Microprocessors and Microsystems 000 (2016) 1–9

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Probabilistic segmentation of time-series audio signals using Support Vector Machines

Haik Kalantarian[a,*], Bobak Mortazavi[b], Mohammad Pourhomayoun[a], Nabil Alshurafa[c], Majid Sarrafzadeh[a]

[a] Dept. of Computer Science, University of California, Los Angeles, United States
[b] School of Medicine, Yale University, United States
[c] Dept. of Preventative Medicine, Northwestern University, United States

ARTICLE INFO

Article history:
Received 29 November 2015
Revised 24 March 2016
Accepted 28 April 2016
Available online xxx

Keywords:
Segmentation
Time-series
Support vector machines
Wearable devices

ABSTRACT

To allow health tracking, patient monitoring, and provide timely user interventions, sensor signals from body sensor networks need to be processed in real-time. Time subdivisions of the sensor signals are extracted and fed into a supervised learning algorithm, such as Support Vector Machines (SVM), to learn a model capable of distinguishing different class labels. However, selecting a short-duration window from the continuous data stream is a significant challenge, and the window may not be properly centered around the activity of interest. In this work, we address the issue of window selection from a continuous data stream, using an optimized SVM-based probability model. To evaluate the effectiveness of our approach, we apply our algorithm to audio signals acquired from a wearable nutrition-monitoring necklace. Our optimized algorithm is capable of correctly classifying 86.1% of instances, compared to a baseline of 73% which segments the time-series data with fixed-size non-overlapping windows, and an exhaustive-search approach with an accuracy of 92.6%.[1]

1. Introduction

Sensing and monitoring technologies have become increasingly popular in recent years, motivated by advances in wearable technology and a rapidly aging global population [1]. The applications of wearable technologies range from monitoring cardiac activity, eating habits, blood pressure, physical activity, brain activity, speech patterns, and more [2–4]. These devices use integrated micro-sized wearable sensors, which provide data that can be analyzed to track user activity. Recently, many works have employed machine learning and data mining techniques to identify health-related events from digital signals [5–7]. These devices and technologies have the potential for significant impact in real-word environments, by enabling *preventative healthcare*. However, the detection of finite events in a large stream of data is not without its challenges. In many cases, sensor data analysis can be quite simple. For example, determining heart rate from heart rate sensors could in some cases require little more than amplification and thresholding [8]. In other cases, such as continuous audio-based monitoring of eating habits, sleep apnea, and coughing, much more advanced techniques are required to discriminate between activities.

Machine learning tools based on classifiers such as Support Vector Machines (SVMs) are often used to transform data to find an optimal boundary between class labels [9]. Such tools enable the development of models to identify the subtle and interrelated conditions needed to distinguish between class labels. However, identifying the intended activity and extracting data points for classification from a continuous time-series signal is quite challenging. Some existing techniques identify peaks and troughs in the signal to identify each data sample, while others perform fixed time subdivisions of the data [10]. The division of a large continuous signal into individually-processed fragments is known as *time-series segmentation*, which is the focus of our work.

This paper is organized as follows. In Section 2, we describe the motivation for our work. In Section 3, we present a brief introduction to the challenges of time-series segmentation. In Section 4, we present related work in this field. In Section 5, we provide an overview of Support Vector Machines and the notion of classifier

* Corresponding author.
E-mail addresses: kalantarian@cs.ucla.edu (H. Kalantarian), bobak.mortazavi@yale.edu (B. Mortazavi), mpourhoma@cs.ucla.edu (M. Pourhomayoun), nabil@northwestern.edu (N. Alshurafa), majid@cs.ucla.edu (M. Sarrafzadeh).

[1] This work is funded by the National Science Foundation AIR Option 1: Award Number 1312310.

http://dx.doi.org/10.1016/j.micpro.2016.04.011
0141-9331/© 2016 Elsevier B.V. All rights reserved.

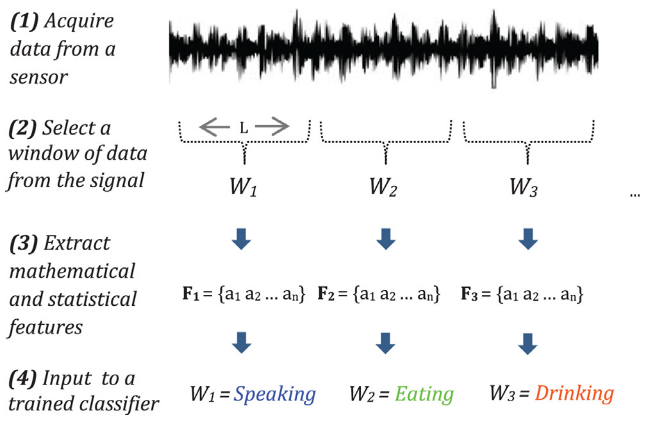
(1) Acquire data from a sensor
(2) Select a window of data from the signal
L
W1
W2
W3
...
(3) Extract mathematical and statistical features
F1 = {a1 a2 ... an}
F2 = {a1 a2 ... an}
F3 = {a1 a2 ... an}
(4) Input to a trained classifier
W1 = Speaking
W2 = Eating
W3 = Drinking

Fig. 1. This figure shows a typical system flow for classifying sensor data. Note that the constant window size with fixed boundaries may not be ideal in order to holistically represent a particular activity in a single window.

confidence. In Section 6, we describe our proposed window selection algorithm. In Section 7, we describe the experimental setup. In Section 8, we present our results, with concluding remarks in Section 9.

2. Motivation

Consider a 24-h stream of audio data from which we intend to identify eating behavior that we must disambiguate from other sounds. A typical system flow is defined as follows. First, audio is recorded using a microphone that is placed somewhere on the body. Examples include a smartwatch [11], smartphone, or a custom hardware device such as a neckband [12]. After audio is recorded, it is either buffered or transmitted to an aggregator for analysis. However, assigning a single class label to an entire days worth of data is impractical for many cases, which attempt to identify brief actions throughout the day. Therefore, the large audio stream is divided into shorter windows, in order to identify activities with a finer granularity and avoid averaging out unique statistical features associated with finite-length events such as coughing, chewing, and swallowing.

After the audio signals are divided into shorter windows, statistical features can be extracted from the window. Examples include mean, interquartile range, standard deviation, variance, kurtosis, and skewness. These features are processed using a pre-trained classifier, which can identify the action being performed based on the distinguishing characteristics of the signal represented by the features selected by the training process. This system flow is shown in Fig. 1. At the top, we have an audio waveform that is several minutes in duration. In order to identify multiple short-duration events from this signal, we extract windows W_1, W_2, and W_3 from the data. We then extract statistical features from each window, and input these features to a trained classifier, which outputs a class label corresponding with that window. In this case, our classifier reports that the user is speaking at times $\tau = 0$ through $\tau = 4$, followed by four minutes of eating, and four minutes of drinking.

Fig. 1 presents an example of a working system with ideal segmentation, but consider a case in which W_1 contains both data corresponding with speaking and eating, with roughly the same proportion. In this case, no class label can holistically represent the contained data. Rather than introducing a new class label corresponding with this particular combination of events, the window size and boundaries could be adjusted to properly contain the events of interest. In the next section, we describe the shortcomings of arbitrary window selection in more detail.

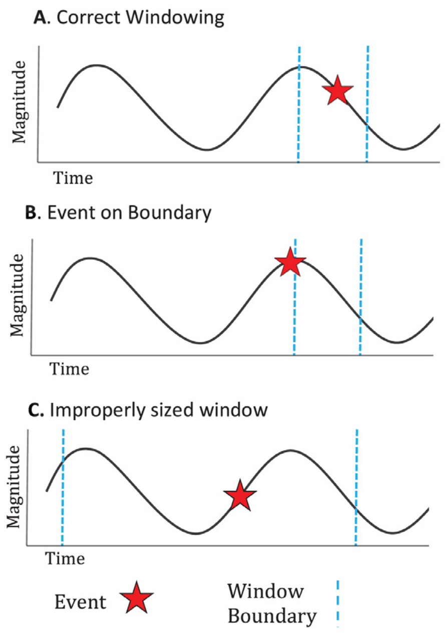

Fig. 2. This figure shows potential problems when attempting to identify an action from a large discrete signal using a windowing approach. The star represents the event of interest, which will be difficult to detect if the window boundaries are not selected appropriately.

3. System challenges and considerations

Most signal classification approaches in academic literature extract windows that are manually centered on the action of interest, with a window size large enough to fully contain the distinguishing features of the signal. Thus, the system has a-priori knowledge about the action being performed. Fig. 1 shows an approach in which the system has no a-priori knowledge about the signal, which is a more realistic scenario for real-world environments. However, there are several significant challenges associated with this approach, which are shown in Fig. 2. In this figure, the sinusoid represents the raw signal, while the red star denotes the short-duration event that we wish to detect. The dashed line represents the boundaries of the extracted window from the original signal. The major challenges are described below:

1) *Window Size Selection*: First, we may fail to select an appropriate window size, as shown in Fig. 2C. Consider an audio-based classification system designed to identify an activity of short duration such as a cough. If the system window size is ten minutes long, the audio features associated with the cough will be averaged together with the rest of the data in the window. Therefore, certain statistical features, such as those which evaluate power spectral density at different frequency ranges, may be rendered ineffective. Furthermore, a window of large length will be assigned a single class label, while multiple actions of interest could have taken place in this time period. Alternatively, a small window size may fail to holistically represent the event in the extracted snippet.
2) *Window Boundary Selection*: We must appropriately place the boundaries of the window. Consider the case in which the first half of the cough is found in window W_1, while the second is in window W_2. An example can be seen in Fig. 2B, in which only a portion of the event of interest is represented in the current window. Therefore, no window may have enough information for a correct classification. However, the classifier must still output a class label for the data, and the accuracy of the system will decrease. A potential solution is to overlap the windows,

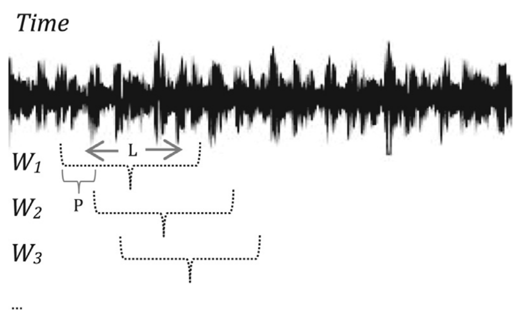

Fig. 3. This figure shows how windows can be overlapped, to avoid boundary issues. This comes at the expense of computational and power overhead. Establishing a separate window for each sample point is prohibitive for an embedded application, with data acquired at 16 kHz or higher.

as shown in Fig. 3. However, this approach is associated with high computational overhead; if we define$L$ as the window size, this approach requires a factor of L more time to complete. An embedded application may not have the power budget or processing power to justify this approach, considering some signals such as audio have sample rates in the kHz range.

3) *Binary Output*: Our classification system does not return a probabilistic output. A class label is assigned to each window, regardless of the classifier confidence that this label is correct. In some cases, it may be preferable to avoid assignment of a class label unless the classification confidence exceeds a predefined threshold. The end user of the classification model can subsequently choose to retain the class label or discard the sample, based on the specific properties of their application.

The key question we aim to address is: *How can we correctly window a continuous signal, in order to maximize classification accuracy of short-duration events?* Fortunately, various algorithms have been proposed for mapping a raw classifier output to a probability, or classification confidence [13,14]. In this paper, we propose a new architecture that can be used for real-time monitoring of sensor signals such as audio, by optimizing window selection based on the classification confidence for that particular instance.

4. Related work

A comprehensive survey of time-series segmentation was provided by Keogh et al. in [15]. The emphasis of existing work on segmentation is to develop a compact representation of a large signal, with applications in data mining and compression. The three primary methods in literature are sliding window approaches, similar to the baseline used in this paper, and recursive top-down or bottom-up techniques that are either partition or merge signals until a stopping criteria is met [15].

Analysis of digital signals, particularly audio signals, has been a subject of many recent works [16,17]. In [17], Rahman et al. present BodyBeat: a robust system for detecting human sounds. A similar work is presented by Yatani et al. in [18]. The BodyScope device can distinguish between twelve kinds of activities including eating, drinking, laughing, and coughing. This device is based on a microphone worn on the neck. By analyzing the magnitude of different frequency ranges over time, the authors are able to recognize various activities using classifiers such as Support Vector Machines [9]. However, the window size from which they extract features is of fixed length, at five seconds. This window size is arbitrary, as some actions are of significantly shorter duration, while others are more protracted. Furthermore, the activities in question were manually annotated, and the audio fragments were extracted by hand.

Adaptive window sizing has been explored in several other works. For example, in [19], Okutomi et al. present a signal matching algorithm that can adapt window size based on the uncertainty of the disparity between two signals. The issue of adaptive window sizing for image filtering using local polynomial approximation is presented by Katkovnik et al. in [20]. In [21], Klinkenberg et al. propose an approach for selection of window size to minimize generalization error based on leave-one-out estimation error. However, to the best of our knowledge, applying posterior probability of a classifier to the issue of window selection has not been explored in academic literature.

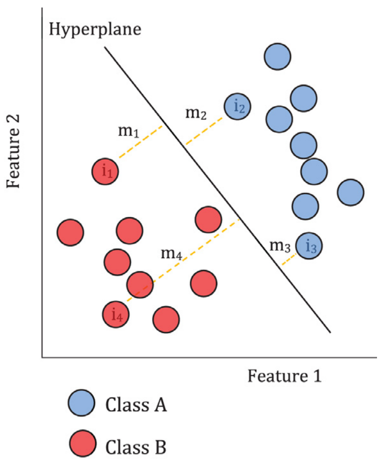

Fig. 4. This figure shows how an SVM hyperplane can separate instances belong to Class A and Class B. However, note that the distance between an instance and the hyperplane is not factored into the final class assignment.

5. Posterior probabilities for SVM

5.1. An introduction to Support Vector Machines

Support Vector Machines (SVMs) are supervised learning models commonly used for classification problems. Given a set of data, the SVM will attempt to separate the two classes using a hyperplane, which is a subspace one dimension less than the ambient space. Various mathematical techniques such as dual quadratic programming and non-linear kernel transformations can be used to maximize the margin between the classes of data. A *hard-margin* SVM will enforce the principle that the data must be separable. However, it may not be possible to achieve perfect separation of the data in some cases. Therefore, a *soft-margin* SVM classifier can be used to maximize the hyperplane margin while allowing some misclassifications.

5.2. Motivation

Consider a system designed for distinguishing between two activities: A and B. Fig. 4 illustrates a simple case in which an SVM model is used to linearly separate the two classes of data using a one dimensional hyperplane in a two-dimensional space. Thus, all 18 instances are labeled as class A or class B, with no distinction made about their original location on the feature plane. An *SVM Score* for an instance i is defined as the distance from i to the decision boundary. Clearly, instance i_4 has a very large SVM Score, m_4, as seen by its distance from the hyperplane separating the two classes. In comparison, note that the margin separating instance i_3 from the hyperplane is quite small. It has been shown that the probability that a classification is correct is a function of the SVM score for that particular instance, m [13]. Therefore, this observation suggests some ambiguity about the true class label associated with i_3. This can be a result of several factors:

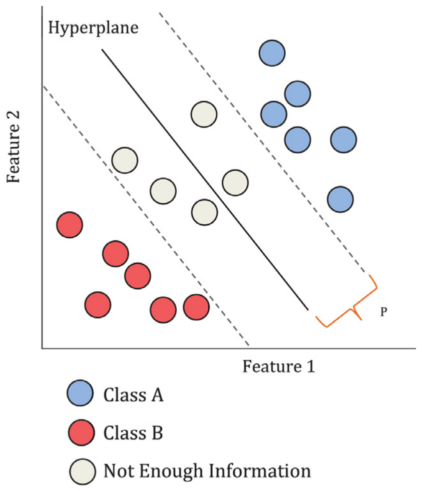

Fig. 5. This figure shows the proposed scheme, in which instances close to the hyperplane are not assigned class labels, and the window parameters are modified to potentially increase the instance margin and improve accuracy.

- The window size may be too small to effectively capture the distinguishing feature of the activity.
- The window size may be too large, polluting the unique properties of the short-duration activity with neighboring noise, or encompassing multiple events within the boundaries of a single window.
- The window boundaries may be bisecting the activity of interest into two separate windows, neither of which can definitively identify the event.
- The sample may not be associated with activities represented by either class. However, a class label will be assigned regardless, because a real time activity monitoring device cannot practically have thousands of class templates for each possible action a user could take throughout the day.
- The window selection is optimal, but the activity has unique properties that are dissimilar to other instances in that class.

We can potentially improve classification accuracy by considering the distance between an instance and the hyperplane to adjust the window index and size. That is, we are no longer exclusively concerned with the specific class an instance is associated with, but with the probability that the instance belongs to that particular class. Our motivation is the observation that window boundaries can be adjusted until the instance's SVM score increases beyond our desired threshold. As shown in Fig. 5, we assign class labels to those instances which are likely to be correct classifications. Those instances associated with lower probabilities, based on their distance from the hyperplane, are labeled as not enough information. In these cases, we refrain from assigning a classification output, and perform minor modifications to the associated window in order to disambiguate the class label. In the event that we are unable to improve the classification confidence score, we tag the data as ambiguous, leaving to the discretion of the user the decision of whether to discard the sample or assign a label.

5.3. Probability model

In the previous section, we referred to the probability that an instance is classified correctly as a function of the distance between the instance and the hyperplane. However, the relationship between distance and probability is not necessarily linear [22]. The work by Platt and Lin et al. in [13,14] explored the probabilistic outputs of classifiers and concluded that the relationship between SVM score and posterior probability assumes a sigmoid distribution. Platt Calibration is a well-known method for transforming classification model outputs to probability distributions. This technique, though generalizable, is often used in the context of Support Vector Machines. Platt Calibration can be used to compute the posterior probability of a class label given an instance x, using the following formulation:

$$P(y = 1|x) = \frac{1}{1 + e^{\alpha \cdot f(x) + \beta}} \tag{1}$$

In this case, *f(x)* is the raw, uncalibrated classifier output, the sign of which is used for binary class assignment. Constants α and β are coefficients obtained using a maximum likelihood estimation. These values are obtained by fitting training data to a sigmoid function, as the sigmoid model has been shown to be generally superior to other models such as Gaussian distributions for probability estimation. Typically, the sigmoid is trained using a subset of the training data (roughly 30%) [13] to ensure an unbiased training set. Thus, the Platt Calibration algorithm returns a posterior probability of classification accuracy for each instance, based on the signed distance between the instance and the decision boundary. Therefore, we can use the Platt Calibration algorithm to produce a classification confidence score for a particular instance, and adjust the window parameters to disambiguate the true label. This procedure is outlined in the following section. For further details about Platt Calibration, please refer to Platt et al. [13].

6. Window size selection

Each window is a function of two parameters: the starting index at which the window boundary is placed, as well as the length of the window. Thus, we define a window as a function of these parameters, $w = (I, L)$. Function f uses a trained SVM classifier as a well as a calibrated Platt model to return a class label c and a classification confidence p, based on an input window defined by the length and starting index, I and L, respectively.

$$[c, p] = f(I, L) \tag{2}$$

The classification confidence is a number between 0 and 1, which reflects the estimated probability that the instance is labeled correctly. Thus, in a binary classification problem, p ranges from .50 to 1. Based on this formulation, we compare three algorithms for window selection.

6.1. Naive window selection

Naive Window Selection is the simplest model that we use as a baseline. This is the scheme shown in Fig. 1, which is computationally simple, and does not use probability in the assignment of class labels. In this model, we simply subdivide the signal into non-overlapping windows and assign class labels to each window as described in the introduction. We use a fixed window length, and the default starting index, both of which are a function of the particular classification problem and cannot be generalized. This very simple scheme is shown in Algorithm 1 . Though classification accuracy is likely to be quite low using this approach, the simplicity of this model is likely to execute quickly because the algorithm

Algorithm 1: Naive Window Selection

```
1 /* We use the default window length and index*/
2 (c,p) = f (I, L);
3 /* Regardless of the probability of correct classification, we
  return the class label. */
4 return (c, p)
5 /* The next index will be $I_{new}$ = I + L */
```

is not iterative in the case of a particular window; the time complexity is $O(\frac{n}{L})$ where n is the number of samples and L is the window size.

6.2. Exhaustive search

In the Exhaustive Search approach, we emphasize classification accuracy over speed, by attempting every combination of window index and size to maximize the Platt classification probability. First, we define vectors **W** and **S** as the possible window sizes and starting indices, respectively, while α represents the smallest possible window length. The value of α is a function of the classification problem. That is, we have a-priori knowledge about the general properties of the signals we are attempting to classify. Therefore, we set our default window bounds to a value that is suitable for detecting the event of interest. The primary criteria used to select this value is if the event in question, if bounded correctly, can be holistically represented within a single window. In our particular application, we believed that 500 ms was the smallest possible reasonable time window. Therefore, the range of possible time windows varied between 500 ms and 2.5 s, as our n parameter was 5. Recall that n represents the number of different window lengths to attempt before selecting the length that yields the highest classification confidence.

The offset of each element index is i, which is the endpoint of the previous window. The possible start indices range from $i + \beta$ to $i + m \cdot \beta$. Therefore, β represents the range of possible start indices. As in the case of α, the value of β is a function of the classification problem based on a-priori knowledge about the general properties of the signal of interest. The intention of this parameter is to shift the window slightly, to prevent cases in which the window boundary is dividing the action of interest into two separate windows. Given that the goal in our particular application is to detect a chew or swallow, or perhaps a combination of both, any more than a few seconds of offset is unnecessary. Therefore, the value of β in our experimentation was fixed to 200 ms with an m value of 5.

Parameters n and m represents the number of possible starting indexes and window lengths to iterate over. The complete algorithm is shown in Algorithm 2 . Note that this algorithm performs an exhaustive search of all combinations of W and S, and selects the window length and starting index based on the combination with the maximum classification probability, using Platt's probability model. The number of iterations, T, is defined in Eq. (3) as the product of the lengths of vectors **W** and **S**.

There is no particular threshold at which the search concludes: the algorithm will iterate through every combination of window length and index until arriving at the optimal solution. Therefore, even if the original classification decision is associated with a posterior probability of over 99%, all other options will be considered. This inefficiency is addressed in the next subsection. The number of iterations necessary for the Exhaustive Search algorithm to terminate is shown below, in Eq. (3). Thus, the exhaustive search will take approximately $n{*}$m more time to complete, compared to the Naive Window Selection model.

$$T = n * m \tag{3}$$

6.3. Optimized search

The Optimized Search algorithm selects window parameters based on classification probability, without performing an expensive search across the entire feature space <I, L>. This may be necessary due to system constraints, as an Exhaustive Search approach could be impractical for wearable systems with battery constraints and basic low-power processors. The Optimized Search algorithm pseudocode is presented in Algorithm 3 .

Algorithm 2: Exhaustive Search Algorithm

```
1  /* Array W represents the possible window lengths. */
2  W = {$\alpha$, 2·$\alpha$, 3·$\alpha$, 4·$\alpha$ ... n · $\alpha$}
3  /* Array S represents starting indices from the window. i
   represents the final window boundary index from the previous
   window. */
4  S = {i+$\beta$, i+2·$\beta$, i+3·$\beta$, ... i+m · $\beta$}
5  maxProbability = 0;
6  /* We iterate through all combinations. */
7  for i = 0; i < n; i++
8      for j = 0; j < m); j++
9          /* We receive a class label and class probability from
           function f, for this particular combination of window
           length and index.*/
10         (c,p) = f (W[i], S[j]);
11         if p > maxProbability
12             /* This combination of window size and window
               length yields the best estimated probability of correct
               classification. */
13             maxProbability = p;
14             ClassFinal = c;
15             BestLength = W[i];
16             BestIndex = S[j];
17         end
18     end
19 end
20 /* The search is complete. We return the output class and the
   corresponding probability. */
21 return (ClassFinal, maxProbability)
```

We begin with line 2, in which we assume a large stream of audio data as a prerequisite for the search. First, we define array **W**, which represents the possible window lengths. Next, we define array **S**, which is required to present possible starting times. Lines 6 and 7 show that the algorithm selects default values of WindowLength and StartingPoint as 2.5 s and τ = i + 200 ms, respectively. Subsequently, we extract a window with those parameters for L and I, extract statistical features, and classify using SVM. The classifier results, shown on line 9, are the predicted output class and classifier confidence derived from the Platt model. In line 14, the classification confidence is compared to a predefined threshold, γ. If the margin exceeds the threshold, set to 0.8 in our experimentation for its relative balance between accuracy and efficiency, the classification result is retained and the search terminates. However, if the classification confidence is low, we evaluate the sensitivity of the classifier probability to each dimension, and proceed along the direction of the maximum gradient in the positive direction. The gradient of function **f** is shown below, in which the unit vectors in both dimensions, length and index, are represented by **x** = <1,0> and **y** = <1,0>.

$$\nabla f = \frac{\partial f(x,y)}{\partial x}\mathbf{x} + \frac{\partial f(x,y)}{\partial y}\mathbf{y}$$

We then calculate the directional derivative $\forall$ **u** $\in$ **K**, where **K** is defined as a set consisting of the following vectors:

$$\mathbf{K} = \{<1,0>, <0,1>, <1,1>\}$$

The motivation for the selection of these particular vectors is the initial condition of the algorithm, which begins at the smallest possible index and vector size. If this were not the case, directional derivatives would be calculated in the positive and negative directions. Also, note that these vector values are array indices, rather than absolute values. Thus, applications which require a finer

Algorithm 3: Optimized Search Algorithm

```
/* Array W represents the possible window lengths. */
W = {$\alpha$, 2·$\alpha$, 3·$\alpha$, 4·$\alpha$ ... n · $\alpha$}
/* Array S represents starting indices from the window. i
   represents the final window boundary index from the previous
   window. */
S = {i+$\beta$, i+2·$\beta$, i+3·$\beta$, ... i+m · $\beta$}
/* We define the initial conditions */
wi = W[1];
si = S[1];
/* We receive a class label and class probability from function f,
   for this particular combination of window length and index. */
(c,p) = f (W[wi], S[si]);
if $p > \gamma$ then
    /* If the probability of correct classification is greater than a
       threshold, we perform no further search, and return the
       current class label and probability.*/
    return (c, p);
maxProb = 0;
while maxProb < $\gamma$ do
    /* We evaluate if any neighboring set of parameters return a
       higher classification probability. */
    (c2, p2) = f (W[wi+1], S[si]);
    (c3, p3) = f (W[wi], S[si+1]);
    (c4, p4) = f (W[wi+1], S[si+1]);
    (maxProb, bestI, bestL, bestClass) =
    max([c2, p2], [c3, p3], [c4, p4]);
    /* We evaluate the directional derivative of f using three
       vectors */
    if maxProb < p then
        /* If neither yielded an improvement, we return the
           original class and probability value. */
        return (c, p);
    else
        /* Otherwise, continue iterating until we hit a local
           maximum or exceed $\gamma$. For the next iteration of the
           while loop, wi and si are set to the values of the
           neighboring set of parameters.*/
        wi = bestL;
        si = bestI;
return (bestClass, maxProb);
```

granularity of search could modify the values of variables α, β, m and n, as defined in Algorithms 2 and 3.

After computing the directional derivatives, we select vector **u** with the highest value, and continue iterating while two constraints are met:

$$\alpha < x < n \cdot \alpha \quad (4)$$

$$\beta < y < m \cdot \beta \quad (5)$$

These conditions ensure that parameters α and β remain within the valid range. However, if we find that the gradient value ∇f is negative for $\forall \mathbf{u} \in \mathbf{K}$, the search terminates, as we have reached a local maximum. This case is shown in Eq. (6).

$$\nexists \mathbf{u} \in \mathbf{K} \;\mid\; \nabla f \cdot \mathbf{u} > 0 \quad (6)$$

Lastly, if we find x and y such that f(x,y) $\geq \gamma$, the search will terminate even if Eq. (6) is satisfied, as the value of p is thought to have reached a point of diminishing returns at this threshold, if selected appropriately. Note that the possibility of an infinite loop condition is negligible, given that the classification confidence of

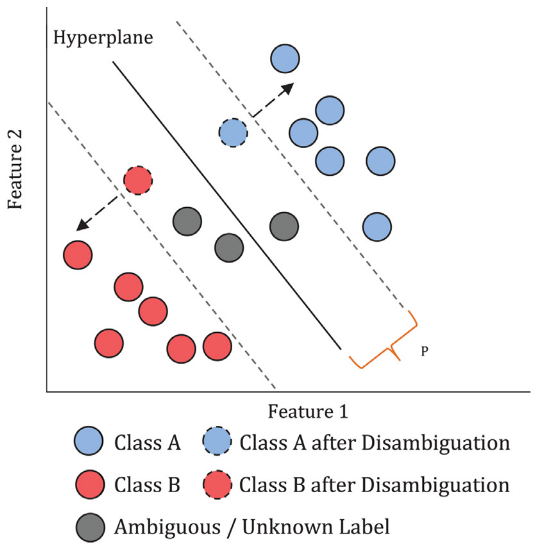
Hyperplane
Feature 2
Feature 1
P
Class A
Class A after Disambiguation
Class B
Class B after Disambiguation
Ambiguous / Unknown Label

Fig. 6. This figure shows how some instances originally labeled as ambiguous due to their distance from the hyperplane, were later assigned a class label by adjusting the parameters of their associated window until a desired confidence was reached.

a continuous real-time signal cannot increase indefinitely in real-world conditions. However, in cases in which this is a concern, a “time to live” value can be included to ensure that the search terminates after a certain number of iterations. After the algorithm is completed, we have a final class label c and probability p. If p is low, it is likely that the instance is not a complete representation of an activity.

A possible outcome is shown in Fig. 6. Note that several instances that were originally labeled as ambiguous are excluded from the list of false positives because adjustments to the window associated with these instances disambiguated their class label. However, in the case of several other instances, no modifications to the window were able to provide sufficient confidence of the node’s true class label. Note that it is possible for a node to drift from one side of the hyperplane to another during the iterative process. A simplified representation of the overall algorithm flow for the optimized search scheme is visually shown in Fig. 7, which illustrates how boundaries are adjusted based on the probability score of the original window.

7. Experimental setup

Data was collected from 20 individuals using a Hyperio Flexible Throat Microphone Headset placed in the lower part of the neck near the collarbone, connected directly to the mobile phones audio input port using a 3.5 mm 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 at a sample rate of 44 KHz. Twenty subjects, who were given two miniature chocolate bars, followed by ten Pringles potato chips. The foods were consumed sequentially, in that order, and the subjects ate one potato chip at a time. 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.

The application we have in mind is assessment of diet. Many works have employed microphones for detecting food intake. For example, the work in [16] 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. Similar examples

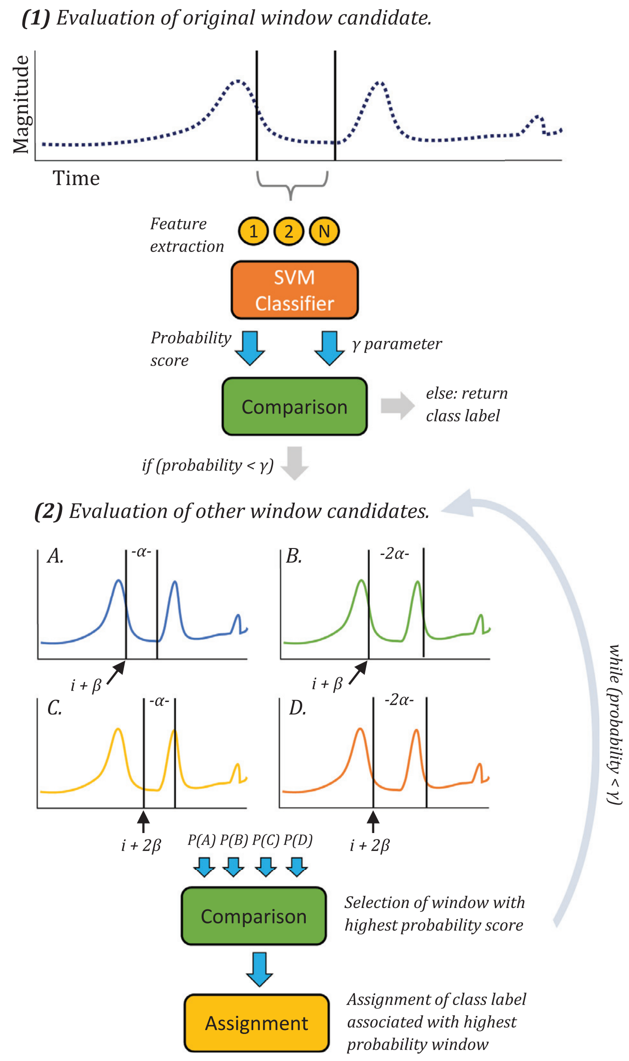

Fig. 7. This figure shows a simplification of the optimized search algorithm for finding an appropriate width and starting index for a particular window.

include [17] by Rahman et al. as well as the work by Yatani et al. in [18].

Other related 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 [23,24].

7.1. Feature extraction

The Munich open Speech and Music Interpretation by Large Space Extraction toolkit, known as openSMILE [25], is a feature extraction tool intended 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, while reducing the dimensionality by eliminating redundant or weakly correlated features. These feature reduction tools are applied to the test data, and evaluated on the training dataset.

Table 1
Partial list of selected features.

Rank	Feature Name
1	Log Energy: Skewness
2	Log Energy: Mean Distance Between Peaks
3	Log Energy: Zero Crossings
4	Mel-Freq: Simple Moving Average[0] Quartile 3
5	Mel-Freq: Simple Moving Average[0] Mean Dist. Between Peaks
6	Mel-Freq: Simple Moving Average[1] Quartile 2
7	Mel-Freq: Simple Moving Average[1] Mean Dist. Between Peaks
8	Mel-Freq: Simple Moving Average[0] Zero Crossings

From the 6555 extracted features, the Correlation Feature Selection (CFS) Subset Evaluator was used to evaluate 991,139 subsets of features. This is necessary to select the features best associated with the desired classifier outcomes. This subset evaluator considers both the individual predictive ability of features, as well as the redundancy between them. The top eight features are listed in Table 1. Mel-Freq features refer to those features associated with the mel-frequency cepstrum: a representation of the power spectrum of sound based on a logarithmic scale.

7.2. Model development

The dataset contained 40 audio clips from 20 subjects as they ate a variety of foods. Each audio clip corresponded with several minutes of data as each subject ate a particular type of food in a lab environment. The length of each clip differed from one subject to another as they ate at various speeds. A custom Java program extracted fixed-size windows from audio clips based on a given input window size (default of 500 ms), and exported these windows as separate WAV files that were processed individually using the openSMILE feature extraction toolkit.

The SVM kernel used in our evaluation was a radial basis function (RBF), which is used to map the original data to a higher dimension such that it is linearly separable by the hyperplane. The optimization technique used for the quadratic programming analysis necessary to generate the best-fit hyperplane was Sequential Minimal Optimization (SMO), which divides the original problem into smallest-subset problems that can be solved analytically. A grid-search approach was used for SVM hyperparameter selection. The implementation of these algorithms was provided by the WEKA Data Mining software, which was used for model development and evaluation [26]. The performance of the WEKA implementation was sufficient for our application as training was done offline and the feature set was quite small. However, for other systems with online training components, more efficient implementations such as Lagrangian Support Vector Machines (LSVM) may be used [27].

In our experimentation, no labels were discarded; all instances were retained to allow an objective comparison between the Optimized Search, Exhaustive Search, and Baseline algorithms.

8. Results

8.1. Classification results

Presented results were based on a total of 1000 data samples split evenly between each class, with 500 samples of chocolate and 500 samples of chips. 10-fold cross validation was used for algorithm evaluation. In other words, 90% samples were used for training and 10% for test for ten iterations, with a different test subset used for each iteration. The results were then averaged together for statistical convergence. The probabilistic model was derived from the same data as the training class, and was not developed using the full data set to prevent overfitting.

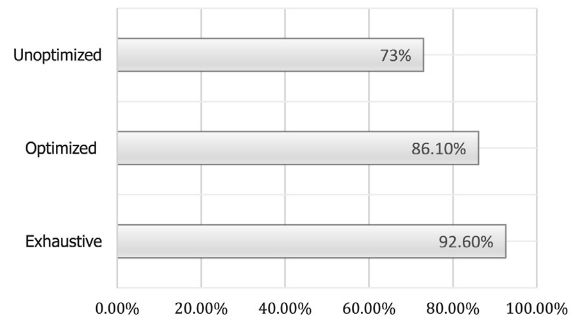

Fig. 8. This figure shows the percentage of correctly classified instances using a baseline with no optimizations, an exhaustive search of window sizes and indexes, and an optimized search.

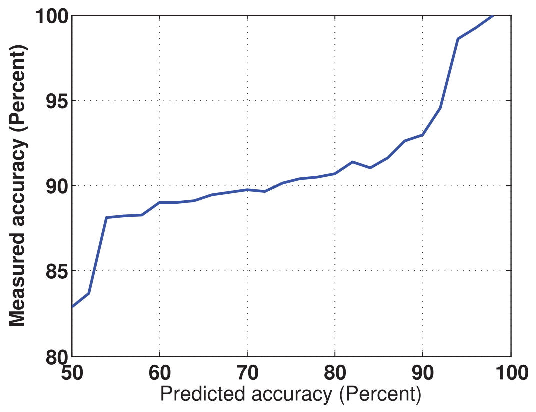

Fig. 9. In this figure, we validate the trained probability model which, as shown, has a sigmoid distribution. The x-axis represents the threshold set for all instances, with respect to reported classification confidence. The y-axis shows that, for instances with that classification confidence (and above), what the true classification accuracy was in our evaluation.

The classification results between the two food groups using the LibSVM classifier library are shown in Fig. 8. Three results are shown in this figure based on the three evaluated algorithms: naive, exhaustive, and optimized. The naive approach was used as a baseline to evaluate classification accuracy without optimizations. Note that the relatively low classification accuracy is because the window selection was completely at random, and it is likely that there was no eating in some samples at all. The samples were generated at random from a larger data stream, in order to replicate a realistic use-case. With no optimizations, only 73% of instances were classified correctly. Using the optimized scheme, the percentage of correctly classified instances rose to 86.1%. However, as expected, the highest performance was achieved using the exhaustive search technique, in which 92.6% of instances were correctly classified.

In our experiments, the values of n and m were set to five. Recall that these values correspond with the number of possible audio clips generated from each original clip, with n different window lengths and m different starting indexes per length. The window sizes ranged from 500 ms to 2.5 s, in increments of 500 ms. The starting indexes varied by increments of 200 ms, with a maximum of 1 s based on the m value of 5. The values for n and m chosen in our evaluation are a function of our particular classification problem; any more than a few seconds of window adjustment could re-center the window around a different activity. In other applications with longer duration events, or larger intervals of time between the activities of interest, these values may not apply.

8.2. Probability model validation

Fig. 9 shows a comparison between predicted and measured classification accuracy. Recall that the training set was used to create a sigmoid-based model for mapping the SVM score to a probability. This model was then applied to the test set, to evaluate how the posterior probability correlated with the reported probability. This figure was generated by listing all the test instances, along with their predicted classification accuracy using Platt Correlation. The predicted accuracy was then swept between 50%, the lowest possible accuracy for a binary dataset, and 100%. For each predicted accuracy value (x-axis), the y-axis represents the percentage of instances with at least that high prediction confidence, which were correctly classified. For example, consider the point *(70,90)*. This can be interpreted such that, for all instances which had a predicted classification confidence of at least 70% based on the sigmoid model developed by the training set, 90% were classified correctly. Generally speaking, there were some discrepancies between the probability model and the observation, particularly at lower prediction accuracies. However, the correlation between predicted accuracy and measured accuracy is still sufficient for window size selection.

9. Conclusion

In this work, we present a window-selection algorithm for identification of short-term events in continuous data streams, which we validate in an audio-classification testcase. By applying Platt Calibration to a Support Vector Machines (SVM) classifier, we are able to identify those instances most likely to be classified incorrectly. Through computationally efficient modifications of the data window, we are able to improve classification accuracy by identifying those instances whose class labels are most uncertain. This technique can be applied to any real-time audio classification application such as voice recognition, swallow detection, or home security, in which there is a sparsity in the signal that makes a sliding-window approach too computationally expensive. Future work will apply this probabilistic segmentation approach to other sensor signals such as inertial and piezoelectric sensors.

References

[1] W. Lutz, W. Sanderson, S. Scherbov, The coming acceleration of global population ageing, Nature 451 (7179) (2008) 716–719.
[2] Misfit wearables faq, (http://www.misfitwearables.com/supportl).
[3] C.-T. Lin, L.-W. Ko, M.-H. Chang, J.-R. Duann, J.-Y. Chen, T.-P. Su, T.-P. Jung, Review of wireless and wearable electroencephalogram systems and brain-computer interfaces–a mini-review, Gerontology 56 (1) (2010) 112–119.
[4] E.M. Tapia, S.S. Intille, W. Haskell, K. Larson, J. Wright, A. King, R. Friedman, Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor, in: Wearable Computers, 2007 11th IEEE International Symposium on, IEEE, 2007, pp. 37–40.
[5] R. Palaniappan, D.P. Mandic, Biometrics from brain electrical activity: a machine learning approach, Pattern Anal. Mach. Intell. IEEE Trans. 29 (4) (2007) 738–742.
[6] F.-T. Sun, C. Kuo, H.-T. Cheng, S. Buthpitiya, P. Collins, M. Griss, Activity-aware mental stress detection using physiological sensors, in: Mobile computing, applications, and services, Springer, 2012, pp. 211–230.
[7] H. Kalantarian, S. Lee, A. Mishra, H. Ghasemzadeh, J. Liu, M. Sarrafzadeh, Multimodal energy expenditure calculation for pervasive health: A data fusion model using wearable sensors, in: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on, 2013, pp. 676–681, doi:10.1109/PerComW.2013.6529578.
[8] S.J. Strath, D.R. Bassett Jr, D.L. Thompson, A.M. Swartz, Validity of the simultaneous heart rate-motion sensor technique for measuring energy expenditure., Med. Sci. Sports Exerc. 34 (5) (2002) 888–894.

[9] M.A. Hearst, S.T. Dumais, E. Osman, J. Platt, B. Scholkopf, Support vector machines, Intell. Syst. Appl. IEEE 13 (4) (1998) 18–28.
[10] A. Parate, M.-C. Chiu, C. Chadowitz, D. Ganesan, E. Kalogerakis, Risq: Recognizing smoking gestures with inertial sensors on a wristband, in: Proceedings of the 12th annual international conference on Mobile systems, applications, and services, ACM, 2014, pp. 149–161.
[11] H. Kalantarian, M. Sarrafzadeh, A smartwatch-based system for audio-based monitoring of dietary habits, in: The Fifth International Conference on Ambient Computing, Applications, Services and Technologies, 2015.
[12] H. Kalantarian, N. Alshurafa, M. Sarrafzadeh, A wearable nutrition monitoring system, IEEE Body Sensor Networks, 2014.
[13] J.C. Platt, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in: Advances in Large Margin Classifiers, MIT Press, 1999, pp. 61–74.
[14] H.-T. Lin, C.-J. Lin, R.C. Weng, A note on platts probabilistic outputs for support vector machines, Mach. Learn. 68 (3) (2007) 267–276.
[15] E. Keogh, S. Chu, D. Hart, M. Pazzani, Segmenting time series: a survey and novel approach, Data Min. Time Ser. Databases 57 (2004) 1–22.
[16] E. Sazonov, S. Schuckers, P. Lopez-Meyer, O. Makeyev, N. Sazonova, E.L. Melanson, M. Neuman, Non-invasive monitoring of chewing and swallowing for objective quantification of ingestive behavior, Physiol. Meas. 29 (5) (2008) 525.
[17] T. Rahman, A.T. Adams, M. Zhang, E. Cherry, B. Zhou, H. Peng, T. Choudhury, Bodybeat: A mobile system for sensing non-speech body sounds, in: Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, in: MobiSys '14, ACM, New York, NY, USA, 2014, pp. 2–13, doi:10.1145/2594368.2594386.
[18] K. Yatani, K.N. Truong, Bodyscope: a wearable acoustic sensor for activity recognition, in: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, ACM, 2012, pp. 341–350.
[19] M. Okutomi, T. Kanade, A locally adaptive window for signal matching, Int. J. Comput. Vis. 7 (2) (1992) 143–162, doi:10.1007/BF00128133.
[20] V. Katkovnik, K. Egiazarian, J. Astola, Adaptive window size image de-noising based on intersection of confidence intervals (ici) rule, J. Math. Imaging Vis. 16 (3) (2002) 223–235, doi:10.1023/A:1020329726980.
[21] R. Klinkenberg, T. Joachims, Detecting concept drift with support vector machines, in: Proceedings of the Seventeenth International Conference on Machine Learning, in: ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000, pp. 487–494.
[22] A. Niculescu-Mizil, R. Caruana, Predicting good probabilities with supervised learning, in: Proceedings of the 22Nd International Conference on Machine Learning, in: ICML 05, ACM, New York, NY, USA, 2005, pp. 625–632, doi:10.1145/1102351.1102430.
[23] M. Nagae, K. Suzuki, A neck mounted interface for sensing the swallowing activity based on swallowing sound, in: Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE, 2011, pp. 5224–5227, doi:10.1109/IEMBS.2011.6091292.
[24] H. Tsujimura, H. Okazaki, M. Yamashita, H. Doi, M. Matsumura, Non-restrictive measurement of swallowing frequency using a throat microphone, IEEJ Trans. Electron. Inf. Syst. 130 (2010) 376–382, doi:10.1541/ieejeiss.130.376.
[25] F. Eyben, M. Wöllmer, B. Schuller, Opensmile: The munich versatile and fast open-source audio feature extractor, in: Proceedings of the International Conference on Multimedia, in: MM '10, ACM, New York, NY, USA, 2010, pp. 1459–1462, doi:10.1145/1873951.1874246.
[26] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, ACM SIGKDD Explor. Newsl. 11 (1) (2009) 10–18.
[27] O. Mangasarian, D.R. Musicant, LSVM Software: active set support vector machine classification software, 2000, www.cs.wisc.edu/~musicant/lsvm/.