CS230

Personalized Motion Models to Predict Short-Term Lower-Limb Kinematics

Akshay Sreekumar
Department of Electrical Engineering
Stanford University
Apple
akshay81@stanford.edu

Abstract

Accurately forecasting lower-limb kinematics in free-living situations presents the opportunity to improve the quality of life for those suffering from mobility impairments. Improvements in wearable intertial sensors and bio-mechanics modeling have made available a rich collection of subject specific free-living kinematics data. In this paper, we use data that measured kinematics (left/right hip and knee angles) of a subject over the course of week to develop personalized motion models using both fully connected neural networks and recurrent neural networks (RNN). The RNN is able to achieve a mean RMSE of 6.11°and generate reasonable forecasts of human motion.

1 Introduction

Several people suffer from challenges with limited mobility. Limited mobility can be the result of a variety of factors: obesity, arthritis, muscle control, etc. Assistive technologies such as exoskeletons and prostheses offer potential solutions, but require intelligent control that can ensure safe assistance in a variety of "free-living" conditions such as walking or running. These assistive systems need to be able to predict joint kinematics (i.e. hip and knee angles) with high accuracy over the broad class of activities involved in day-to-day living. Advances in inertial based sensing and biomechanics modeling have enabled accurate and inexpensive methods for real-time joint estimation. However, the challenge remains of building a real-time system that can *forecast* future motions. The predictive capabilities of such a system would be invaluable for predicting abnormal motion events such as falls, fatigue, or injury while users go about their lives. To this end, we have built personalized motion models that take as input a time series data of left/right knee and hip angles over some duration. We then use an RNN to output a single set of joint angles 150ms in the future from the input data.

2 Related work

Prior work has focused on the lower-limb kinematics estimation problem using statistical models [1,2]. Statistical models are computationally effective and require fewer sensors, but have not met the accuracy requirements for precision applications. In Findlow et. al, a generalized regression neural network (GRNN) was used to predict lower-limb kinematics from inertial data, but they did not seek to do "forecasting". Similarly, the constrained Kalman filter approach used in [2] allowed for reduced computational costs, but struggled with poor accuracy compared to a motion capture system. In general, I believe that more data driven approaches (rather than purely statistical methods) are necessary for this problem due to the complexity and unpredictability of human motion patterns in a variety of free-living situations. Previous methods have not had the luxury of an extensive data set such as the one available for use in this project. The development of a more compact and accurate inertial measurement system that uses OpenSense [3,4], an open source software tool for converting IMU data to joint angle estimates, has allowed for the collection of a tremendous amount of high quality, accurate data that is natural and unconstrained by a laboratory environment.

Similar work has used non-inertial sensor systems such as motion capture or EMG. Motion capture data, while a gold standard in joint-angle estimation, is expensive and only suited for laboratory environments. Thus, it is unable to capture more natural movements that would be found in free-living situations. EMG systems are costly, noisy, and bulky. Wearable IMU systems are compact, low-cost, and provide the necessary sensor information for real-time kinematics.

3 Dataset and Features

A wearable sensor system with IMUs mounted on the pelvis, left/right shank, left thigh, and right thigh has been used to collect inertial data at a sampling rate of 60Hz on 3 different subjects going about their daily activities over several days. OpenSense has been used to compute joint angle estimates from the raw inertial measurements. Thus, we ultimately have as input a time-series of data with joint angles every 60Hz to feed into the model. Because the goal is to construct a personalized motion model, data for only a single subject across several days has been split into several sections of motion and then further split into training (15925 samples), dev (1887 samples), and test (957 samples) sets. The collection of the data and development of the wearable IMU system was performed by Patrick Slade, a 5th year PhD student at Stanford who is part of the Stanford Intelligent Systems, Neuromuscular Biomechanics, and Biomechatronics labs (see acknowledgements).

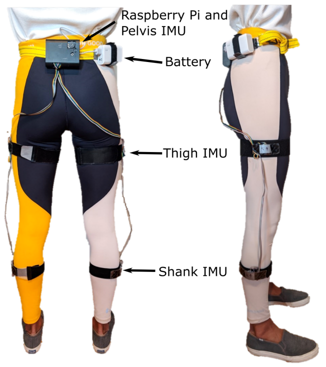

Figure 1: Wearable IMU System for Collecting Inertial Data

4 Methods

The typical human gait cycle is on the order of 1-2s, so the input time has been set to 4s to capture the effect of several gait cycles. This corresponds to $t_{buffer} = 4s$ as the system will first need to accumulate that many samples. A longer prediction horizon gives more information, but one can expect a decrease in accuracy for far out time points. Given that we are "wasting" our resources optimizing for inaccurate and long time-scales, we have opted to use a single point horizon. The final consideration is how far into the future we want to predict, which we shall call horizon delay. Given that this is a real time system computing joint kinematics at a rate of 60Hz, the smallest amount of time we can predict in advance is 16.67ms. Again, it is intuitive that keeping horizon delay as small as possible will give us the most accuracy, but the typical hardware constraints for robotics dictates that the actuation latency, $t_{latency}$, for hardware to act on a prediction is on the order of around 150ms. Thus, we choose to set our horizon delay to 150ms. A cartoon of this prediction architecture for a real time system is shown below.

This overall scheme has been applied to the training of two separate models: a fully connected neural network as well as an RNN. The FC network helped to serve as a baseline for performance using an

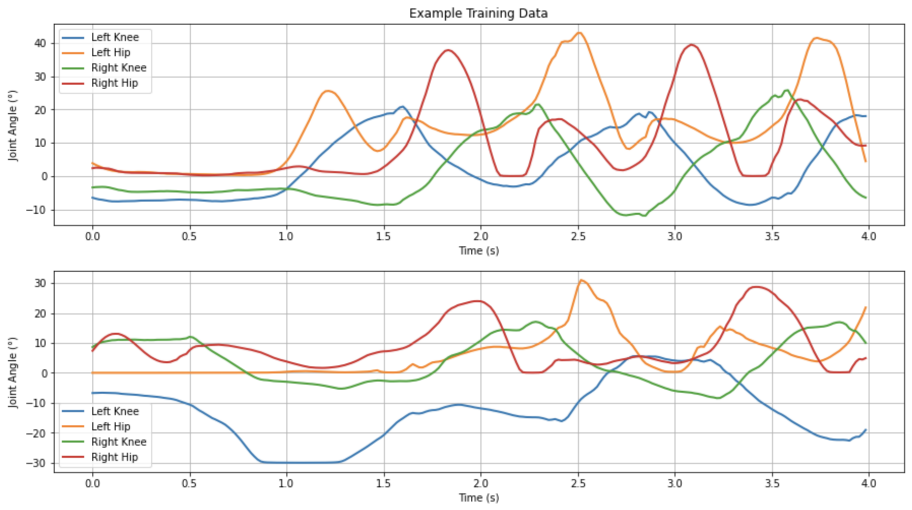

Figure 2: Joint Kinematic Input Data

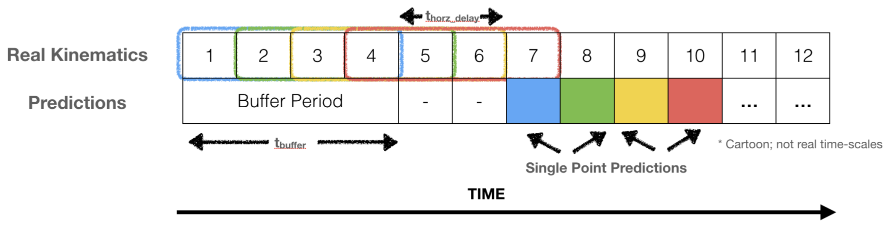

Figure 3: Real Time Observer Architecture for Forecasting Joint Kinematics and Events

architecture that did not exploit the series nature of the data, while the RNN was used to do exactly that. In both cases, I used Adam's as my learning algorithm along with a mean-squared error (MSE) loss function (see section 5 for more reasons why). The FC network is straightforward: it flows the input through a series of ReLU nonlinearities to compute activations, and then follows up with the backpropagation step. The 4D vector of joint angles produced by this network are evaluated using the mean squared error loss function. The MSE loss is defined as:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y-\hat{y})^2$$

The RNN has an advantage over the FC in the sense that is assumes that the data is working with is a sequence, and thus there are relationships that are important between points that are near each other temporally in the input sequence. The RNN is able to accomplish this by allowing previous outputs to be used as inputs while also maintaining internal hidden states. At each time step, backpropagation through time is performed. In this case, an RNN was deemed to be sufficient because motion data largely depends on data that is very close temporally. Unlike in other applications such as NLP where RNN's are popular, there are usually no very long dependencies that occur (such as how the end of a sentence can be linked to the very first word somehow).

5 Experiments/Results/Discussion

The model predicts four different joint angles: left/right knee/hip. These four angles constitute a pose. We want to limit errors on the individual joint kinematics themselves, but we also want some way to consider the accuracy of the overall pose. The first constraint is best captured by mean absolute error (MAE) between the predictions predictions and ground truth. The second goal is best achieved using a mean squared error metric (MSE) which takes into account errors over all the joints and thus better represents the error in a given pose. I have been minimizing the MSE loss function because in practice it is easier to deal with large excursions in a single joint. If there is a large joint angle error, we could in practice recognize it as un-physical and reject it. However, we greatly value the ability to ascertain the overall pose of an individual.

To iterate on models quickly, I compute a single metric to compare the efficacy of different models. For this application, I compute the root mean square error (RMSE) between the predicted joint angles and actual joint angles. I average these RMSE's across the training set to compute what I denote mean RMSE. On top of this set level analysis, I have also performed sliding window real time predictions on time-series data to get a more intuitive feel for the algorithm's performance.

The primary hyperparameters I focused on tuning were the learning rate and the number of hidden units. I have been using the cross-entropy method to efficiently search the hyperparameter space. The cross-entropy method iteratively generates Gaussian distributions to sample from for each hyperparameter—generating the next distribution by examining the best parameters from the previous generation. The "best" parameters are selected from a generation by choosing those that yielded the lowest mean RMSE on the dev set. In this way, instead of doing a grid search or a random search, I am able to perform a more targeted search in successively smaller regions by seeding successive generations with the statistics from the previous one.

Below, I include a result from the FC network which was intended to serve as a baseline model for performance.

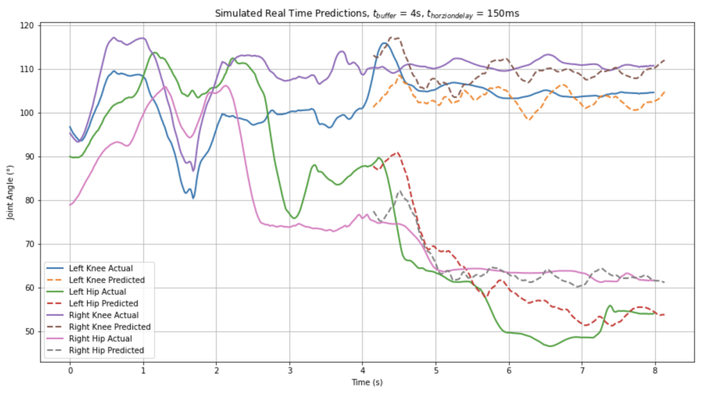

Figure 4: Simulated Real Time Predictions using FC Model

We can see that there seems to be an implicit delay in the predictions, suggesting that the model isn't really learning to predict in advance 150ms, but rather is simply learning to predict the very next time step. Because the motions aren't too abrupt, the model is able to get by doing this. Overall, the trends of the motions seem to match, but there is clearly more work to do in getting better predictions. Instead of continuing to improve the FC model, I spent my remaining time investigating the use of an RNN for this problem as it is more capable of exploiting the sequence structure.

On the same set of input data, I have included the predictions made by the RNN on just the right hip. While there is still the obvious problem of a slight delay (implying that the model needs to be further refined to make more useful predictions), we can see that in general it seems to perform better than the FC network. There are less variations in the shape of the predicted joint angles, and the overall character of the prediction is much more closely aligned with the ground truth.

However, we can see examples of input data for which the RNN appears to have more obvious flaws.

Here, while the RNN is doing a good job of estimating the general periodic motion of the input, it makes gross errors in the peak angle (of 20°). For reference, a clinically acceptable application demands errors no greater than 5°.

In terms of set level analysis, the RNN achieved a mean RMSE of of 2.75°on the training set, and 6.11°. While this seems close to the clinical target, the mean RMSE is perhaps an overly reductive metric and further work would be required to understand max deviations from ground truth in a

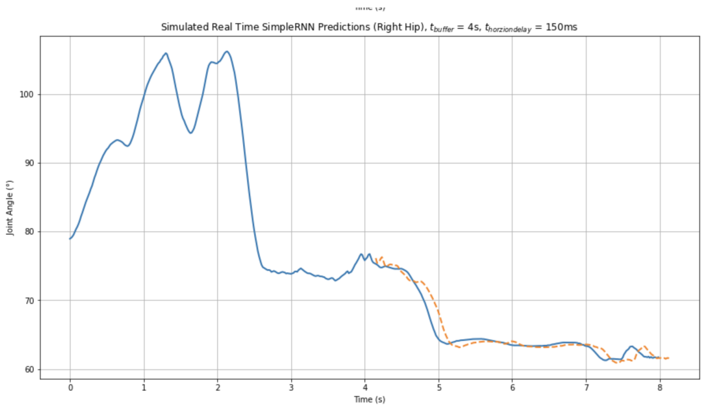

Figure 5: Simulated Real Time Predictions using RNN Model

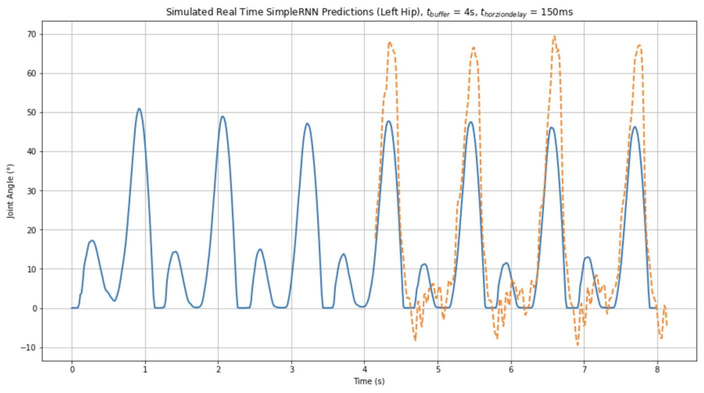

Figure 6: Simulated Real Time Predictions using RNN Model

variety of different motion patterns. The results do suggest that the model has high variance. To mitigate this, I did try adding L2 regularization and inverted dropout, but did not have the opportunity to sweep the probability of dropping a node as a hyperparameter. L2 regularization encourages a smaller model by penalizing weights that are too large, while inverted dropout does so making sure neurons are not overly dependent on each other. I primarily spent my efforts on reducing the bias of the models to begin with.

6 Conclusion/Future Work

We have been able to evaluate two different models, a fully connected NN and an RNN, for predicting lower-limb joint kinematics in free-living conditions. The RNN was able to provide better performance for this task likely because it was able to take advantage of the short-term temporal relationships present in human motion sequences.

In the future, I would be interesting in trying other recurrent models such as LSTMs. Although LSTMs are primarily valuable in modeling long term relationships in sequences (which are not typically present in motion data), I would be curious to understand if there are scenarios (like long periods of walking) where this would indeed be helpful. Furthermore, I would be interesting in seeing if I could exploit the semi-periodic nature of the data by using sparser representations (Fourier) or different choices of activation functions (sinusoidal). Finally, I believe both models could benefit significantly from continued hyperparameter tuning.

7 Acknowledgements

This project was supervised by Patrick Slade (patslade@stanford.edu). Patrick is a 5th year PhD student at Stanford, and this CS230 project is being done as part of his research in the Stanford Intelligent Systems, Neuromuscular Biomechanics, and Biomechatronics labs. Patrick has been invaluable not only in providing the rich set of data that this project depends on, but also in his support and feedback as I made progress on the project. His extensive domain knowledge helped to enable the work done on this problem.

References

[1] A. Findlow, J.Y. Goulermas, C. Nester, D. Howard, L.P.J. Kenney, Predicting lower limb joint kinematics using wearable motion sensors, Gait Posture, Volume 28, Issue 1, (2008), Pages 120-126, ISSN 0966-6362, https://doi.org/10.1016/j.gaitpost.2007.11.001.

[2] Luke Sy, Michael Raitor, Michael Del Rosario, Heba Khamis, Lauren Kark, Nigel H. Lovell, Stephen J. Redmond. (2020). Estimating Lower Limb Kinematics using a Reduced Wearable Sensor Count.

[3] Seth, A., Hicks J.L., Uchida, T.K., Habib, A., Dembia, C.L., Dunne, J.J., Ong, C.F., DeMers, M.S., Rajagopal, A., Millard, M., Hamner, S.R., Arnold, E.M., Yong, J.R., Lakshmikanth, S.K., Sherman,n M.A., Delp, S.L. (2018) OpenSim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. Plos Computational Biology, 14(7).

[4] Delp, S.L., Anderson, F.C., Arnold, A.S., Loan, P., Habib, A., John, C.T., Guendelman, E., Thelan, D.G. (2007) OpenSim: Open-source software to create and analyze dynamic simulations of movement. IEEE Transactions on Biomedical Engineering , vol 55, pp 1940-1950.

[5] Roetenberg, Daniel Luinge, Henk Slycke, Per. (2009) Xsens MVN: Full 6DOF human motion tracking using miniature inertial sensors. Xsens Motion Technol. BV Tech. Rep.. 3.

[6] Scikit-learn: Machine Learning in Python, Pedregosa et al. (2011) JMLR 12, pp. 2825-2830

[7] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng (2015) TensorFlow: Large-scale machine learning on heterogeneous systems

[8] Chollet, Franet al. (2015) "Keras." https://keras.io.