inherently time-series based [16][17].

An issue with traditional or "vanilla" RNNs is their performance efficacy over large numbers of time steps, with empirical demonstrations showing that they cannot "remember" events past 5-10 steps into the past due to the vanishing and exploding gradients problem [22]. LSTMs on the other hand, have a gating mechanisms that enable them to make predictions based on longer sequences. While a vanilla RNN calculates the hidden state at each time-step as:

$$h_t = f(Ux_t + Wh_{t-1})$$

The gating mechanism of the LSTM allows errors to backpropagate through any number of time-steps. In the following equations, i and o represent the input and output gates, respectively. These regulate how the current time t input is combined with the forget gate f and previous hidden state h_{t-1} to produce the new hidden state h_t:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$h_t = o_t \circ \tanh(c_t)$$

For this reason, LSTMs (and similarly performing GRUs [23]) are the default RNN of choice in much of the EHR literature, and thus are the default choice in our implementation.

5 Experiments

To test the viability of the problem, baseline models were created and later used to evaluate further implementations. The first baseline model used linear regression to predict percent change in weight over four weeks just based on starting weight. The second baseline also used linear regression, this time with 16 basic features including user demographics, goals, and high-level food intake.

All models used two metrics: mean squared and absolute error applied to both a test and training set. We used a train/test split of 80/20, enforcing that the same user could not appear in both splits, but could have more than one (non-overlapping) interval within each set.

5.1 Linear Regression

Elastic Net had hyper-tuning of α and ℓ_1 over ranges of $[10^{-5}, 1]$ by factors of 10 and $[0, 1]$ by 0.1 sized steps respectively. Grid search with 5-fold cross validation was performed over these ranges with optimal model parameters found of $\{\alpha = 10{-}4, \ell_1 = 0.3\}$.

5.2 Gradient Boosted Decision Trees

GBM contains hyper-parameters specific to each tree, like depth and leaf size, as well as those specific to boosting, like number of estimators and loss function type. Both tree and boosting parameters were tuned. Grid search with 5-fold cross validation was performed. Optimal performance was found with $\alpha = 0.1$, learning rate $= 0.1$, least-squares loss, max depth $= 8$, min sample splits $= 500$, min samples per leaf $= 90$, 190 estimators, and 80% of samples used for fitting trees. All of the aforementioned parameters were swept over ranges containing between 5 and 25 values with optimal values found within the middle of the ranges, i.e. not the smallest or largest value. Tuning led to a 5% improvement in MSE over initial parameters.

5.3 Neural Networks

The initial phase of our neural network experimentation involved developing an optimal setup for tuning and training our NN architectures. We apply Z-score normalization to our data [24]. Our neural network implementations were built using PyTorch [25].

For training, we investigated a number of hyper-parameters as well as model parameters like hidden layer size. Our final four-layer NN (4L-NN) and LSTM model were trained using an Adam optimizer [26] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, a learning rate of $1\mathrm{e}{-4}$, and $L2$ weight decay [27] with a regularization rate of $1\mathrm{e}{-10}$. We used a mini-batch [28] size of 64 and batch shuffling [29] for our training phase. Training was performed on an NVIDIA K80 GPU, with each epoch taking around 1.9 seconds for the 4L-NN and 12.1 seconds for the LSTM model. Each model converged at around 650 epochs. Our 4L-NN model was trained using 53 features, and the LSTM model with 501 features (53 static and 448 dynamic, as detailed in Section 3.5). For comparison, we trained two linear regression models, one on our baseline feature set (16 features), and the other on the same 53 features used for our 4L-NN. The results of this training can be seen in Figures 2 and 3.

6 Results and Discussion

Our best models (Table 1) were able to predict most users to within a percentage point over a 4-week period, with the LSTM/4L-NN able to predict with an average error of 1.09% body weight, equal to 0.49% above daily weight fluctuation. Our biggest performance increases resulted from data engineering, which yielded a 38.7% error reduction. In comparison, modeling adjustments achieved a relatively low cumulative error reduction of 15.3%.

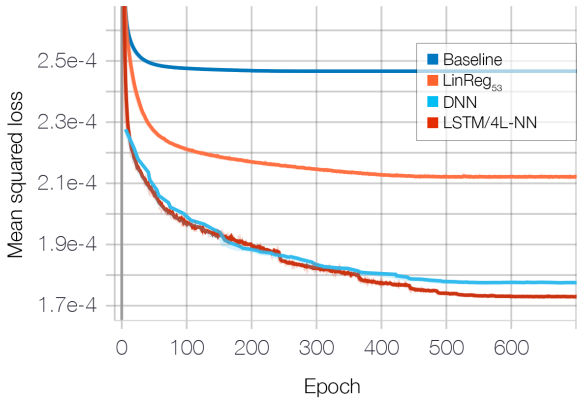

Figure 2: Train loss

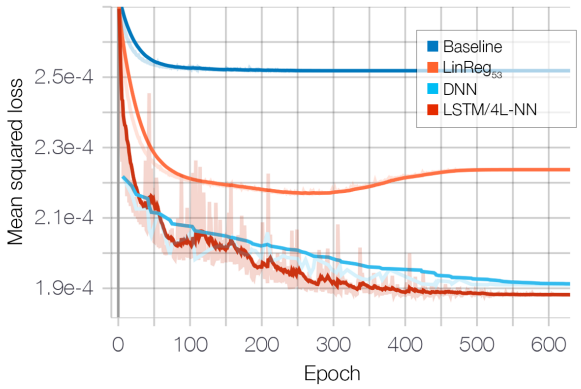

Figure 3: Validation loss

Table 1: Results

Model	Mean squared error % Train	Test	Mean absolute error% Train	Test
Baseline	1.23E+1	1.05E1	2.12	2.10
LinReg_{16}	2.46E-2	2.52E-2	1.27	1.29
LinReg	2.05E-2	2.07E-2	1.17	1.20
ElasticNet	2.07E-2	2.04E-2	1.18	1.16
GBM	1.55E-2	1.99E-2	1.01	1.14
4L-NN	1.78E-2	1.91E-2	1.08	1.10
LSTM/4L-NN	1.73E-2	1.88E-2	1.06	1.09

6.1 Error analysis

In order to perform qualitative analyses of our results we, connected our models to a user interface[1] where we could vary user attributes and nutrition variables. These analyses revealed systematic biases in the model predictions. Firstly, the models biased predictions towards weight loss, reflecting a data bias towards weight losers (62% of the training set). This was reinforced by our model scoring 30% better on weight losers.

Secondly, our linear models in particular predicted over-conservatively. For instance, increasing the daily caloric intake of a generic user to 5,000 calories only increased the predicted weight by a few pounds. Again, we found that the data was biased towards smaller weight changes, with only 16.7% of users weight changing by more than 5 lbs in the month. Therefore, models were able to score well by predicting conservatively.

Correcting for these biases resulted in our neural network models producing sensible results across a wide range of inputs, while the linear models, although improved, still lacked the expressiveness to produce common sense predictions for input ranges outside a narrow band of typical input.

7 Conclusion

While MyFitnessPal may be the largest nutritional dataset ever collected, leveraging this wealth of data for the benefit of user health requires significant processing in order to produce credible, de-biased training sets, as well as careful modeling in order to achieve predictions that can be trusted over a range of inputs. This paper forms a baseline for producing such predictions, with further work required in behavior modeling in order to produce nutritional advice that takes into account an individual's likelihood of sticking to a particular regimen over longer timespans.

[1] `https://mfp.ai`

Source Code

`https://github.com/williamlocke/cs229-project`

Contributions

All team members contributed to the overall approach. W.L. pre-processed the source dataset, built features, designed and trained neural networks, performed qualitative error analysis. M.L. built features and characterized the datasets. N.C. designed and trained linear regression and Gaussian Boosted Tree models and implemented feature selection.

References

[1] Cynthia L. Ogden, Margaret Deutsch Carroll, Lester R. Curtin, Margaret A. McDowell, Carolyn J Tabak, and Katherine M. Flegal. Prevalence of overweight and obesity in the united states, 1999-2004. *JAMA*, 295 13:1549–55, 2006.

[2] WHO. Obesity and overweight. *World Obesity Federation*, 2018.

[3] Cynthia Ogden. Overweight & obesity statistics. *National Institute of Diabetes and Digestive and Kidney Diseases*, 2017.

[4] Crescent B. Martin, Kirsten A. Herrick, Neda Sarafrazi, and Cynthia L. Ogden. Attempts to lose weight among adults in the united states, 2013-2016. *NCHS Data Brief*, 313, 2018.

[5] Mitchell L. Gordon, Tim Althoff, and Jure Leskovec. Goal-setting and achievement in activity tracking apps: A case study of myfitnesspal. pages 571–582, 2019.

[6] Benjamin Shickel, Patrick Tighe, Azra Bihorac, and Parisa Rashidi. Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. *IEEE J. Biomedical and Health Informatics*, 22(5):1589–1604, 2018.

[7] Gloria Hyun-Jung Kwak and Pan Hui. Deephealth: Deep learning for health informatics, 2019.

[8] Edward Choi, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. Medical concept representation learning from electronic health records and its application on heart failure prediction, 2016.

[9] Fenia Christopoulou, Thy Thy Tran, Sunil Kumar Sahu, Makoto Miwa, and Sophia Ananiadou. Adverse drug events and medication relation extraction in electronic health records with ensemble deep learning methods. *Journal of the American Medical Informatics Association*, 08 2019. ocz101.

[10] Riccardo Miotto, Li Li, and Brian Kidd. Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific Reports*, 6:26094, 05 2016.

[11] Yu Cheng, Fei Wang, Ping Zhang, and Jianying Hu. Risk prediction with electronic health records: A deep learning approach. pages 432–440, 06 2016.

[12] Peter Jensen, Lars Jensen, and Søren Brunak. Mining electronic health records: Towards better research applications and clinical care. *Nature reviews. Genetics*, 13:395–405, 05 2012.

[13] Nicole G. Weiskopf, George Hripcsak, Sushmita Swaminathan, and Chunhua Weng. Defining and measuring completeness of electronic health records for secondary use. *J. of Biomedical Informatics*, 46(5):830–836, October 2013.

[14] Nicole Weiskopf and Chunhua Weng. Methods and dimensions of electronic health record data quality assessment: Enabling reuse for clinical research. *Journal of the American Medical Informatics Association : JAMIA*, 20, 06 2012.

[15] Juliano Gaspar, Emanuel Catumbela, Bernardo Marques, and Alberto Freitas. A systematic review of outliers detection techniques in medical data - preliminary study. pages 575–582, 01 2011.

[16] Juan Zhao, Qiping Feng, Patrick Wu, Roxana Lupu, Russell Wilke, Quinn Wells, Joshua Denny, and Wei-Qi Wei. Learning from longitudinal data in electronic health record and genetic data to improve cardiovascular event prediction. *Scientific Reports*, 9, 01 2019.

[17] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks, 2015.

[18] Cristóbal Esteban, Oliver Staeck, Yinchong Yang, and Volker Tresp. Predicting clinical events by combining static and dynamic information using recurrent neural networks, 2016.

[19] Anna-Leena Vuorinen, Elina Mattila, Miikka Ermes, Mark Gils, Brian Wansink, and Ilkka Korhonen. Weight rhythms: Weight increases during weekends and decreases during weekdays. *Obesity facts*, 7:36–47, 01 2014.

[20] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[22] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318. JMLR.org, 2013.

[23] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[24] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, UK, 1998. Springer-Verlag.

[25] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[27] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, pages 950–957, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[28] D. Wilson and Tony Martinez. The general inefficiency of batch training for gradient descent learning. *Neural networks : the official journal of the International Neural Network Society*, 16:1429–51, 01 2004.

[29] Qi Meng, Wei Chen, Yue Wang, Zhi-Ming Ma, and Tie-Yan Liu. Convergence analysis of distributed stochastic gradient descent with shuffling, 2017.