Activity Recognition from Accelerometer Data

Nishkam Ravi and **Nikhil Dandekar** and **Preetham Mysore** and **Michael L. Littman**
Department of Computer Science
Rutgers University
Piscataway, NJ 08854
{*nravi,nikhild,preetham,mlittman*}@*cs.rutgers.edu*

Abstract

Activity recognition fits within the bigger framework of context awareness. In this paper, we report on our efforts to recognize user activity from accelerometer data. Activity recognition is formulated as a classification problem. Performance of base-level classifiers and meta-level classifiers is compared. Plurality Voting is found to perform consistently well across different settings.

Introduction

A triaxial accelerometer is a sensor that returns a real valued estimate of acceleration along the x, y and z axes from which velocity and displacement can also be estimated. Accelerometers can be used as motion detectors (DeVaul & Dunn 2001) as well as for body-position and posture sensing (Foerster, Smeja, & Fahrenberg 1999). Apple's iLife Fall Detection sensor which embeds an accelerometer and a microcomputer to detect falls, shocks or jerky movements is a good example. Active research is being carried out in exploiting this property for determining user *context* (Randell & Muller 2000). Advances in miniaturization will permit accelerometers to be embedded within wrist bands, bracelets and belts and to wirelessly send data to a mobile computing device that can use the signals to make inferences. User context can be utilized for ambulatory monitoring (Makikawa *et al*. 2001; Foerster, Smeja, & Fahrenberg 1999) and is the key to minimizing human intervention in ubiquitous computing applications.

Making devices aware of the activity of the user fits into the bigger framework of context awareness. Ubiquitous computing is centered around the idea of provisioning services to the user in a seamless manner. Provisioning services to the user based on his location and/or activity is an active research area. While the research thrust is on automatically determining user location (Want *et al*. 1992; Harter & Hopper 1994; Priyantha, Chakraborty, & Balakrishnan 2000), determining user activity is getting a lot of attention lately. Attempts have been made at recognizing user activity from accelerometer data (Lee & K.Mase 2002; Bussmann *et al*. 2001). The most successful and exhaustive work in this regard is that of Bao & Intille (2004). In their experiments, subjects wore 5 biaxial accelerometers on different body parts as they performed a variety of activities like walking, sitting, standing still, watching TV, running, bicycling, eating, reading etc. Data generated by the accelerometers was used to train a set of classifiers, which included decision trees (C4.5), decision tables, naive Bayes classifier and nearest-neighbor algorithm found in the Weka Machine Learning Toolkit (Witten & Frank 1999). Decision tree classifiers showed the best performance, recognizing activities with an overall accuracy of 84%.

We have attempted to recognize activities using a single triaxial accelerometer worn near the pelvic region. Activity recognition is formulated as a classification problem. In addition to analyzing the performance of base-level classifiers (Bao & Intille 2004), we have studied the effectiveness of meta-level classifiers (such as boosting (Freund & Schapire 1996), bagging (Breiman 1996), plurality voting, stacking using ODTs, and stacking using MDTs (Todorovski & Dzeroski 2003)) in improving activity recognition accuracy. We have tried to answer the following questions: (1) Which are the best classifiers for recognizing activities; is combining classifiers a good idea? (2) Which among the selected features/attributes are less important than others? (3) Which activities are harder to recognize?

In the following sections, we describe our data collection methodology and our approach to recognize activity from accelerometer data, followed by results.

Data Collection

Data from the accelerometer has the following attributes: time, acceleration along x axis, acceleration along y axis and acceleration along z axis. We used a triaxial accelerometer CDXL04M3 marketed by Crossbow Technologies, which is capable of sensing accelerations up to 4G with tolerances within 2%. The accelerometer is mounted on a hoarder board (which samples at 50Hz), as shown in Figure 1. The accelerometer was worn near the pelvic region while the subject performed activities. The data generated by the accelerometer was transmitted to an HP iPAQ (carried by the subject) wirelessly over Bluetooth. The Bluetooth transmitter is wired into the accelerometer. A Bluetooth enabled HP iPAQ running Microsoft Windows was used. The Windows'

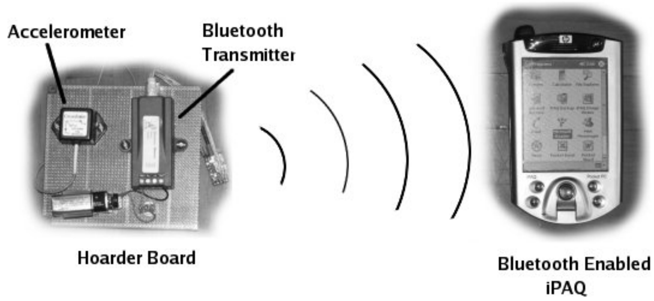


Figure 1: Data Collection Apparatus

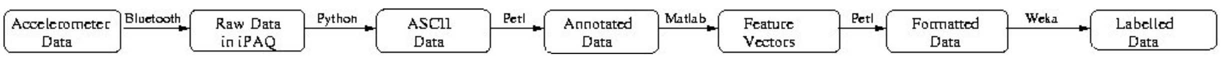


Figure 2: Data Lifecycle

Bluetooth library was used for programming Bluetooth. The data was then converted to ASCII format using a Python script.

We collected data for a set of eight activities:

- Standing
- Walking
- Running
- Climbing up stairs
- Climbing down stairs
- Sit-ups
- Vacuuming
- Brushing teeth.

The activities were performed by two subjects in multiple rounds over different days. No noise filtering was carried out on the data.

Label-generation is semi-automatic. As the users performed activities, they were timed using a stop watch. The time values were then fed into a Perl script, which labeled the data. Acceleration data collected between the start and stop times were labeled with the name of that activity. Since the subject is probably standing still or sitting while he records the start and stop times, the activity label around these times may not correspond to the actual activity performed.

Figure 2 shows the lifecycle of the data. To minimize mislabeling, data within 10 s of the start and stop times were discarded. Figure 3 shows the x-axis readings of the accelerometer for various activities.

Feature extraction

Features were extracted from the raw accelerometer data using a window size of 256 with 128 samples overlapping between consecutive windows. Feature extraction on windows with 50% overlap has demonstrated success in previous work (Bao & Intille 2004). At a sampling frequency of 50Hz, each window represents data for 5.12 seconds. A window of several seconds can sufficiently capture cycles in activities such as walking, running, climbing up stairs etc. Furthermore, a window size of 256 samples enabled fast computation of FFTs used for one of the features.

Four features were extracted from each of the three axes of the accelerometer, giving a total of twelve attributes. The features extracted were:

- Mean
- Standard Deviation
- Energy
- Correlation.

The usefulness of these features has been demonstrated in prior work (Bao & Intille 2004). The DC component of the signal over the window is the mean acceleration value. Standard deviation was used to capture the fact that the range of possible acceleration values differ for different activities such as walking, running etc.

The periodicity in the data is reflected in the frequency domain. To capture data periodicity, the energy feature was calculated. Energy is the sum of the squared discrete FFT component magnitudes of the signal. The sum was divided by the window length for normalization. If $x_1, x_2, \ldots$ are the FFT components of the window then, *Energy* = $\frac{\sum_{i=1}^{|w|} |x_i|^2}{|w|}$.

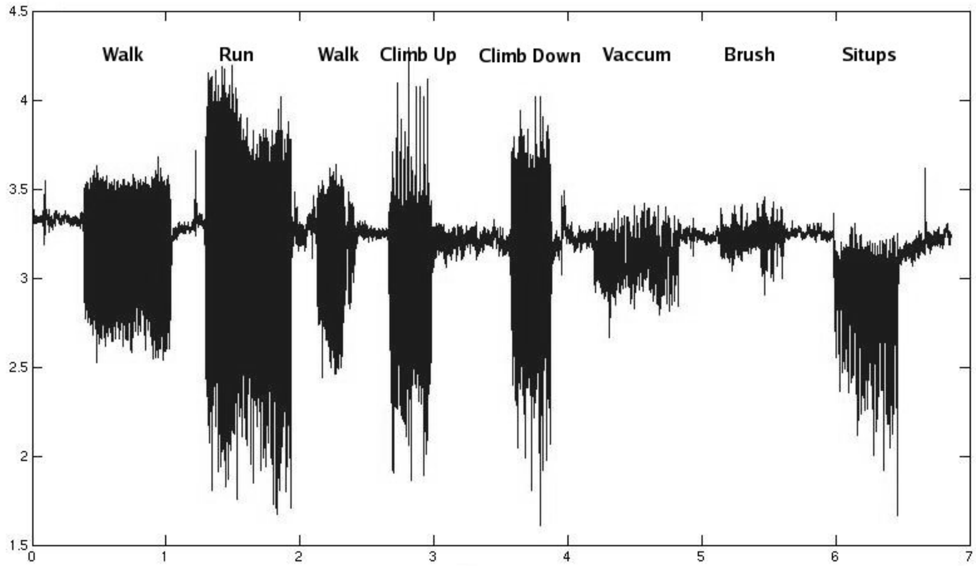


Figure 3: X-axis readings for different activities

Correlation is calculated between each pair of axes as the ratio of the covariance and the product of the standard deviations $corr(x, y) = \frac{cov(x,y)}{\sigma_x \sigma_y}$. Correlation is especially useful for differentiating among activities that involve translation in just one dimension. For example, we can differentiate walking and running from stair climbing using correlation. Walking and Running usually involve translation in one dimension whereas Climbing involves translation in more than one dimension.

Data Interpretation

The activity recognition algorithm should be able to recognize the accelerometer signal pattern corresponding to every activity. Figure 3 shows the x-axis readings for the different activities. It is easy to see that every activity does have a distinct pattern. We formulate activity recognition as a classification problem where classes correspond to activities and a test data instance is a set of acceleration values collected over a time interval and post-processed into a single instance of $\{$mean, standard deviation, energy, correlation$\}$. We evaluated the performance of the following base-level classifiers, available in the Weka toolkit:

- Decision Tables
- Decision Trees (C4.5)
- K-nearest neighbors
- SVM
- Naive Bayes.

We also evaluated the performance of some of the state-of-the-art meta-level classifiers. Although the overall performance of meta-level classifiers is known to be better than that of base-level classifiers, base-level-classifiers are known to outperform meta-level-classifiers on several data sets. One of the goals of this work was to find out if combining classifiers is indeed the right thing to do for activity recognition from accelerometer data, which to the best of our knowledge, has not been studied earlier.

Meta-level classifiers can be clustered into three frameworks: voting (used in bagging and boosting), stacking (Wolpert 1992; Dzeroski & Zenko 2004) and cascading (Gama & Brazdil 2000). In voting, each base-level classifier gives a vote for its prediction. The class receiving the most votes is the final prediction. In stacking, a learning algorithm is used to learn how to combine the predictions of the base-level classifiers. The induced meta-level classifier is then used to obtain the final prediction from the predictions of the base-level classifiers. The state-of-the-art methods in stacking are stacking with class probability distributions using Meta Decision Trees (MDTs) (Todorovski & Dzeroski 2003), stacking with class probability distributions using Ordinary Decision Trees (ODTs) (Todorovski & Dzeroski 2003) and stacking using multi-response linear regression (Seewald 2002). Cascading is an iterative process of combining classifiers: at each iteration, the training data set is extended with the predictions obtained in the previous iteration. Cascading in general gives sub-optimal results compared to the other two schemes.

To have a near exhaustive set of classifiers, we chose the following set of classifiers: Boosting, Bagging, Plurality Voting, Stacking with Ordinary-Decision trees (ODTs) and Stacking with Meta-Decision trees (MDTs).

- Boosting (Meir & Ratsch 2003) is used to improve the classification accuracy of any given base-level classifier.

Boosting applies a single learning algorithm repeatedly and combines the hypothesis learned each time (using voting), such that the final classification accuracy is improved. It does so by assigning a certain weight to each example in the training set, and then modifying the weight after each iteration depending on whether the example was correctly or incorrectly classified by the current hypothesis. Thus final hypothesis learned can be given as

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x),$$

where α_t denotes the coefficient with which the hypothesis h_t is combined. Both α_t and h_t are learned during the Boosting procedure. (Boosting is available in the Weka toolkit.)

- Bagging (Breiman 1996) is another simple meta-level classifier that uses just one base-level classifier at a time. It works by training each classifier on a random redistribution of the training set. Thus, each classifier's training set is generated by randomly drawing, with replacement, N instances from the original training set. Here N is the size of the original training set itself. Many of the original examples may be repeated in the resulting training set while others may be left out. The final bagged estimator, $h_{bag}(.)$ is the expected value of the prediction over each of the trained hypotheses. If $h_k(.)$ is the hypothesis learned for training sample k,

$$h_{bag}(.) = \frac{1}{M} \sum_{k=1}^{M} h_k(.).$$

- Plurality Voting selects the class that has been predicted by a majority of the base-level classifiers as the final predicted class. There is a refinement of the plurality vote algorithm for the case where class probability distributions are predicted by the base-level classifiers. In this case, the probability distribution vectors returned by the base-level classifiers are summed to obtain the class probability distribution of the meta-level voting classifier:

$$P_{ML}(x) = \frac{1}{|C|} \sum_{c \in C} P_c(x).$$

- Stacking with ODTs is a meta-level classifier that uses the results of the base-level classifiers to predict which class the given instance belongs to. The input to the ODTs are the outputs of the base-level classifiers i.e. class probability distributions (CPDs) — $p_{C_j}(c_i|x)$, as predicted over all possible class values c_i, by each of the base-level classifiers C_j. The output of the stacked ODT is the class-prediction for the given test instance.
- Stacking with MDTs (Todorovski & Dzeroski 2003) learns a meta-level decision tree whose leaves consist of each of the base level classifiers. Thus, instead of specifying which class the given test instance belongs to, as in a stacked ODT, an MDT specifies which classifier should be used to optimally classify the instance. The MDTs are also induced by a meta-level data set that consists of the CPDs — $p_{C_j}(c_i|x)$.

All the above meta-level classifiers, except MDTs, are available in the Weka toolkit. We downloaded the source code for MDTs and compiled it with Weka.

Alternate approaches to activity recognition include use of Hidden Markov Models(HMMs) or regression. HMMs would be useful in recognizing a sequence of activities to model human behavior. In this paper, we concentrate on recognizing a single activity. Regression is normally used when a real-valued output is desired, otherwise classification is a natural choice. Signal processing can be helpful in automatically extracting features from raw data. Signal processing, however, is computationally expensive and not very suitable for resource constrained and battery powered devices.

Results

All the base-level and meta-level classifiers mentioned above were run on data sets in four different settings:

Setting 1: Data collected for a single subject over different days, mixed together and cross-validated.

Setting 2: Data collected for multiple subjects over different days, mixed together and cross-validated.

Setting 3: Data collected for a single subject on one day used as training data, and data collected for the same subject on another day used as testing data.

Setting 4: Data collected for a subject for one day used as training data, and data collected on another subject on another day used as testing data.

Data for settings 1 and 2 is independently and identically distributed (IID), while that for settings 3 and 4 is not. Running classifiers on both IID and non-IID data is important for a thorough comparison.

We did a 10-fold cross-validation for each of the classifiers in each of the above settings. In a 10-fold cross-validation, the data is randomly divided into ten equal-sized pieces. Each piece is used as the test set with training done on remaining 90% of the data. The test results are then averaged over the ten cases.

Table 1 shows the classifier accuracies for the four settings respectively. It can be seen that Plurality Voting performs the best in the first three settings, and second best in the fourth setting. Boosted/Bagged Naive Bayes, SVM and kNN perform consistently well for the four settings. Boosted SVM outperforms the other classifiers by a good margin in the fourth setting. In general, meta-level classifiers perform better than base level classifiers.

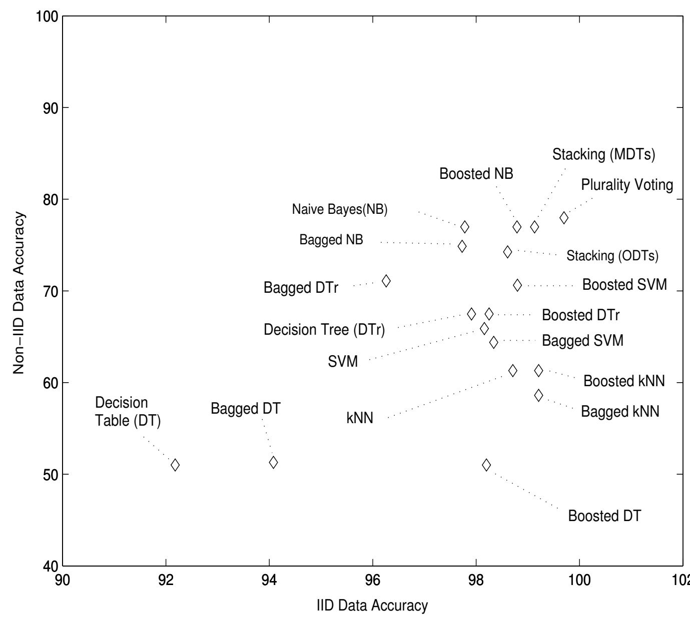
The scatter-plot in Figure 4 shows the correlation in the performance of each classifier on IID and non-IID data. Values on x-axis correspond to the accuracy of classifiers averaged over settings 1 and 2, while the values on y-axis correspond to the accuracy of classifiers averaged over settings 3 and 4. Plurality Voting has the best performance correlation (0.78).

Plurality voting combines multiple base-level classifiers as opposed to boosting and bagging which use a single

Table 1: Accuracy of classifiers for the four different settings

Classifier	Accuracy(%)			
	Setting1	Setting2	Setting3	Setting4
Naive Bayes(NB)	98.86	96.69	89.96	64.00
Boosted NB	98.86	98.71	89.96	64.00
Bagged NB	98.58	96.88	90.39	59.33
SVM	98.15	98.16	68.78	63.00
Boosted SVM	99.43	98.16	67.90	**73.33**
Bagged SVM	98.15	98.53	68.78	60.00
kNN	98.15	99.26	72.93	49.67
Boosted kNN	99.15	99.26	72.93	49.67
Bagged kNN	99.15	99.26	70.52	46.67
Decision Table(DT)	92.45	91.91	55.68	46.33
Boosted DT	97.86	98.53	55.68	46.33
Bagged DT	93.30	94.85	55.90	46.67
Decision Tree(DTr)	97.29	98.53	77.95	57.00
Boosted DTr	98.15	98.35	77.95	57.00
Bagged DTr	97.29	95.22	78.82	63.33
Plurality Voting	**99.57**	**99.82**	**90.61**	65.33
Stacking (MDTs)	99.00	99.26	89.96	64.00
Stacking (ODTs)	98.86	98.35	84.50	64.00

Figure 4: Performance correlation for IID and non-IID data



base-level classifier. Voting can therefore outperform boosting/bagging on certain datasets. From our results, it is clear that plurality voting does better than boosting and bagging consistently, although by a small margin. Plurality voting outperforming MDTs and ODTs is not very intuitive. A careful analysis however explains this finding. (Todorovski & Dzeroski 2003) showed that MDTs and ODTs usually outperform plurality voting on datasets where the error diversity of base-level classifiers is high. Plurality Voting on the other hand outperforms MDTs and ODTs on datasets where base-level classifiers have high error correlation (low error diversity), the cutoff being approximately 50%. The error correlation between a pair of classifiers is defined as the conditional probability that both classifiers make the same error given one of them makes an error:

$$\phi(C_i, C_j) = p(C_i(x) = C_j(x) | C_i(x) \neq c(x) \vee C_j(x) \neq c(x)),$$

where $C_i(x)$ and $C_j(x)$ are the predictions of classifiers C_i and C_j for a given instance x and $c(x)$ is the true class of x.

We calculated error correlation between all the base-level classifiers (which is defined as the average of pairwise error correlations) for all the four settings. The error correlation came out to approximately 52%. This high value of error correlation may explain why Plurality Voting does better than MDTs and ODTs on accelerometer data.

We wanted to find out which features/attributes among the selected ones are less important than the others. To this end, we ran the classifiers on the data with one attribute removed at a time. Table 3 shows the average number of misclassifications for data of setting 2, with one attribute dropped at a time. The Energy attribute turns out to be the least significant. There is no significant change in accuracy when Energy attribute is dropped. Since we could recognize activities with fairly high accuracy, we did not explore the possibility of adding more features/attributes.

In order to find out which activities are relatively harder to recognize, we manually analyzed the confusion matrices obtained for different data sets for different classifiers. The confusion matrix gives information about the actual and predicted classifications done by the classifiers. The confusion matrix in Table 2 is a representative of the commonly observed behavior in setting 3. It shows that climbing stairs up and down are hard to tell apart. Brushing is often confused with standing or vacuuming and is in general hard to recognize.

Conclusions and Future work

We found that activities can be recognized with fairly high accuracy using a single triaxial accelerometer. Activities that are limited to the movement of just hands or mouth (e.g brushing) are comparatively harder to recognize using a single accelerometer worn near the pelvic region. Using meta-classifiers is in general a good idea. In particular, combining classifiers using Plurality Voting turns out to be the best classifier for activity recognition from a single accelerometer, consistently outperforming stacking. We also found that energy is the least significant attribute.

An interesting extension would be to see whether "short activities" (e.g opening the door with a swipe card) can be recognized from accelerometer data. These could be instrumental in modeling user behavior. Along similar lines, it would be interesting to find out how effective an ontology of activities could be in helping classify hard-to-recognize activities.

Table 2: Representative Confusion Matrix for Setting 3

Activity	Classified As							
	Standing	Walking	Running	Stairs Up	Stairs Down	Vacuuming	Brushing	Situps
Standing	**63**	0	0	0	0	0	0	0
Walking	0	**44**	0	1	0	0	0	0
Running	0	0	**17**	16	20	0	0	0
Stairs Up	0	0	0	**9**	12	0	0	0
Stairs Down	0	0	0	19	**0**	0	0	0
Vacuuming	0	0	0	0	0	**45**	0	0
Brushing	18	0	0	0	0	15	**0**	0
Situps	0	0	0	0	0	7	0	**24**

Table 3: Effect of dropping an attribute on classification accuracy

Attribute	Average no. of misclassifications
Drop None	14.05
Drop Mean	21.83
Drop Standard Deviation	32.44
Drop Energy	**14.72**
Drop Correlation	28.38

Acknowledgments

Our sincere thanks to Amit Gaur and Muthu Muthukrishnan for lending us the accelerometer.

References

Bao, L., and Intille, S. S. 2004. Activity recognition from user-annotated acceleration data. In *Proceceedings of the 2nd International Conference on Pervasive Computing*, 1–17.

Breiman, L. 1996. Bagging predictors. *Machine Learning* 123–140.

Bussmann, J.; Martens, W.; Tulen, J.; Schasfoort, F.; van den Bergemons, H.; and H.J.Stam. 2001. Measuring daily behavior using ambulatory accelerometry: the activity monitor. *Behavior Research Methods, Instruments, and Computers* 349–356.

DeVaul, R., and Dunn, S. 2001. Real-Time Motion Classification for Wearable Computing Applications. Technical report, MIT Media Laboratory.

Dzeroski, S., and Zenko, B. 2004. Is combining classifiers with stacking better than selecting the best one? *Machine Learning* 255–273.

Foerster, F.; Smeja, M.; and Fahrenberg, J. 1999. Detection of posture and motion by accelerometry: a validation in ambulatory monitoring. *Computers in Human Behavior* 571–583.

Freund, Y., and Schapire, R. E. 1996. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, 148–156.

Gama, J., and Brazdil, P. 2000. Cascade generalization. *Machine Learning* 315–343.

Harter, A., and Hopper, A. 1994. A distributed location system for the active office. *IEEE Network* 8(1).

Lee, S., and K.Mase. 2002. Activity and location recognition using wearable sensors. *IEEE Pervasive Computing* 24–32.

Makikawa, M.; Kurata, S.; Higa, Y.; Araki, Y.; and Tokue, R. 2001. Ambulatory monitoring of behavior in daily life by accelerometers set at both-near-sides of the joint. In *Proceedings of MedInfo*, 840–843.

Meir, R., and Ratsch, G. 2003. An introduction to boosting and leveraging. 118–183.

Priyantha, N. B.; Chakraborty, A.; and Balakrishnan, H. 2000. The cricket location-support system. In *Mobile Computing and Networking*, 32–43.

Randell, C., and Muller, H. 2000. Context awareness by analysing accelerometer data. In MacIntyre, B., and Iannucci, B., eds., *The Fourth International Symposium on Wearable Computers*, 175–176. IEEE Computer Society.

Seewald, A. K. 2002. How to make stacking better and faster while also taking care of an unknown weakness. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 554–561. Morgan Kaufmann Publishers Inc.

Todorovski, L., and Dzeroski, S. 2003. Combining classifiers with meta decision trees. *Machine Learning* 223–249.

Want, R.; Hopper, A.; Falcao, V.; and Gibbons, J. 1992. The active badge location system. Technical Report 92.1, ORL, 24a Trumpington Street, Cambridge CB2 1QA.

Witten, I., and Frank, E. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kauffman.

Wolpert, D. H. 1992. Stacked generalization. *Neural Networks* 241–259.