Probabilistic Time-Series Segmentation

Haik Kalantarian, Majid Sarrafzadeh

Abstract—Among the major challenges in the realization of practical health monitoring systems is the identification of short-duration events from larger signals. Time-series segmentation refers to the challenge of subdividing a continuous stream of data into discrete windows, which are individually processed using statistical classifiers to recognize various activities or events. In this paper, we propose a probabilistic algorithm for segmenting time-series signals, in which window boundaries are dynamically adjusted when the probability of correct classification is low. Our proposed scheme is benchmarked using an audio-based nutrition-monitoring case-study. Our evaluation shows that the algorithm improves the number of correctly classified instances from a baseline of 75% to 94% using the RandomForest classifier.

Index Terms—time-series segmentation, pervasive computing, signal processing, machine learning

1 INTRODUCTION

This paper addresses the issue of efficient time-series segmentation and classification: important topics in real-time embedded systems such as those used in wearable health and fitness monitoring devices which collect and process continuous sensor data. To understand why time-series segmentation is important requires a basic understanding of sensor systems and statistical classifiers. In this section, we begin with some of these preliminaries.

1.1 Sensing

Real-time wearable sensor systems have become very popular in recent years, and address a variety of health needs ranging from fitness, health monitoring, and object tracking. Such systems can range from a simple activity monitor such as the Jawbone and Misfit [1], [2] to more complex examples such as wearable diet-monitoring devices [3] [4]. These diet monitors, which are the motivating use case of our work, characterize eating habits from audio data through the analysis of chew and swallow sounds acquired from a wearable microphone.

Regardless of their specific function, wearable devices typically acquire signals from sensors such as accelerometers, gyroscopes, and microphones, as individuals go about their normal daily activities. Subsequently, various algorithms are used to identify actions of interest from the continuous stream of real-time data.

1.2 Classification

Various techniques have been proposed to identify activities of interest from time-series sensor data. One example of such a system was proposed by Alshurafa et al. in [5], in which the authors extract statistical features from sensor data, and determine the activity being performed using machine learning tools. However, in the case of a 24 hour stream of data it is not practical to assign a single class label to the whole dataset. Expectedly, the subject may be running for one hour, walking for one hour, and sitting for the rest of the day. Therefore, prior to classification, the data must be segmented into smaller windows that are each assigned a separate class label.

1.3 Segmentation

The efficient segmentation of time-series data requires that we select correct start indices and window sizes. Figure 1 shows various challenges associated with windowing a signal. In this figure, the vertical dashed lines represent the boundaries of our window, and the perturbations in the otherwise constant signal are the events of interest.
In Figure 1-A, a correct windowing approach is shown; the boundaries of the window are selected such that the event of interest is centered, with minimal margins on both sides. In Figure 1-B, the window boundaries are selected such that the event of interest is bisected. Thus, there is no single window which holistically contains the representative features of the event that we wish to detect. Figure 1-C, the window is improperly sized. This may be problematic because the distinguishing features of the event in question may be averaged together with other irrelevant properties that are associated with unrelated actions or events. Figure 1-D shows another case in which an empty portion of the signal is windowed.

We can improve our ability to segment a signal with the empirical observation that a higher degree of classification confidence is typically associated with correct windowing. Intuitively, this is because our notion of an 'ideal' window segment is based on our training data, which is generally manually annotated and postprocessed. That is, a classification model that reports 99% confidence that a window is associated with Class A suggests a high likelihood that we have selected boundaries that best preserve the distinguishing attributes of the event in question. Otherwise, the window size or indices may require adjustment.

1.4 Classification Confidence

Most classifiers assign distinct class labels based on a particular set of input features. However, for our purposes we are interested in knowing the probability that a classification is correct, rather than the class itself. The issue of deriving classification confidence scores for various classifiers has been explored in prior works. In [6], Platt et al. proposed a method for probabilistic classification in SVM (Support Vector Machines).

In this work, Platt et al. model the probability of a classification based on the distance between an instance and the hyperplane. The conclusion of this study was that instances further from the hyperplane were more likely to be correct classifications than those close to it. When the distance between the instance and the hyperplane is transformed via a sigmoid function and trained on a sample data set, this approach has been shown to be an effective [6] heuristic for classification confidence.

However, other classifiers do not have the concept of a hyperplane that separates the various classes of data. Thus, other heuristics are necessary to estimate classification confidence. In [7], Bostrom et al. describe various approaches to estimate the probability of classification using Random Forest classifiers. In the Random Forest algorithm, several bagged trees are created to classify the original dataset. The process of bagging involves sampling from the training set with replacement, and using a random set of features per tree [8]. The algorithm yields multiple trees, each of which predict an output class. A voting-based approach can be used to select a final class label. The specific classification probabilities can be inferred through the percentage of individual trees which predict a particular class label.

Though various other techniques for obtaining classification confidence scores have been proposed [9], [10], [11], we focus our analysis on the highly generalizable approach described by Langford et al. in [12]. We provide a description of this technique in Section 3.

1.5 Summary

We have briefly described the challenges of segmenting time-series data, as well as obtaining classification confidence scores from classifiers. The focus of our paper is on the application of confidence scores to the issue of window selection, for the application of detecting short-duration events from larger signals.

Figure 2 shows the general flow of our proposed segmentation algorithm. In Figure 2-A, we acquire discrete data from a wearable sensor with the objective of detecting some properties associated with it. In this case, the simplest example would be a peak detector algorithm, though a more fitting example could be to detect a cough or a sneeze from a 24-hour audio recording. Once the signal has been sampled, it would be divided into discrete windows based on some criteria as shown in Figure 2-B. This could involve dividing the signal into n non-overlapping windows with a fixed window size. Figure 2-C shows how the windows can be adjusted as a function of their classification confidence score. Those windows with a low probability of correct classification will be moved, or resized, to better match the training data.

The primary contributions of our paper are:

- The application of a technique for estimating classification confidence to the challenge of segmenting time-series data.
- The evaluation of the proposed algorithm to the problem of audio-based analysis of eating behavior from an internal dataset consisting of twenty subjects.

This paper is organized as follows.

- In Section 2, we provide a brief overview of existing time-series segmentation techniques.
- In Section 3, we describe the approach for obtaining classification confidence scores based on the technique proposed by Lanford et al. in [12] called probing.
- In Section 4, we describe our proposed algorithm for segmenting time-series data using classification confidence scores.
- In Section 5, we describe our experimental setup based on our prior dataset from [3], [13], [14].
- In Section 6, we present our experimental results.
- In Section 7, we analyze the performance of our scheme.
- In Section 8, we describe possibilities for future work.
- In Section 9, we provide concluding remarks.

2 Related Work

In this section, we provide a brief overview of related work in time-series segmentation and nutrition monitoring technologies.
In [21], Junker et al. propose a method for spotting gestures using inertial sensors in continuous activity monitoring. The author’s approach is a two-stage technique consisting of a cheap, sensitive preselection technique followed by a highly selective second-stage. The original data signal is partitioned into a series of non-overlapping windows and identified based on an understanding of human motion patterns.

### 2.2 Diet Monitoring

Many different techniques have been proposed for monitoring diet, ranging from traditional techniques such as manual record-keeping approaches [22], [23], [24], [25], to custom hardware solutions such as smart table surfaces and wearable cameras [26], [27], [28], [29], [30], [31]. We evaluate our proposed algorithm on an audio-based nutrition monitoring dataset based on our prior work [14].

Despite the potential privacy issues, [32] [33], microphone-based audio monitoring is among the most common technique in academic literature to monitor eating behavior. For example, the work in [34] uses acoustic data acquired from a small microphone placed near the bottom of the throat. Their system is coupled with a strain gauge placed near the ear. Other works suggest the use of throat microphones as a means of acquiring audio signals from throat and extracting swallowing sounds, for evaluation of dysphagia symptoms in seniors [35] [36]. Similarly, the work featured in [35] by Nagae et al. attempts to distinguish between swallowing, coughing, and vocalization using wavelet-transform analysis of audio data. In [37], Rahman et al. present BodyBeat: a robust system for detecting human sounds. A similar work is presented by Yatani et al. in [38].

In the work by Amft et al. in [39], authors analyze bite weight and classify food acoustically from an earpad-mounted sensor. In [40], the authors present a similar earpad-based sensor design to monitor chewing sounds. Food grouping analysis revealed three significant clusters of food: wet and loud, dry and loud, soft and quiet. Other studies use neural-network approaches [41] or Support Vector Machines (SVMs) for characterization of diet [42].

### 3 Probing: A Model for Classification Confidence

In this section, we provide an overview of the technique to estimate classification confidence, based largely on the algorithm described by Langford et al. in [12] called probing; the readers are encouraged to refer to their original work for a detailed discussion. Generally, in a binary classifier with classes $A$ and $B$, a class prediction $A$ for an instance $i$ indicates that:

$$\Pr(i \in A) > \Pr(i \in B) \quad (1)$$

In other words, the classifier predicts it is more likely that instance $i$ is associated with class $A$ than class $B$. Though this may not be the case, this is the prediction made by the classifier with the limited knowledge available at the time. That is, $\Pr$ refers to the algorithm’s classification confidence estimate rather than a true probability.
Next, we define parameter \( k \); the penalty for misclassifying an instance associated with Class A. The baseline of \( k \) is the \( k \)-1 case for the binary classification problem; the cost of misclassifying a Class A instance is equal to that of Class B. Subsequently, we modify the weights of the training samples such that incorrect classifications of Class A instances are \( k \) times costlier to classify incorrectly than those of class B. Thus, a classifier that labels instance \( i \) as class A suggests that:

\[
Pr(i \in A) > \frac{1}{1+k}
\]  

By evaluating different values of \( k \), we can find a value \( k' \) such that the criteria shown in Equation 3 is met.

\[
Pr(i \in A) = Pr(i \in B)
\]  

The condition in Equation 3 will be satisfied at the first value of \( k \) at which the classifier changes its predicted class label for instance \( i \) from class A to class B. We define this particular value as the classification confidence score associated with a particular instance. However, it is not computationally feasible to evaluate every value of \( k' \). Therefore, we select \( n \) candidates and evaluate them sequentially until a stopping criteria is met. More specifically, we discretize the possible values for \( k \) to \( n \) candidates, selecting this coefficient based on the specific classification problem, available computational resources, and required accuracy.

This algorithm is described in Algorithm 1. Note that initially, the value of \( k \) is set to 0. Because there will be no penalty for incorrectly classifying instances that belong to Class B, the output of the classifier will be Class A for all instances. Next, we increment the \( k \) coefficient by some interval \( \Delta k \). A smaller value of \( \Delta k \) may provide higher accuracy, at the expense of computational overhead; the value used in our experiments was 0.1. This process continues while \( k < 1 \), until the predicted class label changes from Class A to Class B at some point, \( k' \), that we refer to as the left margin.

Next, this process is repeated starting from a default class label of Class B and a high value of \( k \) designated as \( k_{\text{max}} \), which was twenty in our evaluation. The \( k \) coefficient is decremented until the instance class label changes to Class A while \( k > 1 \). This is referred to as the right margin. The function then returns \( \max(\text{RightMargin}, \frac{1}{\text{LeftMargin}}) \) as the final classification confidence estimate. In summary, the confidence score is decided by the boundary penalty at which the classifier outputs a different class label. While a higher confidence score is not equivalent to a higher probability of correct classification, it suggests a similarity to a specific class in the training set. Therefore, the confidence score is heuristically associated with higher confidence.

4 Algorithm

Given a series of data, we subdivide the data into \( n \) non-overlapping windows of a fixed size. For each window \( w \), using the previously outlined probing technique, we obtain a classification confidence score, \( Pr(w) \). We then compare the classification confidence score with a predefined threshold \( \beta \). If the condition in Equation 4 is satisfied, the window is flagged for possible adjustment. An alternative is to maximize the classification confidence score, regardless of the \( \beta \) parameter. While this may be a valid approach for some use-cases, this parameter is introduced in our algorithm to reduce the computational overhead of the proposed approach: an important consideration in lightweight embedded systems with battery constraints.

\[
Pr(w) < \beta
\]  

The remaining challenge is to adjust the flagged windows by searching for a maximum for \( Pr(w) \) within the boundaries of the search. Let us assume that \( I_w \) is the starting index associated with a particular window \( w \), which is the time at which we place the leftmost boundary. We can find a local maximum for the classification confidence as a function of starting indices, as shown by Equation 5 and Equation 6 below.

\[
\frac{\partial Pr(w)}{\partial I_w} = 0
\]
More specifically, we create temporary windows around each flagged window and evaluate its classification confidence using probing. We then select the temporary window as a candidate to replace the original if its classification confidence score is higher than the original. Algorithm 2 shows the procedure in more depth. Note that after the window is relocated, we must then adjust the starting index of the subsequent window to ensure that the entire signal is represented.

The process of window adjustment may involve moving the window boundaries forward or backward from their arbitrary default positions. This is decided by comparing the classification confidence score of the windows immediately around the current window. If neither window is associated with a higher classification confidence, the window is retained without change. Otherwise, the window with the higher classification confidence score is selected. If both windows are associated with a higher classification confidence score, the default case is to pick the successor rather than the predecessor. The segmentation algorithm operates incrementally beginning with the earliest data window, proceeding to the most recent samples over time. This incremental operation is necessary to support real-time operation, as data is processed in the order in which it is received. In the event that a window is shifted forward, all subsequent window boundaries are adjusted to ensure the windows do not overlap. This can result in a small gap of noise between the window being processed and the previous window, which is discarded. In the event that a window is moved backwards such that it overlaps with the previous window, we discard this sample. This is because the window size of our algorithm is selected such that it is extremely unlikely for two events of interest to be contained within a single window. In the context of our nutrition monitoring application, it is unlikely that two discrete swallow events are contained within a window that is several hundred milliseconds in length.

4.1 A Multilabel Classification Example

The proposed algorithm can be applied to multiclass problems with few additional changes. Figure 3 shows a complete example for a problem with three class labels. In Step I, we subdivide the original signal into fixed-sized class labels of arbitrary length and position, with a value of $k=2$; there is no additional penalty for incorrectly identifying one particular class. Note that the default case does not correspond with $k=1$, as in the earlier binary classification example. Next, based on the training dataset, a tentative class label will be assigned to each window. At this stage, the percentage of correctly classified instances is the baseline to which we compare the results of our proposed scheme.

The next step, shown in Figure 3-II, is to identify those signals that are most likely to be incorrectly classified based on our probing heuristic. Thus, certain windows will be assigned a new class label as we sweep the value of $k$ from $k=0$ to $k=2$ at intervals of $\Delta k$. Subsequently, this step is repeated by sweeping from the other direction; we begin with a high value of $k$ and sweep until $k = 2$, noting the point at which the class label changes. Using these heuristics, each window $w$ will be assigned a classification confidence score, $k_w$, that represents its classification confidence; those windows with $k_w$ less than or equal to our $\beta$ parameter are flagged for possible adjustment in the next stage of the algorithm.

In Figure 3-III, we have identified two windows with low confidence thresholds, such that $k_w$ is less than the value of $\beta$. Next, we create several windows around these original windows, shifted only slightly by some value $\Delta T$; in practice, a value between 10% and 50% of the default window length appears to be a reasonable compromise between accuracy and performance. Naturally, highest performance would be achieved by shifting the window by intervals of one sample while the classification confidence score increases monotonically. Once these additional windows are created, tentative class label assignments are made to them as shown in Figure 3-IV. These labels are assigned $k$-values of 1, corresponding with the base case as in 3-I.

In Figure 3-IV, we perform another probing operation for the new windows we have selected; starting from the base case of $k=0$, we increase $k$ from $k=0$ to $k=1$ at intervals of $\Delta k$. Each window may change class labels at some value $k$; this value is used to calculate the confidence score assigned to that particular window. A hypothetical result is shown in Figure 3-IV; for the window on the left, confidence scores for the windows that precede and succeed the original are 80%, 30%. By comparison, assume the confidence score of the original window value is $k = 50%$. In this case, the original window will be shifted to the location of the succeeding window, because its confidence score is higher than the current. This process will continue until the confidence score of the window is at a local maximum; windows on both sides have lower (or equal) scores. An example can be seen

Algorithm 2: Segmentation Algorithm

```plaintext
/* First, we divide the signal into N non-overlapping windows, each of size L. */
Window [] windows = Signal.Split();
/* We iterate through each window in the signal. */
foreach Window w in windows do
    /* Extract features to be used by classifier. */
    features = extractFeatures (w);
    /* Obtain class label and confidence score using the previously outlined probing technique. */
    (ProbScore) = ObtainProbability (w);
    /* The window is flagged for adjustment. */
    if ProbScore < $\beta$ then
        /* Search for a nearby window that has a local maximum confidence score. If current instance is the maximum, retain original window. */
        OptimizeIndex();
        /* Adjust starting index of next window. */
        adjustBoundaries ();
```

4.1 A Multilabel Classification Example

The proposed algorithm can be applied to multiclass problems with few additional changes. Figure 3 shows a complete example for a problem with three class labels. In Step I, we subdivide the original signal into fixed-sized class labels of arbitrary length and position, with a value of $k=2$; there is no additional penalty for incorrectly identifying one particular class. Note that the default case does not correspond with $k=1$, as in the earlier binary classification example. Next, based on the training dataset, a tentative class label will be assigned to each window. At this stage, the percentage of correctly classified instances is the baseline to which we compare the results of our proposed scheme.

The next step, shown in Figure 3-II, is to identify those signals that are most likely to be incorrectly classified based on our probing heuristic. Thus, certain windows will be assigned a new class label as we sweep the value of $k$ from $k=0$ to $k=2$ at intervals of $\Delta k$. Subsequently, this step is repeated by sweeping from the other direction; we begin with a high value of $k$ and sweep until $k = 2$, noting the point at which the class label changes. Using these heuristics, each window $w$ will be assigned a classification confidence score, $k_w$, that represents its classification confidence; those windows with $k_w$ less than or equal to our $\beta$ parameter are flagged for possible adjustment in the next stage of the algorithm.

In Figure 3-III, we have identified two windows with low confidence thresholds, such that $k_w$ is less than the value of $\beta$. Next, we create several windows around these original windows, shifted only slightly by some value $\Delta T$; in practice, a value between 10% and 50% of the default window length appears to be a reasonable compromise between accuracy and performance. Naturally, highest performance would be achieved by shifting the window by intervals of one sample while the classification confidence score increases monotonically. Once these additional windows are created, tentative class label assignments are made to them as shown in Figure 3-IV. These labels are assigned $k$-values of 1, corresponding with the base case as in 3-I.

In Figure 3-IV, we perform another probing operation for the new windows we have selected; starting from the base case of $k=0$, we increase $k$ from $k=0$ to $k=1$ at intervals of $\Delta k$. Each window may change class labels at some value $k$; this value is used to calculate the confidence score assigned to that particular window. A hypothetical result is shown in Figure 3-IV; for the window on the left, confidence scores for the windows that precede and succeed the original are 80%, 30%. By comparison, assume the confidence score of the original window value is $k = 50%$. In this case, the original window will be shifted to the location of the succeeding window, because its confidence score is higher than the current. This process will continue until the confidence score of the window is at a local maximum; windows on both sides have lower (or equal) scores. An example can be seen

Algorithm 2: Segmentation Algorithm

```plaintext
/* First, we divide the signal into N non-overlapping windows, each of size L. */
Window [] windows = Signal.Split();
/* We iterate through each window in the signal. */
foreach Window w in windows do
    /* Extract features to be used by classifier. */
    features = extractFeatures (w);
    /* Obtain class label and confidence score using the previously outlined probing technique. */
    (ProbScore) = ObtainProbability (w);
    /* The window is flagged for adjustment. */
    if ProbScore < $\beta$ then
        /* Search for a nearby window that has a local maximum confidence score. If current instance is the maximum, retain original window. */
        OptimizeIndex();
        /* After optimizing, obtain final class label. */
        (ClassLabel, ProbScore) = classify (features, DefaultK);
        /* Adjust starting index of next window. */
        adjustBoundaries ();
```
in the second window shown in Figure 3-IV. Note that the windows surrounding the second window both have classification confidence scores less than or equal to that of the original window. Therefore, no window is selected and the original is retained for the rest of the computation. The final results are shown in Figure 3-V; the succeeding window is elected in the first case, while the original window is retained in the second.

5 EXPERIMENTAL METHODOLOGY

Our proposed scheme is validated with two datasets from two different experiments. Experiment 1 is based on audio signals collected from a smartwatch, while Experiment 2 is based on signals from a throat microphone. In this section, we describe our experimental methodology.

5.1 Experiment 1 - Binary Classification

The first case study we use to evaluate our proposed scheme is a nutrition monitoring application. In this study, we obtained audio recordings from 10 individuals who ate three different foods while wearing a smartwatch device. From this data, we extract features and use various classifiers in an attempt to identify the correct food. The dataset used in this study is the same as that used in our prior work on smartwatch-based detection of eating habits [3], [43].

5.1.1 Data Collection

A total of ten subjects were used for data collection, with ages ranging from 22 to 35 in order to develop a model for identifying swallows. The subjects included 8 males and two females. Each subject was asked to eat several foods, among which were: three apple slices with at least two bites per slice, and one bag of potato chips. The moments at which the food was bitten into were manually annotated by the subject, though these events were clearly audible on the resulting waveform. The hand on which the smartwatch was worn was used to pick up the food items and water, which happened to be the left hand for all subjects.

Data collection took place in a laboratory environment which had a minimal level of background noise including talking and doors opening, most of which is barely audible in the recording. However, pre-recorded background noise from a public shopping square was combined with the original data, to produce clips that more accurately reflect a real-world use case. It was assumed that the background noise should be quieter than the original waveforms because in our experiments, the watch was inches away from the mouth at the time of the extracted audio clips. Regardless of the food or activity type, each sample was exactly 0.25 seconds in length, and the peak of the wave amplitude was not necessary centered in the window. In some cases, such as during the biting of an apple, one quarter of a second was not sufficient to capture the entire bite. Therefore, the relevant information was partially truncated.

5.1.2 Classifier Training

The training set consisted of 50 samples from the apple, and chips dataset. To ensure that the output data is not biased, separate training and test datasets were used with 50% of
instances in each category. The samples were selected to ensure that none of the training data was derived from the same subject as any instance of the test data. The WEKA [44] data mining software provided the implementations for the evaluated classifiers, and the work by Desai et al. was used for cost-sensitive classification [45]. Each audio sample was 0.25 seconds long; this parameter was fixed for the duration of the experimentation as it was specific to the use case outlined in our work. However, the windows were shifted by increments of 100 ms when the probability of correct classification varied.

5.1.3 Feature Extraction
The Munich open Speech and Music Interpretation by Large Space Extraction toolkit, known as openSMILE [46], is a feature extraction tool for producing large audio feature sets. This tool is capable of various audio signal processing operations such as applying window functions, FFT, FIR filterbanks, autocorrelation, and cepstrum. In addition to these techniques, openSMILE is capable of extracting various speech related features and statistical features. Audio-based features include frame energy, intensity, auditory spectra, zero crossing rate, and voice quality. After data is collected from a variety of subjects eating several foods, feature selection tools can be used to identify strong features that are accurate predictors of swallows and bites for various foods.

Table 1 shows the 10 features ranked according to their correlation with the desired classifier outcomes: their ability to distinguish between the two types of food. In this table, sma represents the simple moving average of a signal characteristic, while melspec refers to the mel-frequency cepstrum, which is a representation of the power spectrum of a sound on a non-linear scale. These features are produced using an InformationGain attribute evaluation scheme provided by the WEKA data mining software [44]. For a detailed explanation of these features and a more qualitative explanation of what they represent, we refer the reader to the openSMile documentation [46]; the specific setting used was the emo_large configuration.

5.2 Experiment 2 - Multilabel Classification
The second case-study in our evaluation is based on a different dataset— one that is described in our prior work [14] and a later extension of this work. A key difference between Experiment 1 and Experiment 2 is that the second experiment focuses on multiclass data, as we wish to see how the proposed algorithm scales in scenarios with non-binary class labels. Furthermore, while a smartwatch was used in Experiment 1, Experiment 2 is based on a throat microphone. The emphasis of the throat microphone is to detect bolus swallows that were manually annotated during the experiment. As the training data is based on manually annotated swallows, it can be inferred that classification confidence and selection will be closely correlated. In this experiment, we use three different kinds of food: chocolate, nuts, and a meat-like patty with the consistency of a hamburger. An additional difference between the two datasets was the default window size, which was set to one second rather than the quarter second in Experiment I; this allows us to evaluate our methodology on windows of different lengths.

5.2.1 Data Collection
Data was collected from 20 individuals using a Hyperio Flexible Throat Microphone Headset similar to the model shown in Figure 4. The microphone was placed in the lower part of the neck near the collarbone, and connected directly to the mobile phones audio input port using a 3.5mm male audio cable. 16 of the subjects were male, and 4 were female. The ages ranged from 21 to 31 years old, with a median age of 22. Commercially available audio-recording technology was used to acquire the audio recordings from the microphone. The subjects were given a small portion of nuts, chocolate, a vegetarian meat-substitute patty. The foods were consumed sequentially, in that order. Over 50 additional samples were manually extracted from this experiment. These recordings formed the basis of the algorithm design and experimental evaluation.

The data collection took place in a lab environment; people can be faintly heard speaking in the background, and the microphone occasionally recorded doors closing and nearby footsteps. In most audio classification works, ambient noises can interfere with the signal and decrease classification accuracy. This issue is partially rectified by placing the throat microphone in the lower part of the neck. This microphone placement emphasizes swallow sounds, as they are in much closer proximity to the device than ambient noises. Furthermore, most commercial throat microphones contain active circuitry for filtering out these ambient signals. These factors make throat microphones particularly well-suited for the task of recognizing eating behavior from chew and swallow sounds.

5.2.2 Classifier Training
In this experiment, our training set consisted of forty samples from each class, for a total of 120. Our test set consisted of 180 samples, with sixty from each category. Our total training set consisted of a total of 180 windows. No window in the test set was derived from the same individual as any waveform in the training set; these two datasets were kept independent to avoid biasing the results. However, cross-validation was not used as it was important to preserve the order in which audio signals were originally recorded; otherwise, unnatural discontinuities would be created between windows while the boundaries were shifted during the probing adjustment process.
In this section, we provide results as we apply our segmentation and classification algorithm to our two audio-based nutrition monitoring datasets. To gain some intuition into why the proposed algorithm is efficient for segmentation of audio data, we refer the reader to Figure 10. This shows how the probability score for each window varies for three audio recordings, each of which corresponded exclusively to one food type. Note that each waveform contains several spikes; times at which the classifier probability score varies. Our technique identifies those windows with a low $k$ score and uses information from the neighboring waveform to provide corrections if necessary. We begin by describing our results from Experiment I.

### 6 Results and Discussion

Table 2 shows the 10 features ranked according to their correlation with the desired classifier outcomes: their ability to distinguish between the two types of food. These features are produced using an InformationGain attribute evaluation scheme provided by the WEKA data mining software [44].

#### 5.2.3 Selected Features

Table 2 shows the 10 features ranked according to their correlation with the desired classifier outcomes: their ability to distinguish between the two types of food. These features are produced using an InformationGain attribute evaluation scheme provided by the WEKA data mining software [44].

### 6.1 Experiment I

Figure 7 shows how the percentage of instances classified as class $A$ varies as we sweep the value of $k$ from 0.1 to 10, for three classifiers: BayesNet, RandomForest, and SimpleLogistic. As expected, both extremes bias towards one particular class, as the objective function of the classifier is to minimize total cost. However, some instances change class labels at a lower threshold than others as shown in Figure 8. Instances whose class labels change at lower $k$ thresholds are considered to have a lower classification confidence, and are appropriate candidates for boundary optimization.

Figure 6 shows the percentage of incorrectly classified instances that were identified using our algorithm. Note that the ability to recognize incorrect instances is generally a function of the classifier used. As expected, the SimpleLogistic classifier (WEKA’s implementation of Simple Logistic Regression) shows poor results because of the simplicity of this technique and inability to produce an effective probability margin. By comparison, the RandomForest classifier produced the strongest results.

#### 6.1.1 Comparison to Baseline

Figure 5 shows the total classification results as a function of the $\beta$ parameter for three classifiers. Note that in some cases, such as the SimpleLogistic classifier, a higher value of $\beta$ does not lead to higher classification results. This is suggested by Figure 6, which demonstrates the inability of the algorithm to identify mislabeled instances using this classifier. The performance of RandomForest was improved by the widest margin, with 75% of instances correctly identified with $\beta = 1$ (baseline), and 94% correctly identified with $\beta = 2$. By comparison, the Bayesian Networks classifier showed more modest performance, with accuracy increasing from the baseline of 75% at $\beta = 1$, to a maximum of 83% at $\beta = 4$.

#### 6.1.2 Selection of $\beta$

In this subsection, we describe the selection of the $\beta$ threshold. Setting the $\beta$ threshold can improve accuracy, at the cost...
of performance. As Algorithm 2 shows, the $\beta$ parameter is the threshold at which a window is analyzed. Thus, a high value of $\beta$ will attempt to modify each window in the signal, regardless of its estimated classification confidence. Therefore, the optimal value of $\beta$ is a function of the ability of the classifier to identify incorrectly labeled instances. As the results in Figures 6 and 5 show, the SimpleLogistic classifier does not benefit from a high value of $\beta$ because the true probability of classification does not appear to be correlated with the probing heuristic. By comparison, the RandomForest classifier benefits most from a higher value of $\beta$ because of the applicability of the heuristic to the challenge of identifying misclassifications. In this experiment, best results were achieved with a value of $\beta = 4$, with no noticeable improvement observed at higher values of $\beta$.

6.2 Experiment II

Figure 8 shows how the number of correctly classified instances varied as the k weight for each class was adjusted. Note that the number of instances correctly classified decreases only slightly for the nuts class, compared to the much sharper dropoff of chocolate and meat. As expected, chocolate had the highest classification accuracy both at the baseline and after optimization; even when the penalty for misclassifications was raised sixfold, the majority of chocolate instances were assigned the chocolate label. This confirms our intuition that the probing technique is a useful heuristic for classification confidence.

6.2.1 Comparison to Baseline

Figure 9 shows the results of our algorithm as we sweep through various values of $\beta$. The base case is defined by $\beta=1$. In this scenario, the time-series signal is divided into equally sized windows and classified on a per-window basis. However, the classification confidence score of each window is unused; the original class label is retained and no probabilistic segmentation is applied to the signal. Therefore, the $\beta=1$ case corresponds with a naive baseline with which to compare the classification accuracy of our algorithm. In this scenario, 128 out of 180 instances were correctly classified. At a value of $\beta = 10$, the highest number of instances were classified correctly (147).

7 PERFORMANCE AND SCALABILITY

In this section, we analyze the computational overhead of the proposed scheme based on a theoretical model as well as empirical observations.

Fig. 10. This figure shows the variations in probability score from three long samples of data in which subjects ate a single food item.
7.1 Cost analysis
First, we define the cost of processing a window of data as $C_w$. This cost entails extracting features, running the attributes through a classifier, and assigning a class label. We define the new cost, $C_{new}$, as a function of the original cost in Equation 11. In this formula, $C_{probe}$ refers to the cost of determining the probability score for one particular window. The $\alpha$ coefficient refers to the percentage of windows whose probability score is less than the $\beta$ threshold. Lastly, $C_{adjust}$ refers to the cost of adjusting a particular window's location.

$$C_{new} = \alpha \cdot C_w \cdot C_{probe} \cdot C_{adjust} \quad (7)$$

7.1.1 Analysis of $C_w$
The cost of classifying a particular window, with no optimizations, is roughly equivalent to the sum of the cost of extracting the features and running the classifier.

$$C_w \approx C_{feat} + C_{class} \quad (8)$$

7.1.2 Analysis of $C_{probe}$
The cost of probing one particular window is a factor of $\Delta k$; the step size as we search for the point at which the class label changes. This gives us:

$$C_{probe} \approx \frac{C_w}{\Delta k} \quad (9)$$

7.1.3 Analysis of $C_{adjust}$
The cost of adjusting a window can be defined as a function of gamma, $\gamma$; the number of alternate windows attempted per original window that is flagged for adjustment. For $\gamma$ new windows, the probing analysis $C_{probe}$ must be evaluated to obtain a classification confidence score. The formulation is therefore:

$$C_{adjust} \approx \gamma \cdot C_{probe} = \gamma \cdot \frac{C_w}{\Delta k} \quad (10)$$

7.1.4 Summary
In summary, the algorithm’s total cost can be defined as follows:

$$C_{new} = \alpha \cdot \gamma \cdot \frac{(C_w)^3}{(\Delta k)^2} \quad (11)$$

7.2 Power characterization
Our prior work in [47] uses the same audio dataset and an MSP430-based development board to characterize the power consumption of various feature extraction algorithms. We refer readers to this work for a detailed power evaluation of the audio-based diet monitoring algorithm used in this paper. The same platform was used to characterize the performance tradeoffs of various classification algorithms on the same dataset in [48]. Using the 16-point FFT as a baseline to characterize the power overhead of extracting a frequency-domain feature, these results suggest a mean power of 0.62 mW for 128 FFT operations per second. Readers can refer to [48] for a discussion of the relative power and performance tradeoffs associated with $C_{feat}$ and $C_{class}$ in the context of computation offloading.

8 Future Challenges
Experimental results show that this technique for time-series segmentation is able to improve classification confidence compared to the baseline. However, it should be noted that segmentation may become much more challenging in systems with multiple class labels. In the future it would be necessary to evaluate the proposed scheme on scenarios in which there are a dozen or more possible class labels.

Moreover, systems with multiple class labels may require different window sizes for efficient recognition of different activities. Having to optimize both the window location and size would become a quadratic time problem that would not scale well.

9 Conclusion
In this paper, we describe how a model for classification confidence can be applied to the issue of segmenting long-duration time-series data for the purposes of classifying short duration events. Our results are benchmarked using two audio-based nutrition-monitoring case studies in which we identify various food types from audio data recorded from a smartwatch microphone. Results confirm the efficacy of our approach; future work will validate the proposed method to other forms of data.

Haik Kalantarian is a PhD candidate in the Department of Computer Science at the University of California, Los Angeles. His primary research interests are in the domain of real-time embedded systems, mobile computing, pervasive systems, and applied machine-learning.

Majid Sarrafzadeh is a Distinguished Professor of Computer Science at the University of California, Los Angeles. His research interests are in Healthcare technology, computer system architecture, embedded systems, VLSI CAD, algorithms, and biomedical informatics.
REFERENCES


