Ethan Powell
11/6/18
Experiment #3: Glucose Prediction for T1D Patients using MLP's in Python

Summary

I trained two Multilayer Perceptrons with three months of continuous glucose monitor data to predict the glucose values of a type 1 diabetic. The model performances suggest some promise for the use of neural networks in glucose forecasting. However, more research is needed to optimize the model suggested herein.

Dataset and Preprocessing

The CGM data was taken directly from the current iteration of GluGo2.0's dataset. That dataset consists of roughly three months of pump and insulin data from a type 1 diabetic. The CGM data spans from June 20th to September 26th, and it consists of 18,493 glucose readings. Outliers (suggesting a misreading) and calibration values were culled prior to my use of the GluGo2.0 dataset. It should be noted that the dataset only includes data for times at which a CGM reading took place. Therefore, the dataset is marked by a few temporal gaps. The original dataset can be found through the following Github repository path:
/danny98m/GluGo2.0/tree/master/csvData/csvOutData

Yun Zhang's analysis of the GluGo2.0 glucose data reveals the strongly seasonal character of glucose trends among type 1 diabetics:

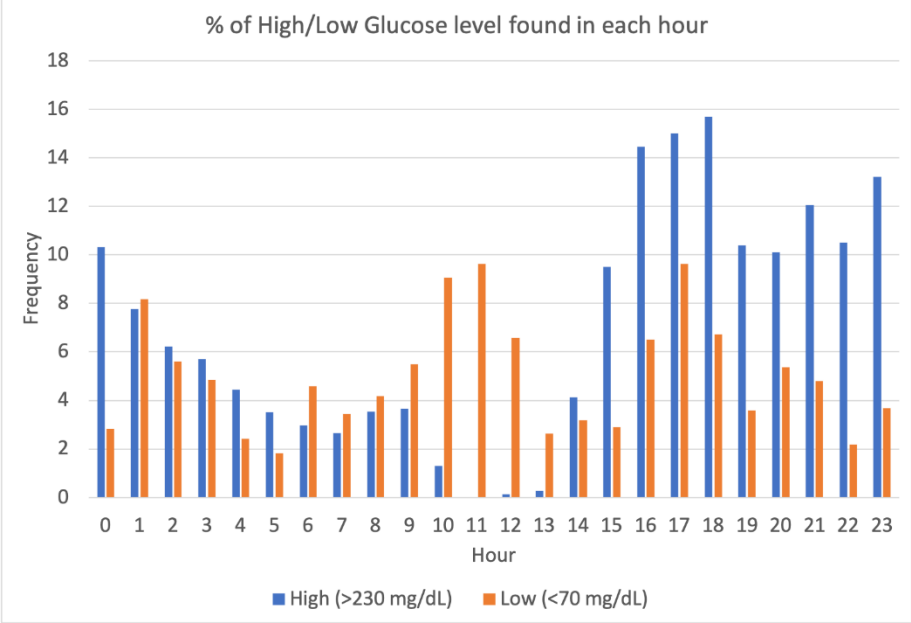

Fig. 1: For each hour of the day, the percentage of CGM readings spent high/low of all CGM readings taken. For each pair of boxplots, the left boxplot represents frequency of highs and the right boxplot represents frequency of lows

The bar plot suggests some correlation between the frequency of highs/lows and the time of day. It also suggests that the structure of the correlation is different for highs than it is for lows. Furthermore, my analysis of the broader CGM dataset of Experiment #1 suggests a correlation between glucose level and glucose slope:

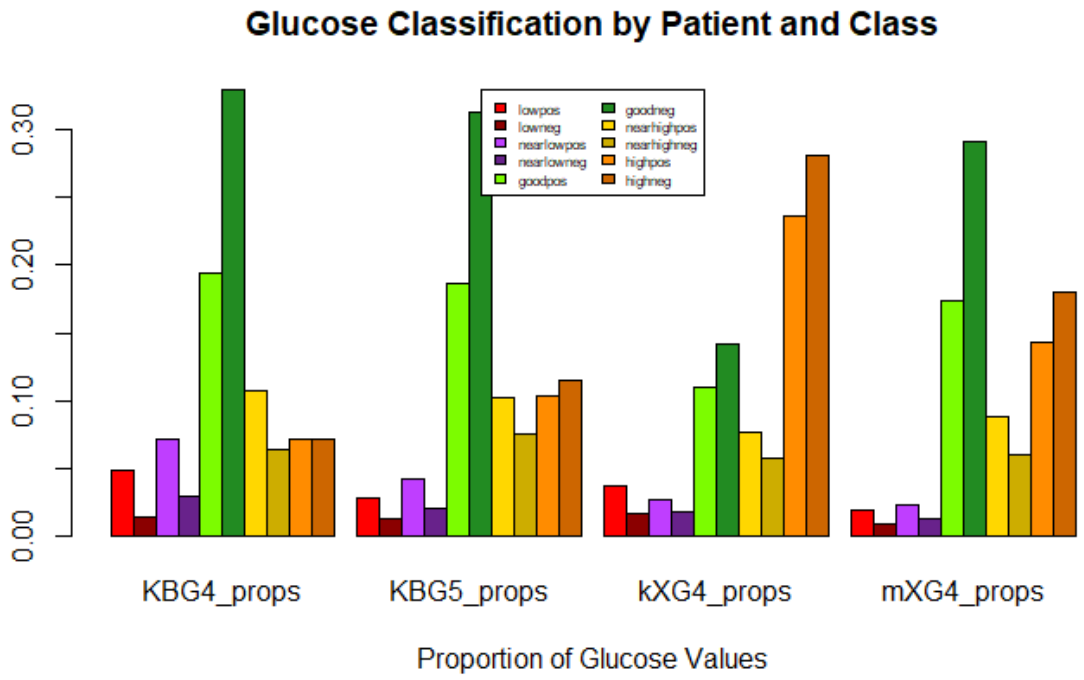

Fig. 2: Proportion of Glucose Values by patient and classification. Each classification denotes a glucose level and slope simultaneously. The prefix of each classifier (e.g. low – less than 70 mg/dL, high – greater than 270 mg/dL, good – in between) denotes the level, and the suffix (i.e. -neg, -pos) denotes the slope. Each set of bar plots represents a different patient.

For example, when glucose is neither high (>270 mg/dL) nor low (<70 mg/dL), all patients in the sample experience a greater frequency of declining glucose values than increasing glucose values. Lastly, the recent work of GluGo2.0 in fitting an ARIMA model to CGM data indicates that glucose values also have a mild correlation with values in their immediate vicinity (autocorrelation). These complex dependencies in glucose trends are the result of exogeneous biological indicators, like insulin sensitivity, and carbohydrate intake. Models built using biological indicators have been successful[1], and offer the potential for great explanatory insight into the mechanisms behind such dependencies. But neural networks are capable of automatically modeling complex dependencies[2], and thus might offer a means of predicting glucose trends without depending on biological markers, which are often impractical for a type 1 diabetic to obtain or update in real time.

Therefore, I chose to predict glucose values based on 1. the time of day, and 2. a set of glucose values k-lags in the past for -4 >= k >= -16. Each lag represents a jump to the previous consecutive glucose value. Glucose readings for this dataset were usually taken at five minute intervals. Therefore, the model simulates the prediction of a glucose value 20 minutes into the future using 1.

[1] Plis, K., Bunescu, R. C., Marling, C., Shubrook, J., & Schwartz, F. (2014, March). A machine learning approach to predicting blood glucose levels for diabetes management. In *AAAI Workshop: Modern Artificial Intelligence for Health Analytics* (No. 31, pp. 35-39).
[2] See *Deep Learning for Time Series Forecasting* by Jason Brownlee

glucose values from the past hour, and 2. the hour of day associated with the k=-1 glucose values. The 16 earliest glucose values in time were removed from the dataset because none of them have the full set of lagged glucose values as predictors. The final dataset can be found in ***glucose_dataset_-_lags,_hour_of_day.csv***.

The setup of the problem involves using continuous features to predict a continuous variable. Therefore, there were many regression algorithms to choose from. However, many regression algorithms assume that all predictors are independent of one another, which is not the case in our dataset. Neural nets don't make this assumption. Indeed, they leverage dependencies among the predictors to make better predictions. I hypothesized that a neural net was most appropriate because it would automatically learn the autocorrelational structure of the glucose trends from the lagged values, and daily seasonality from the hour of day.

Methods: Single Hidden Layer Perceptron with Python

A single hidden-layer perceptron was coded with the Sci Kit Learn and Karas modules in Python. Unless noted otherwise, all methods regarding the specification, training, and testing of the model were replicated from the following Jason Brownlee tutorial:
machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python

We did not include the results of the standardization performed in the second iteration of Brownlee's model, because they were worse than the results with non-standardized data.

Because the training process for neural nets is iterative[3], it involves dividing the training set into samples ("batches"), and resampling after all batches are seen (an "epoch" denotes one resampling). Guess-and-check was used to find the values of these hyperparameters that generated the optimal model.

The mean square error (MSE) was the performance metric used to determine the optimal model. The Kappa statistic was avoided because of concerns in the literature that it's inappropriate when the predicted variable is continuous[4]. k-fold cross validation was used to test each model. We found that for k>3, runtimes for model testing were prohibitively slow. Therefore, we used 3-fold cross validation for all models.

To compare our model against a baseline, we tested a "naïve" model, in which the glucose value for a given time is assumed to be the same as the glucose value of the k=-4 lag. In other words, it predicts that the glucose value 4 lags (20 minutes usually) into the future is the same as the current glucose value. Because the model isn't stochastic, we did not perform a k-fold cross validation, but simply calculated the MSE based on the complete dataset. Details of the calculation can be found in ***naive_prediction_MSE.xlsx***.

[3] ***towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9***

[4] Mandrekar, J. N. (2011). Measures of interrater agreement. *Journal of Thoracic Oncology*, *6*(1), 6-7.

Results: Single Hidden Layer Perceptron with Python

MSEs for selected Single Hidden Layer Perceptron Models

			Epoch #					
Batch Size		**10**	**15**	**20**	**25**	**35**	**45**	**55**
	N = 10	426.87	369.62	363.38		349.47	344.85*	347.45
	N = 20	378.36				349.69	346.22	352.53
	N = 35					357.27		

Fig.3: A matrix of model mean squared errors in which rows indicate the batch size of the model training, and columns indicate the epoch number of the model training. All values rounded to two decimal places. The minimum mean squared error is starred.

Performance Metrics for the Optimal Single Hidden Layer Perceptron Model and Naïve Model

	Optimal Perceptron Model	**Naive Model**
MSE	344.85 mg/dL^2	433.16 mg/dL^2
RMSE	18.57 mg/dL	20.81 mg/dL

Fig. 4: A matrix of performance metrics in which columns indicate the model, and rows indicate the performance metric. MSE is the mean squared error, and RMSE is the root mean squared error. All values rounded to two decimal places.

We found that the optimal perceptron model performed significantly better than the naïve model. The optimal perceptron's predictions reduced the degree of error by about 10.76%. The trial and error approach to model selection suggests some properties of the training hyperparameters: Increasing the epoch number while holding batch size constant seems to result in better scores than increasing both in a fixed ratio. Furthermore, there appears to be an upper bound in accuracy when increasing epoch number and holding batch size constant.

Methods: Double Hidden Layer Perceptron with Python

We respecified the Perceptron model to include a second hidden layer of neurons. In accordance with Brownlee's suggestion, we included half as many neurons in the second layer as input features (7 neurons). The model was trained with 45 epochs and a batch size of 20 (the specification which resulted in the optimal single-layer model). All other methods remained unchanged from the single hidden layer model procedure. The code for both models can be found in ***Keras_NN.py***.

Results: Double Hidden Layer Perceptron with Python

Performance Metrics for the Optimal Single Hidden Layer Perceptron Model and the Double Hidden Layer Perceptron Model

	Optimal Single Layer Perceptron Model	**Double Layer Perceptron Model**
MSE	344.85 mg/dL^2	364.74 mg/dL^2
RMSE	18.57 mg/dL	19.10 mg/dL

Fig. 5: A matrix of performance metrics in which columns indicate the model, and rows indicate the performance metric. MSE is the mean squared error, and RMSE is the root mean squared error. All values rounded to two decimal places.

We found that the double layer perceptron model performed worse than the optimal single layer perceptron model. The double layer perceptron's predictions reduced the degree of error from the naïve model by only 8.22%, compared to the 10.76% reduction offered by the optimal single-layer model.

Discussion

Our results suggest that neural nets generally surpass naïve models. Much more research is required to verify my hypothesis regarding the suitability of deep learning for time series forecasting, and to assess the true degree to which neural nets can surpass other models. Research is needed on many fronts: greater background in the theory and methodology of neural nets is needed in order to 1. properly specify the hyperparameters of the training process, 2. to specify the appropriate width and depth of the neural net, and 3. to specify the optimal number of folds to use in cross-validation. Furthermore, greater background in the application of neural nets to time series analysis is needed to specify an appropriate dataset. More research into other models is needed to establish a more robust baseline to test against. Greater familiarity with the Sci Kit Learn and Karas modules is needed in order to generate detailed statistics and visuals from the test results. Given the room for improvement, the current results are promising.