Your Next Personal Trainer: Instant Evaluation of Exercise Form

Brandon Garcia, Russell Kaplan, Aditya Viswanathan

Introduction

Maintenance of a healthy lifestyle depends on regular exercise. However, many exercisers suffer from improper form during their workouts, reducing the health benefits of exercise and increasing risk of injury. Identification of proper form generally requires the aid of a personal trainer—we aim to do so with machine learning.

We approach this problem by focusing on a canonical lower body exercise: the free-standing squat. Given input data from a Microsoft Kinect v2, our objective is predict which aspects of form the exerciser can improve the most. We use a series of independent squat component classifiers (e.g. a depth classifier, a knees-over-toes classifier, and so on) to output independent binary or multinomial classifications: whether that component of form is good or bad (or in the case of multinomial, in what way it is bad). We also expand our classification pipeline to include basic support for push-ups in order to demonstrate the ease with which our system generalizes to other exercises. In addition to demonstrating a viable machine learning approach to evaluate exercise form, we include a live system that can perform evaluation of exercisers on the fly using our models (see references for our GitHub repository and iPython Notebooks).

Related Work

The Kinect has already been used for form detection in rehabilitation patients with reasonable success. Such experiments have classified exercises and served as automated data collection tools by mapping skeletal coordinates afforded by the Kinect to expected physical characteristics [1]. However, this form of data collection does not provide real-time feedback by analyzing the correct and incorrect characteristics of exercises and merely enables post-processing of rep features in bulk. Recent collaboration between Nike and Microsoft led to the development of the Nike + Kinect Training exercise game dedicated to improving form by scoring workouts monitored by the Kinect. While this platform accounts for a variety of user sizes, it applies pattern matching on scaled Kinect skeletal data to a specific "form ideal" rather than generalizing from a set of examples of good form across various user sizes [2]. Moreover, the platform emphasizes that users increase the number of reps that fit an acceptable framework rather than improve the success of individual reps (and thus constrain the bounds of some acceptable framework) [2]. This solution is therefore unhelpful in diagnosing specific issues with bad form.

Because of the Kinect's ability to track a variety of limb-specific coordinates across the body, it has also been instrumental in the design of training systems for Taekwondo and other martial arts. However, these approaches simply score the similarity between a series of recorded frames and the expected results and manually set tolerances to simulate the classification of good and bad form in martial arts techniques [3]. Therefore, such systems are not easily generalized to broader bases of users across a variety of sizes.

Machine learning has also been applied to extracted skeletal data for the recognition and classification of human gestures. By articulating a gesture vocabulary with respect to joint-wise components, Kinect streams were parsed into discrete gestures and then each gesture was classified based on component-level motion of skeletal coordinates [4]. In a case study classifying airplane gestures used during takeoff, Support Vector Machines and Decision Trees proved fruitful in proper classification across a variety of subjects [4]. The success of these approaches, especially given the similarity of the problem space (generally, splice Kinect streams into discrete units for component-wise classification), motivated our design of Kinect-based exercise analysis. Activity recognition has been applied to fitness with weight lifting exercises, but has required the involvement of a variety of wearables to deliver accurate feedback [5]. Our approach seeks to use the Kinect as the sole, noninvasive sensor from which to derive all necessary data to provide specific rep-level feedback to users and enable a pipeline for a variety of standard exercises.

Dataset and Features

Due to the novelty of our objective, we could not find a suitable existing dataset to use for our project. We collected and labeled our own dataset of roughly 250 squats and 100 push-up from a group of nearly 40 volunteers. To do so, we recorded exercisers with a Microsoft Kinect v2 doing sets of 10 squats or push-ups and extracted the x,y,z coordinates of 25 joints for each frame of the recording. We wrote an algorithm to segment our raw time-series data into discrete repetitions ("reps") of each exercise. After this segmentation, we normalized each rep to a coordinate system with unit range for each x,y,z. We gave each rep labels in eight categories, corresponding to the eight component classifiers we sought. Five components proved useful: knees-over-toes, bend hips and knees at the same time, squat depth, back-versus-hip angle, and stance. To minimize inconsistency, the same reviewer was responsible for all labels of a component.

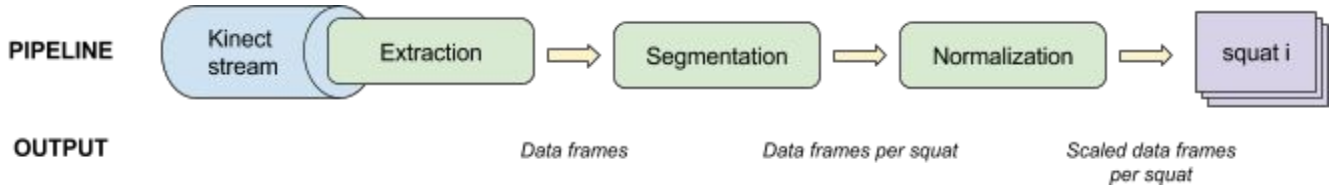

Figure 1: Data pipeline

To extract features, we select *r* discrete frames from a given rep and compute a variety of features using joint coordinates in that frame, where *r* is a frame resolution hyperparameter that we tuned separately for each component classifier. We designed features for each component based on a review of fitness literature [6]. We found that some component classifiers performed best with our entire feature set, whereas others work better with only the features we designed for that specific component. Common features are angles between joints at key frames (like the bottom of the rep), changes in angles over time, scaled positions of joints at key frames, changes in those positions over time, and relative distances between joints. The set of values for each feature across all of our data was normalized to unit variance and 0 mean.

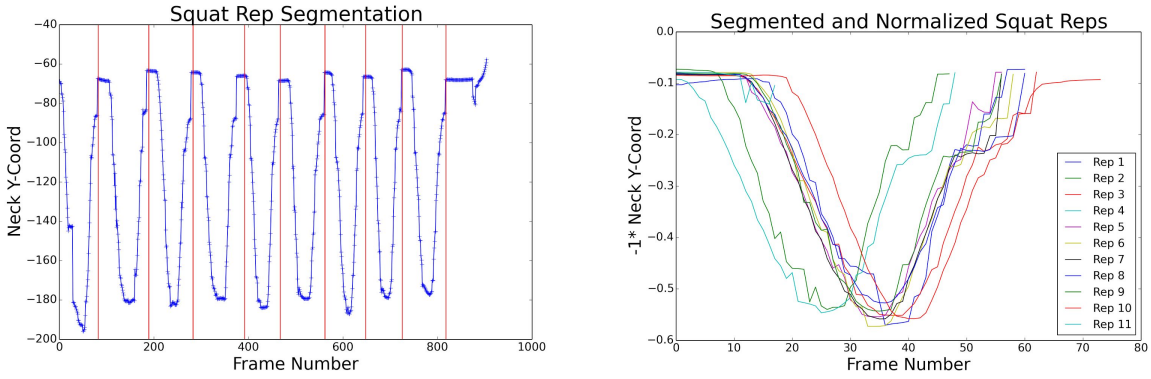

Figure 2: (left) Rep segmentation, denoted by the red vertical lines; (right) Each rep's changing neck height over time, after superimposing reps and normalizing height.

Methods

After experimentation with several different learning models, we settled on using one of two for each component classifier: logistic regression or decision trees. Logistic regression works by squashing the output of a linear regression model into the range {0,1} with the logistic function, where for binary classification any output of the logistic function above 0.5 is classified as positive and outputs are negative otherwise. We use L1 regularization with logistic regression to prevent overfitting (which performed better than L2 regularization in our testing). We use the hypothesis function for logistic regression, both binary and generalized to K classes, the latter being defined via the *softmax* function:

$$h(x) = p(y = 1 \mid x) = \frac{1}{1 + \exp(-\theta^T x)} \qquad h(x) = p(y = k \mid x) = \frac{\exp(\theta_k^T x)}{\sum_{i=1}^{K} \exp(\theta_i^T x)}$$

The binary objective function, using L1 regularization with different values of λ depending on the component classifier, is:

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} |\theta_j| \right]$$

We used decision trees for two component classifiers: knees-over-toes and bend hips and knees at the same time. Decision trees learn by recursively splitting data into subsets based on attribute values at each node. When the remaining data in a subset all have the same value at a particular node, the recursion is complete and that node becomes a leaf. At test time, data flows through the decision tree branching left or right at each node depending on the attribute value of the relevant variable. The class label at the leaf node at the end of this process is the output of the classifier.

How do decision trees decide how to split the data at each node? The default splitting metric in scikit-learn [7]'s decision tree implementation is Gini impurity, which measures how often a randomly chosen element in the set would be labeled incorrectly if labeled randomly based on the label distribution in the subset. This is good for regression problems but didn't perform well in the classification problem we faced. Instead, we used entropy as the splitting metric, which seeks to maximize information gain:

$$I_E(f) = -\sum_{i=1}^{m} f_i \log_2 f_i$$

Where f_i is the probability of the *i*th item being chosen. Intuitively, this metric splits data based on the *most informative metric* for that data, where most informative is defined in information-theoretic terms. This means that mutual information within the tree is maximized, which often yields better results than Gini impurity on classification problems where one class dominates the training dataset (as is the case for us: we have a scarcity of positive labels). Furthermore, for our decision tree classifiers we use a max-depth parameter of 3, which helps guard against overfitting by preventing the tree from growing too deep.

Experimental Results and Analysis

For each classifier, we ran experiments using SVMs, logistic regression, decision trees and random forests. We tuned the hyperparameters for these models independently for each classifier and experimented with regularization (for logistic regression and SVMs) as well as the maximum depth hyperparameter and different splitting metrics for random forests and decision trees largely in an effort to reduce overfitting. The performance of each model with optimized hyperparameters is reported in the table below.

We tested each squat component individually. For each, we ran LOOCV on our various classifiers with little to no hyper parameter tuning aside from varying resolution and component-specific features vs the entire feature set. We tested on as few as four frames ($r = 4$) to as many as two hundred frames ($r = 200$). The latter would often result in some frames being counted twice for any given rep and would take between six and eight minutes to extract our features. Training generally took less than two minutes, even with maximum frames and features while classification was near instant. LOOCV was chosen due to the small size of our training set.

We first ran each iteration of inference evaluation with component-specific features extracted from our frames while varying *r*. Similarly, we then ran our classifiers with all features extracted from our *r* frames. This would give us an initial intuition for whether the component classification being analyzed fared better with just component-specific features, or the entire set of features we originally constructed. We primarily evaluated our models' performance by comparing F1 scores and accuracies.

This all served to help us to decide whether to use component-specific features or our entire feature set, a decision that usually became quickly apparent after some initial runs. We then looked more closely at the individual performance of our classifiers across all iterations of *r* but with the now-specified feature set. This allowed us to narrow down our original set of classifiers and then start testing the most promising models while tuning the aforementioned hyper parameters until we reached our optimal solution.

Component classifier	**Optimal model and parameters**	**Test results with LOOCV**	
Squat: knees over toes	Decision tree. Maximum depth of 3 and using entropy as the splitting metric.	Accuracy 0.805 Precision: 0.819	Recall: 0.916 F1-score: 0.865
Squat: bend hips and knees at same time	Decision tree. Maximum depth of 3 and using entropy as the splitting metric.	Accuracy 0.898 Precision: 0.556	Recall: 0.577 F1-score: 0.566
Squat: depth	Logistic regression with L1 regularization with a regularization constant of 0.05	Accuracy 0.824 Precision: 0.757	Recall: 0.920 F1-score: 0.831
Squat: back-hip angle	Logistic regression with L1 regularization with a regularization constant of 2.5	Accuracy 0.758 Precision: 0.670	Recall: 0.788 F1-score: 0.724
Squat: stance	Logistic regression with L1 regularization with a regularization constant of 0.75	Accuracy 0.897 Precision: 0.829	Recall: 0.723 F1-score: 0.773
Push-up: elbow angle	Logistic regression with L1 regularization with a regularization constant of 1	Accuracy 0.964 Precision: 0.917	Recall: 0.786 F1-score: 0.846
Push-up: head and back alignment	Logistic regression with L2 regularization with a regularization constant of 1	Accuracy 0.618 Precision: 0.549	Recall: 0.609 F1-score: 0.577
Push-up: leg straightness and alignment	Logistic regression with L1 regularization with a regularization constant of 8	Accuracy 0.927 Precision: 0.714	Recall: 0.714 F-score: 0.714

Figure 3: Performance of different component classifiers in their optimized form

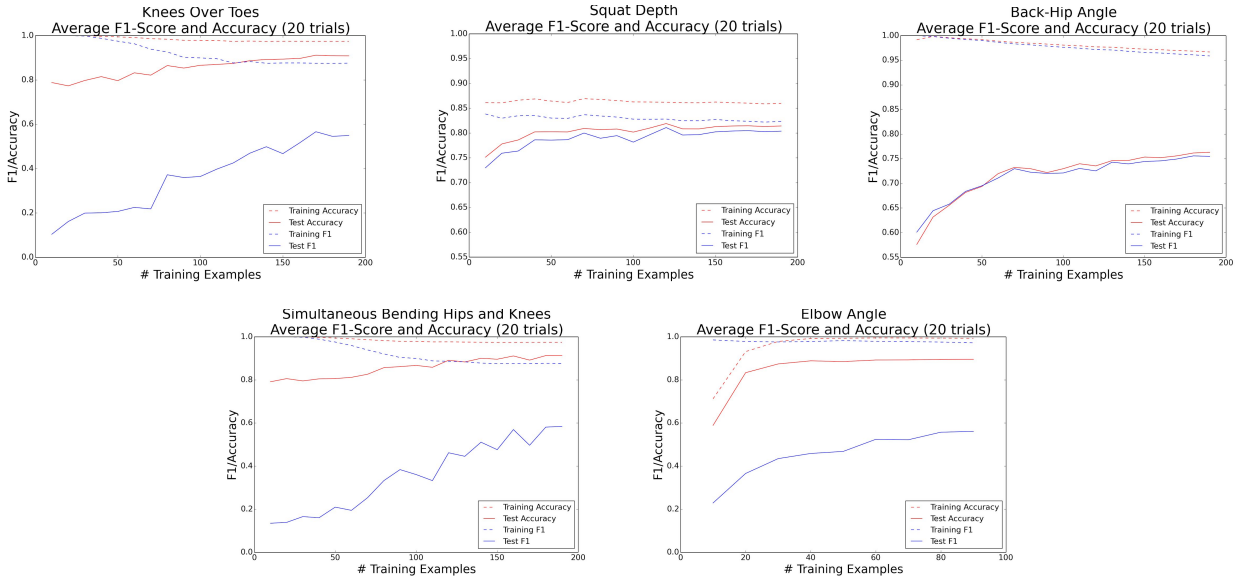

Figure 4: Train vs test accuracy / F1 score for select component classifiers

There are several interesting takeaways we can draw from our experimental results. For one, we observe that for most component classifiers, test error is distinctly lower than training error. This reflects some of the challenges of our dataset: namely, we are extremely data-constrained. In several of the graphs, such as knees-over-toes and simultaneous-hips-and-knees, it seems that the F1 test performance is not yet close to leveling out, suggesting that more training data would noticeably improve performance.
Overfitting was a consistent challenge throughout this project. For example, SVMs, even with a heavy regularization penalty, often overfit so thoroughly to our training data that the SVM test results could not present F1 scores comparable to decision trees or logistic regression. Knowing how sensitive the regularization constant can be for SVMs, it might be possible that we did not do a granular enough search across constants.

We tried several approaches to combat this overfitting problem. We experimented with both L1 and L2 regularization for our logistic regression models for each classifier and found that L1 performed best in several of them—likely because we had somewhat sparse feature vectors. Whereas we originally used every feature for each component classifier, we later tried removing most features for each component classifier to constrain the hypothesis class of each. This worked well for the classifiers of state-based components like squat depth, which is determined almost entirely by the bottom frame of the rep. For others, like knees-over-toes, we found that keeping in all features produced better performance. Our hypothesis for this difference is twofold: 1) state-based components are simpler and generally determined by the instantaneous positions of the relevant joints in the relevant frames, whereas components that depend on consistency, like simultaneous bending of hips and knees, are determined over time and usually require full-body involvement; 2) performance on some components will influence others by basic rules of anatomy and physics. For example, it is nearly impossible to keep your back vertical throughout a squat (which is bad—the back should be angled 45º down) without bending your knees past your toes (also bad). You would almost certainly fall backward if your knees were correct. Thus, features that we originally wrote with our back-hip angle classifier in mind were also relevant for the knees-over-toes classifier.

This analysis is one example of the value of our domain-specific knowledge throughout this project. By studying the exercise literature—plus the quirks of our dataset—closely, we gained several advantages in improving performance. As an example, knowing that we were data-constrained led us to rethink the states we were using for feature extraction. We originally only considered extracting features with resolution r =4. But after running initial experiments and concluding that we needed more data, we observed we could multiply our effective amount of contributing data considerably without any new recordings by simply increasing the resolution size and no longer throwing out so many frames per rep. Experiments confirmed that higher resolution is better for components that occur over time, and low resolution excels for simpler components that depend less on motion in time-series.

However, we still have significant room for improvement. We ran several of the above experiments for the purposes of error analysis, and we can see different components may require different adjustments to improve. For example, our test F1 score is rather low in elbow angle and simultaneous hip/knee bending. We looked at our training labels for these components and observed that they also tended to have the fewest nonzero labels in the training set. This suggests that getting more examples of poor form would help us improve in these areas. The problem may be particularly acute due to our use of LOOCV for performance measurement. For elbow angle in push-ups, only three people ever received nonzero labels out of eighteen push-up volunteers recorded. When we leave one of

these three people out from the training phase, there are only two sets out of seventeen with examples of bad elbow angles. Even though each set contains 10 reps of push-ups, this is hardly enough data to achieve strong results with machine learning.

Data scarcity wasn't our only issue. Unlike the clear overfitting problems already discussed, our squat depth classifier has relatively consistent training versus test performance. But its training performance is the lowest in the group by a noticeable margin. Why? Upon closer inspection, we realized that squat depth was the hardest component for us to label consistently. There were so many borderline cases where we were unsure of the correct label (maybe a squat was almost deep enough or just barely deep enough) that our dataset likely contains several incorrect labels. This would make training accuracy impossible to get near 100% with a linear classifier and the feature set we chose. Even a nonlinear classifier wouldn't be able to achieve 100% training accuracy without rote memorization. Thus, for future improvements we would experiment with more robust labeling techniques, like making sure each squat is labeled by multiple people and taking the median label and/or hiring professional fitness experts to do the labeling directly.

Another source of error was due to noise from our recording instrument: the Kinect. Inspection of the raw skeletal recording data revealed that oftentimes, the Kinect records joints as jumping several inches to the side and then back again over the course of a few frames, despite such behavior being anatomically impossible. We originally had variance of joint position as a feature in several of our component classifiers, because when people shake during their squats or push-ups it indicates they are having trouble, oftentimes a corollary of bad form. After our discovery of this error source we removed variance from most feature sets and saw improved performance (which also suggests overfitting: if joint variance primarily reflected noise from the Kinect and not a meaningful indication of bad form, we would expect the feature to be weighted near-zero and thus not notice a performance improvement after removing it. The fact that we did observe an improvement suggests it was not, and thus that we were fitting at least in part to random Kinect noise.) Sensor noise was especially challenging for push-up classification, as the Kinect is not designed to predict skeletons from a side angle and so the joint coordinates are particularly unreliable. We had to discard entire components we originally intended to use for both squats and pushups because of how inaccurate the head joint is from the Kinect.

Surprisingly, we also observed increased test error when using random forests in place of decision trees for the component classifiers where we report decision trees as the best model. We did not expect this result, as random forests rely on decision trees internally but incorporate a voting mechanism among several trees to correct for their tendency to overfit when used in isolation. We suspect this result is due to suboptimal hyperparameters for our random forests, and that with more extensive tuning of maximum tree depth, node-splitting and data sampling hyper parameters, we could get them to outperform decision trees.

Conclusion

In summary, we present a robust pipeline for exercise form evaluation, built on production-ready infrastructure and independent classifiers for several distinct exercise form components. We demonstrate promising results for many of the component classifiers; notably knees-over-toes, stance, and squat depth. Furthermore, we show that our approach is generalizable with acceptable baseline results for push-ups, an entirely separate exercise. To continue this work, we would like to collect more training data to better understand how our test performance will scale. We would also like to explore further parameter tuning and invest in professional labeling. Finally, it would be exciting to use machine learning on the outputs of these component classifiers to generate English sentences characterizing the results, as a personal trainer might in the real world.

References

[1] Muijzer, Frodo. "Development of an Automated Exercise Detection and Evaluation System Using the Kinect Depth Camera." *Faculty of Electrical Engineering, Mathematics, and Computer Science, Twente University*. Web.

[2] Palma, Carlos, Augusto Salazar, and Francisco Vargas. "Real time Kinect evaluation of therapeutical gestures." *Signal Processing, Images and Computer Vision (STSIVA), 2015 20th Symposium on*. IEEE, 2015.

[3] Sani, Nisfu Asrul, M. Afif Hendrawan, and Febriliyan Samopa. "Development Of Basic Taekwondo Training System Application Based On Real Time Motion Capture Using Microsoft Kinect." *ISICO 2015* (2015).

[4] Bhattacharya, Surya, Bogdan Czejdo, and Noel Perez. "Gesture classification with machine learning using kinect sensor data." *Emerging Applications of Information Technology (EAIT), 2012 Third International Conference on*. IEEE, 2012.

[5] Velloso, Eduardo, et al. "Qualitative activity recognition of weight lifting exercises." *Proceedings of the 4th Augmented Human International Conference*. ACM, 2013.

[6] Schoenfeld, Brad. “Squatting Kinematics and Kinetics and their Application to Exercise Performance.” *Journal of Strength and Conditioning Research.* 2010.

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

Links to iPython Notebook viewers and our GitHub repo:

1. Github Repo: https://github.com/bgarcia7/ai_trainer
2. Data Preprocessing iPython NB Viewer: http://nbviewer.ipython.org/github/bgarcia7/ai_trainer/blob/master/development_notebooks/Data%20Preprocessing%20Demonstration.ipynb
3. Squat Inference iPython NB Viewer: http://nbviewer.ipython.org/github/bgarcia7/ai_trainer/blob/master/development_notebooks/Squat%20Inference%20Demonstration.ipynb
4. Full Pipeline iPython NB Viewer: http://nbviewer.ipython.org/github/bgarcia7/ai_trainer/blob/master/development_notebooks/Pipeline%20Demonstration.ipynb