

---

# A Practical Application of Body Sensor Data Classification Final Report

---

**Eugene E. Marinelli III**  
Department of Computer Science  
Carnegie Mellon University  
Pittsburgh, PA 15213  
emarinel@andrew.cmu.edu

## 1 Introduction

Personal health monitoring is an emerging technology which gives physicians the ability to monitor the activity of their patients throughout their daily lives, allowing them to make more informed diagnoses. Since these portable personal monitoring devices are worn constantly by their users, a large amount of data is generated. This wealth of data lends itself to analysis using machine learning techniques, which have the potential to detect health problems before they become serious and, more immediately, customize the feedback given to users of the device based on their age, gender, and current activity.

In this project, the ICML-2004 BodyMedia dataset, a collection of body sensor data collected using the BodyMedia SenseWear device, was used to investigate how to classify user properties based on sensor readings for both prior and live data.

### 1.1 Related Work

Several reports based on this data were submitted to the ICML-2004 Physiological Data Modeling Contest. [1] discusses a Bayesian network approach to diagnosing physiological conditions based on the BodyMedia dataset. [2] applies decision trees, neural networks, naïve Bayes, logistic regression, and SVM to classifying the data, giving an overview of the implementation and results of each method.

This project is related to the Healthnet project in the course 18-549 Embedded Systems Design which developed a wearable wireless multi-node body sensor system this semester [4].

## 2 Problem Definition

In this project, we aimed to answer the questions of how to train accurate body data classifiers, mapping body sensor readings to the user's age, gender, handedness, and current activity, and how to develop an online classifier that can actually be deployed in such an embedded system to provide customized feedback to the user. This included studying the effects of preprocessing the data in the offline setting.

## 2.1 Dataset

The BodyMedia dataset released for the ICML-2004 Physiological Data Modeling Contest was used for this project. It associates user characteristics like age, handedness, gender, and current activity with several sensor readings. Using this data, it is feasible to train supervised classifiers that guess these characteristics of the user using only the sensor data.

The user properties in the dataset were user ID, age, handedness, gender, and current activity. The sensor properties in the dataset were near body temperature, galvanic skin response, heat flux, pedometer, skin temperature, longitudinal accelerometer (SAD and average), and transverse accelerometer (SAD and average). Only these nine sensor values were used to classify user properties. The dataset includes data from 23 users, each with a different age ranging from 18 to 61. Each handedness and gender is represented, though not in globally accurate fractions.

## 3 Method

### 3.1 Offline

The first component of the implementation is the set of offline algorithms and validation functions. We define an “offline” machine learning algorithm to be one whose results do not change dynamically with newly available information. They use only the available data to build a classifier. This describes typical machine learning algorithms such as logistic regression, naïve Bayes, and  $k$ -nearest neighbors.

#### 3.1.1 Algorithms

Logistic regression, Gaussian Naïve Bayes, and  $k$ -nearest neighbors were implemented. These algorithms were chosen because they are supervised and can handle real-valued properties. Regular old naïve Bayes and decision trees, for example, would not be suitable for this application. Although it is possible to implement these such that an arbitrary number of classes can be mapped to, the implementations only support binary classification.

Gaussian Naïve Bayes involves estimating the parameters for the equation:

$$Y \leftarrow \underset{y_k}{\operatorname{argmax}} \frac{P(Y = y_k) \prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}$$

A Naïve Bayes classifier simply computes the value of the expression on the right for the given example for each possible class and returns the one which maximizes the expression. In the case of GNB, the parameters of Gaussian probability distributions, mean and variance/standard deviation, are optimized to fit the training data [5].

Logistic regression involves fitting a logistic function of the form:

$$P(Y = 1 | X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^n w_i X_i)}$$

to the data. This is done by finding the vector of weights  $W$  maximizing the expression:

$$\prod_l P(Y^l | X^l, W)$$

where each  $(X^l, Y^l)$  is an example in the training data. This weight vector is found using gradient descent, stepping each weight by

$$w_i \leftarrow w_i + \eta \sum_l X_i^l (Y^l - P(Y^l = 1 | X^l, W))$$

until convergence [5].

$k$ -nearest neighbors is a conceptually simpler supervised learning algorithm that consists of finding the  $k$  nearest neighbors of the example to be classified and returning the class that is most popular among those neighbors. Each ( $n$ -dimensional) example is taken to be a point in  $n$ -dimensional space, and some distance function between them is minimized. In this implementation the Euclidean distance was used, and no normalization or feature selection was performed.

### 3.1.2 Programming

These classifier-builder algorithms and their validation functions were implemented in OCaml because this language is statically type-checked and compiles to efficient machine code, giving it the ability to handle larger datasets in a smaller amount of time. It also features a convenient standard library.

It was later observed that a machine learning framework could be expressed concisely using a simple type structure and certain aspects of functional programming. The following types were used throughout my implementation:

```
type property = RealProperty of float | IntProperty of int
type class' = int
type unlabeled_example = property list
type example = property list * class'
type classifier = unlabeled_example -> class'
```

The classifier-building functions had the following types:

```
val build_gnb_classifier : example list -> classifier
val build_lr_classifier : float -> float -> example list
  -> classifier
```

The classifier type is simply an arrow (function) type, so the classifier-building functions return a function (“closure”) from an unlabeled example to a class instead of an implementation-specific result such as a list of weights in the case of logistic regression, or simply an accuracy which must be computed in the context of the classifier-builder (this turns out to be a convoluted mess). This type structure also allows for control over when computation actually takes place, be it eagerly when the classifier is built (which occurs by default in the case of logistic regression) or lazily deferred to classification time (which occurs in the case of naïve Bayes).

Using the type structure, a generic  $k$ -folds cross validation function could be declared as:

```
val kfold_validation_accuracy: int -> example list ->
  (example list -> classifier) -> float
```

and used by currying the specific parameters of the classifier-builder, for instance:

```
let gradient_step = 0.0000001 in
let lambda = 0.5 in
let k = 10 in
let lr_accuracy = kfold_validation_accuracy k examples
  (build_lr_classifier lambda gradient_step) in
  printf "Logistic regression accuracy: %f\n" lr_accuracy
```

Using functional programming for machine learning may also allow for easy parallelization of classification and building classifiers. This is currently being done in industry and ma-

chine learning research in a specific, somewhat limited way using the MapReduce paradigm [3]. Generalizing this sort of functional parallel computing for machine learning is an area of interest for my future research.

## **3.2 Online**

### **3.2.1 Algorithm**

A simple online algorithm was designed to test the basic feasibility of online learning for body sensor to user classification. First, offline logistic regression was performed to compute classification weights. These weights were loaded into the online classifier. As each new example was received, the online classifier counted the number of times each classification occurred and output the one that was most common (a simple voting scheme).

### **3.2.2 Programming**

The online classifier was implemented such that it can be run on an embedded system running Linux, such as the Gumstix platform. These systems are limited in processing speed, memory, and battery power compared to typical desktop computers and servers. The nature of the application, which involves reading in new examples and updating outputs accordingly, requires the program to be fast to meet input and output deadlines. To meet these performance requirements, the program was written in C because it is portable, fast, and memory-efficient.

## **4 Experiment**

### **4.1 Offline algorithms**

Since the full BodyMedia dataset is fairly large (about 71MB), subsets of the data, each consisting of 3000 examples sampled from the dataset in varying ways, were used to build and evaluate the classifiers. Only sensor values were used to build the classifiers. For each class, a subset was chosen such that each value of the class was equally represented to make the particular prior user property distributions of this dataset less significant. Age classification was between an 18 year old and a 61 year old. These subjects were chosen as a nice “proof-of-concept” for age classification. Activity classification was done between sleep and non-sleep since sleep was a very popular activity annotation, and most of the other activity annotations were not identified. The accuracies of the classifiers generated by each algorithm were estimated using 10-folds cross validation.

The amount of smoothing of the data was varied in order to determine the effect of smoothing data in order to reduce noise on classifier accuracy. A simple moving average was used with varying window sizes. Smoothing seems to make sense in this application because biological sensors are prone to noise.

### **4.2 Online algorithm**

The online classifier was evaluated by taking a set of at least 64 consecutive samples from each user. The correctness of the classifier was checked for 1, 4, 16, and 64 samples. In the case of the gender and handedness experiments, the input data was not constructed to balance gender or handedness, so accuracies above those observed in the offline experiments were expected.

## 5 Results

### 5.1 Offline algorithms

The following table summarizes the results of the experiments. The “smoothing” value is the window size of the moving average (i.e. the second parameter of MATLAB’s “smooth” function). The values under “gender”, “handedness”, “age”, and “activity” are the accuracies of these experiments (average fraction of correctly classified examples in 10-folds cross validation). The “age” column refers to the classification experiment between the 18 year-old’s data and the 61 year-old’s data. The “activity” column refers to the classification experiment between sleeping and non-sleeping. The baseline accuracy in each case is 0.5 because of the way in which each data subset was constructed. Question marks (?) refer to missing data resulting from mysterious software bugs.

	Parameters	Smoothing	Gender	Handedness	Age	Activity
LR	$\lambda = 0.5,$	0	0.633	0.525	0.889	0.847
LR	$\eta = 0.0000001$	5	0.642	0.536	0.923	0.878
LR		15	0.655	0.541	0.939	0.894
LR		50	0.676	0.552	?	0.907
GNB	None	0	0.639	0.550	0.914	0.793
GNB		5	0.639	0.550	0.901	0.823
GNB		15	0.655	0.549	0.912	0.859
GNB		50	0.671	0.558	0.935	0.890
$k$ -NN	$k = 10$	0	0.505	0.503	0.659	0.499
$k$ -NN		5	0.536	0.525	0.738	0.483
$k$ -NN		15	0.580	0.558	0.828	0.476
$k$ -NN		50	0.653	0.625	0.903	0.517

In general, kNN improved the most as a result of smoothing. LR and GNB also improved somewhat with smoothing, but not by as much.

In the gender experiment, LR and GNB performed about equally well. kNN did about as well as LR and GNB with a large smoothing window.

In the handedness experiment, LR and GNB did relatively poorly regardless of smoothing, but kNN improved significantly with smoothing, outperforming both LR and GNB with a large degree of smoothing.

In the age experiment, both GNB and LR did very well with over 0.9 accuracy. kNN did poorly for low smoothing but almost matched the performance of GNB and LR for high smoothing.

In the activity experiment, the accuracy of both GNB and LR improved significantly with smoothing, whereas the performance of kNN was very bad regardless of the smoothing level, hovering around the baseline of 0.5.

### 5.2 Online algorithm

The results of the online algorithm verification are summarized below. Two experiments were performed: one for classifying the user’s gender and one for classifying the user’s handedness. Each value is the accuracy of the classifier after a certain number of examples. Accuracy in this case is the number of correctly classified users divided by the total number of users (23).

Number of samples	Gender	Handedness
1	0.696	0.652
4	0.696	0.826
16	0.783	0.826
64	0.682	0.727

Increasing the number of examples read by the online classifier does not seem to help the classification accuracy. This is probably because the algorithm itself is fairly limited. Weighting the classifications or averaging examples together might yield better results.

## 6 Conclusions

Based on the results of this experiment, there seems to be potential for classifying any of the user properties that we attempted to classify, including gender, handedness, age, and distinctive activities such as sleeping vs. other activities based on sensor readings. Different algorithms performed differently for different classification tasks. LR and GNB generally performed well and about the same, but there were some cases where kNN outperformed them. In general, we can conclude that smoothing the sensor data has a positive effect on classification. Furthermore, considering how small the data subsets used in this experiment were compared to the full dataset, significant improvement could probably be seen by simply classifying using more training data.

Online classification basically worked, but there was no clear benefit from observing multiple sequential examples without updating the actual classification weights. Given more time, it would be worthwhile to come up with a way to update the online LR weights as new examples are received and try different weighting and averaging schemes for the live data.

Using functional programming for machine learning was discussed tangentially and may lead to further research in parallel computing for machine learning.

## 7 References

- [1] Kayalp, M. (2004) Bayesian Methods for Diagnosing Physiological Conditions of Human Subjects from Multivariate Times Series Sensor Data. *Physiological Data Modeling Contest*.
- [2] Andrews, S., Cai, L., Gondek, D., et al. (2004) Astrology: The Study of Astro Teller. *Physiological Data Modeling Contest*.
- [3] Dean, J., and Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters.
- [4] Marinelli, E., Bauman, J., Choi, K., and Goldhammer, A. (2008) Healthnet. <http://www.ece.cmu.edu/ece549/spring08/team6/index.html>.
- [5] Mitchell, T. (2005) Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression.